AE646: SCIENTIFIC MACHINE LEARNING FOR FLUID MECHANICS

PINN05: PHYSICS INFORMED NEURAL NETWORKS

TRANSFER LEARNING IN PINNS

Course Website: https://hello.iitk.ac.in/studio/ae646sem12324

Murali Damodaran
Aerospace Engineering
IIT Kanpur
dmurali@iitk.ac.in
ESB-317
0512-679-2408

PINNs SOLUTION OF ODE/PDE

Have seen that PINNs

-are meshless

-are discretization error free as derivatives are approximated using AD

-same approach for all ODEs/PDES

-gives a continuously differentiable solution in the domain of interest.

PINN gives a solution for an ODE/PDE

A different PINN must be created for another ODE/PDE or for the same ODE/PDE with a different parameter

Forward problem:

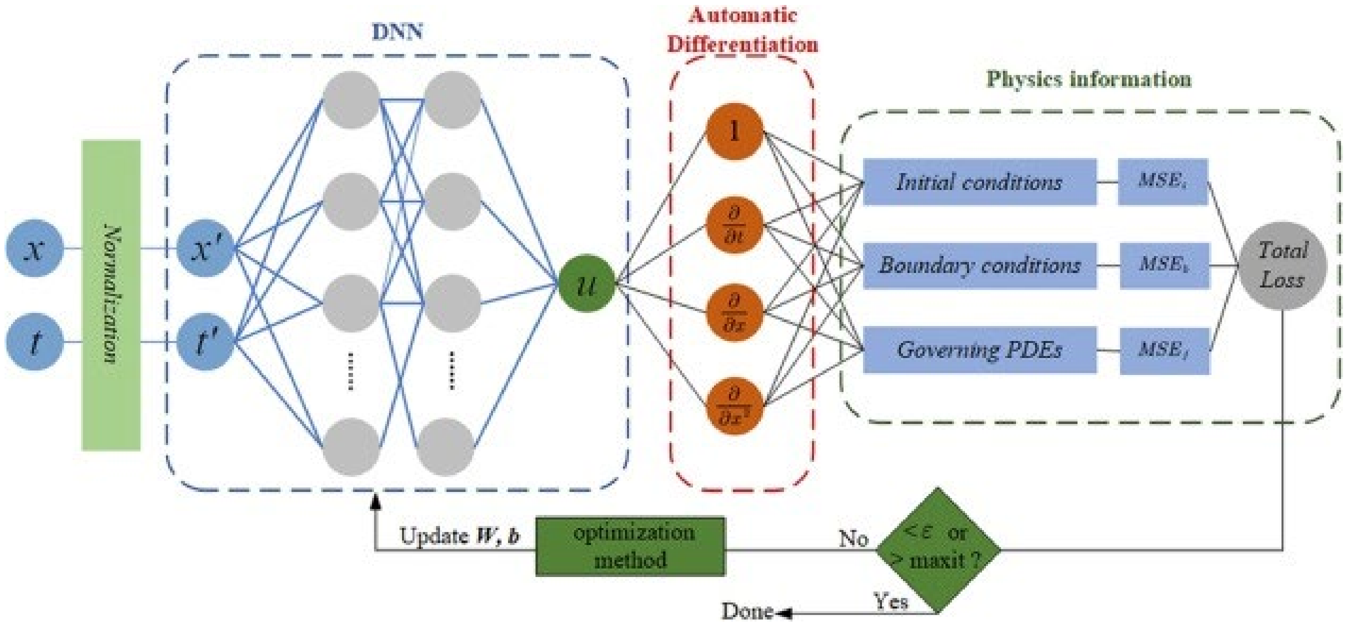
Parameter λ in the PDE/ODE are known and the unknown *u(x,t)* are to be determined for the given parameter and *f(x,t)*

$$PDE: \quad \lambda \frac{\partial^n u(x,t)}{\partial x^n} + f(x,t) = 0$$

λ is a parameter of the PDE which is known for forward problem

$$BC: \quad u(0,x) = \varphi(0,t)$$

$$IC: \quad u(x,0) = g(x,0)$$



PINNs SOLUTION OF ODE/PDE

Inverse problem:

Discover parameters in the PDE/ODE for a desired output, i.e., *Discover PDE parameter λ if measured data are known.*

Note- No information on Boundary conditions/Initial conditions

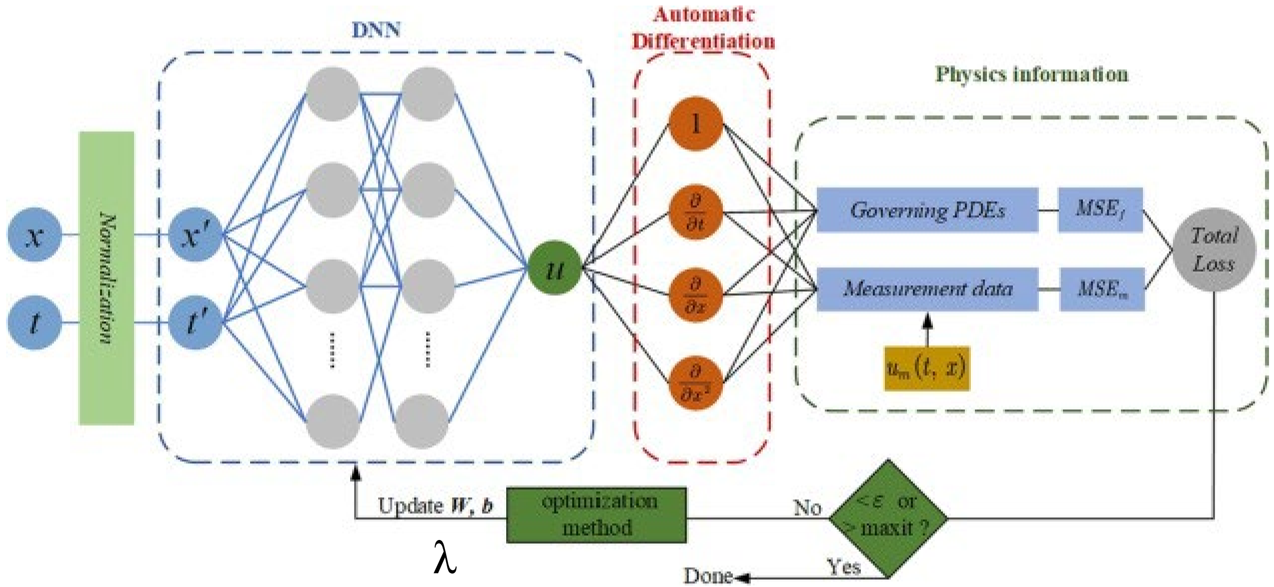
Modify the original PINN so that the **PDE parameter λ is treated as a hyperparameter and is optimised to fit the measured data.**

$$PDE: \quad \lambda \frac{\partial^n u(x,t)}{\partial x^n} + f(x,t) = 0$$

λ is a parameter of the PDE which must be **discovered for inverse problem**

$$BC: \quad u(0,x) = \varphi(0,t)$$

$$IC: \quad u(x,0) = g(x,0)$$



TRANSFER LEARNING IN PINNS

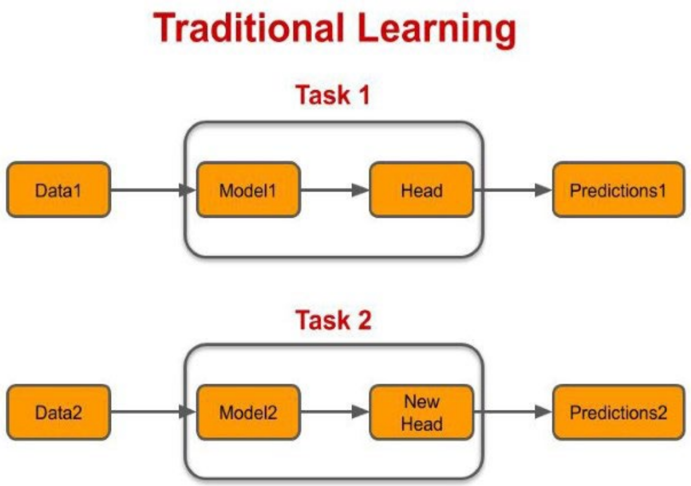
Transfer learning is **a process of pre-training a ANN/PINN on similar data to enhance performance for a new task instead of building the PINN from scratch.**

It is an **optimization** aimed at reducing training time and improving performance for predicting outcomes of new similar tasks

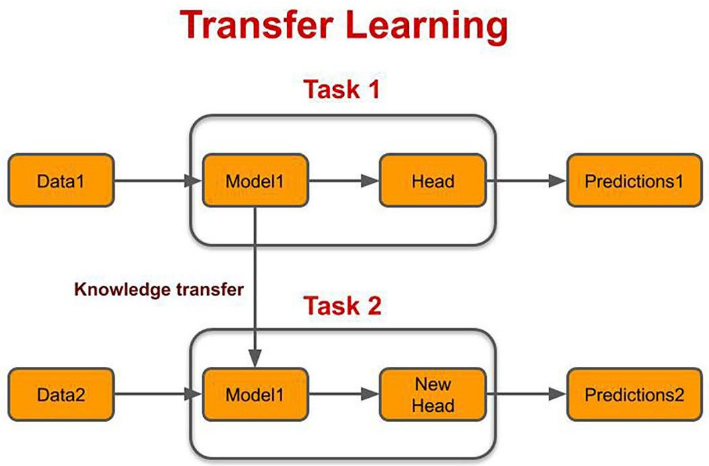
Approaches to Transfer Learning:

Feature Extraction TL:
Use the representations learned by a previous network to extract meaningful features from new samples and reuse this for training new network. (Only train the last layer/s of the network)

Fine Tuning TL:
Unfreeze a few of the top layers of a frozen model based network and jointly train both the newly-added layers and the last layers of the base model. (Train the whole network)



One ANN/PINN for Each Task



Transfer Knowledge from a Pre-trained ANN/PINN for a New Task

Images Source: https://www.topbots.com/transfer-learning-in-nlp/

TRANSFER LEARNING IN PINNS

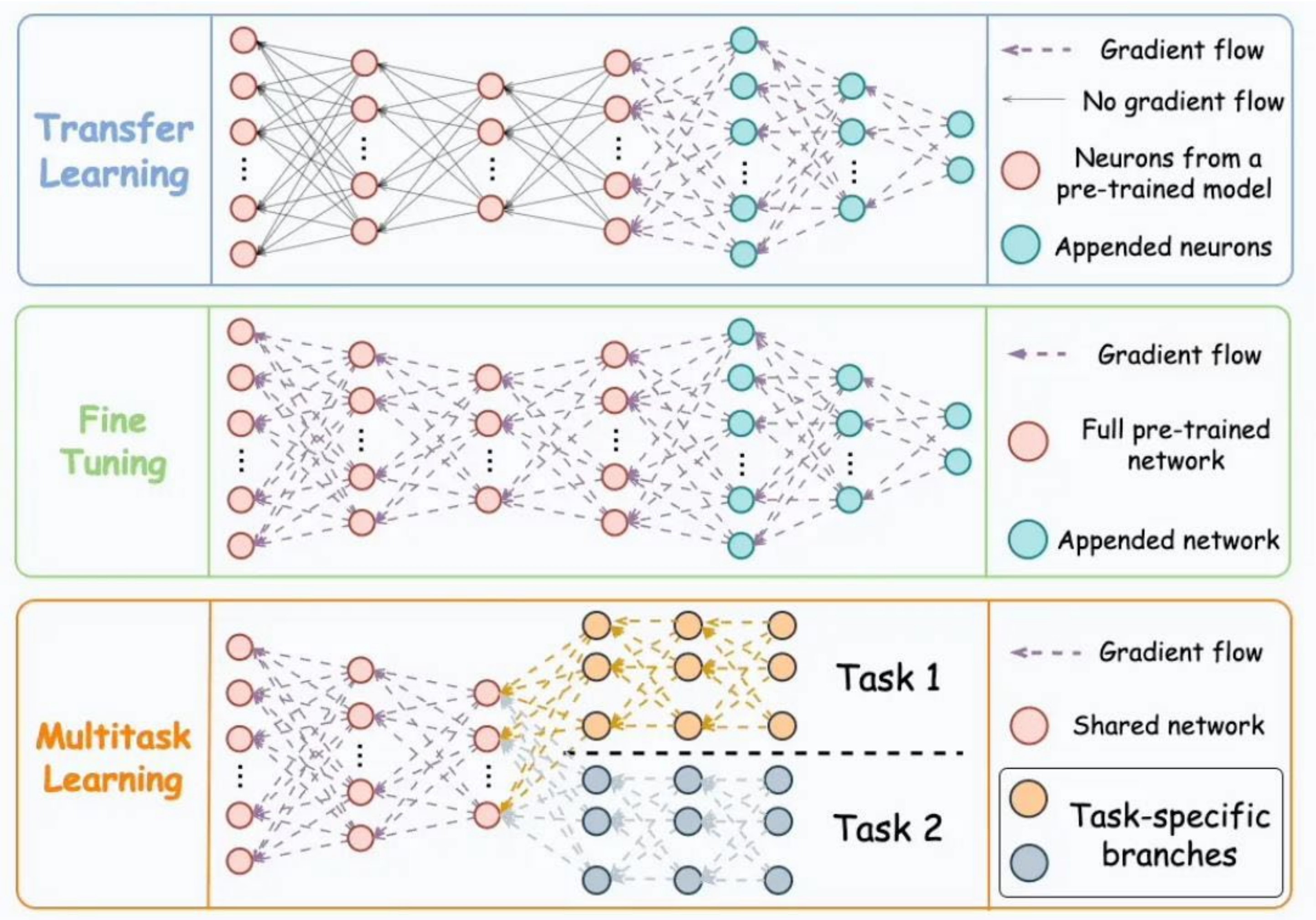


Image Source: https://blog.dailydoseofds.com/p/transfer-learning-vs-fine-tuning

Data Flow in a Feed-Forward Neural Network
The Feedforward Algorithm

R

The Forward Propagation

$$z^1 = W^1 X + b^1$$
$$z^2 = W^2 \sigma_1\left(z^1\right) + b^2$$
$$\dots$$
$$z^{L-1} = W^{L-1} \sigma_{L-2}\left(z^{L-2}\right) + b^{L-1}$$
$$z^L = W^L \sigma_{L-1}\left(z^{L-1}\right) + b^L$$
$$Y = z^L$$

$$z^1 = \overline{W}^1 X$$
$$z^2 = \overline{W}^2 \sigma_1\left(z^1\right)$$
$$\dots$$
$$z^{L-1} = \overline{W}^{L-1} \sigma_{L-2}\left(z^{L-2}\right)$$
$$Y = z^L = \overline{W}^L \sigma_{L-1}\left(z^{L-1}\right)$$

Training of the network with known values of Y for a given X requires optimization of weights and biases or the elements of the augmented weight matrix.

Consider a 2 Hidden Layer Fully Connected Neural Network:

$$z^1 = \overline{W}^1 X$$
$$z^2 = \overline{W}^2 \sigma_1\left(z^1\right) = \overline{W}^2 \sigma_1\left(\overline{W}^1 X\right)$$
$$z^3 = \overline{W}^3 \sigma_2\left(z^2\right) = \overline{W}^3 \sigma_2\left(\overline{W}^2 \sigma_1\left(\overline{W}^1 X\right)\right)$$
$$Y = \overline{W}^3 \sigma_2\left(\overline{W}^2 \sigma_1\left(\overline{W}^1 X\right)\right)$$

ONE SHOT TRANSFER LEARNING IN PINNS: ODE

The **general form** of an n - th order ODE:

$$F\left(t,\psi,\psi^{(1)},\psi^{(2)},.....,\psi^{(n-1)}\right)=\psi^{(n)}$$

$\psi^{(i)}=\dfrac{d^{(i)}\psi}{dt^{(i)}}$ is the i-th derivative of the solution $\psi(t)$

Non - homogeneous linear ODEs is represented as follows:

$$\hat{D}_n\psi = f(t);$$

$$\hat{D}_n\psi = \sum_{i=0}^{n} a_i(t)\psi^{(t)}$$

$\hat{D}_n$: differential operators; $a_i(t)$: coefficients

Initial Conditions ICs:

$$u_{ic} = \left[\psi_0,\psi_0^{(1)},\psi_0^{(2)},....,\psi_0^{(n-1)}\right]^T$$

Loss Function :

$$\left(\hat{D}_n\psi_\theta(t)-f(t)\right)^2+\left(\hat{D}_0\psi_\theta(t)-\psi_{IC}\right)^2$$

$$\hat{D}_0\psi=\left[\psi(0),\psi^{(1)}(0),\psi^{(2)}(0),....,\psi^{(n-1)}(0)\right]^T$$

Input $t\in\mathbb{R}^{t\times 1}$ is **transformed / mapped** to **output** $\psi_\theta(t)$ via a PINN which uses an ANN parameterized by weights and biases $\theta=\left[\theta_H,\theta_W,\theta_B\right]$

to form an approximate (PINN) solution to the ODE at time t as:

$$\psi(t)=H(t)_{\theta_H}W_{\theta_W}+B_{\theta_B}\approx H(t)_{\theta_H}\bar{W}_{\theta_W}$$

where

W_{θ_W} and B_{θ_B} : network weights and biases

$\bar{W}_{\theta_W}$: Augmented weight matrix absorbing the bias B_{θ_B}

$H(t)_{\theta_H}$: latent space consisting of hidden layers and activation functions

ONE SHOT TRANSFER LEARNING IN PINNS: ODE

$H \in \mathbb{R}^{t \times h}$: latent space composed of hidden layers and activation functions

Final weights layer

$W_{\theta_W} \in \mathbb{R}^{h \times q}$: weights

to consist of multiple outputs $\psi(t) \in \mathbb{R}^{t \times q}$

to satisfy and train ODEs with different linear operators i.e. $\hat{D}_n$ and different coefficients $a_i(t)$ and different initial conditions simultaneously.

After training with multiple ODEs weights for hidden layers are frozen.

Solution at time $\hat{t}$: $\psi(\hat{t}) = H(\hat{t}) W_{OUT}$

W_{OUT} : weights to be trained for new sets of ICs: ψ'_{IC}, f' and differential operators, $\hat{D}'_n$

Loss function for new ODEs unseen in Frozen H

$$\mathcal{L} = \mathcal{L}_{DE} + \mathcal{L}_{IC} = \left(\hat{D}'_n H W_{out} - f'(t)\right)^2 + \left(\hat{D}_0 H W_{out} - \psi'_{IC}\right)^2$$

(Note that superscript $'$ represents new conditions and not 1st-order time derivatives)

ONE SHOT TRANSFER LEARNING IN PINNS: ODE

ReX

Loss function for new ODEs unseen in H

$$\mathcal{L} = \mathcal{L}_{DE} + \mathcal{L}_{IC} = \left(\hat{D}_n' H W_{out} - f(t)\right)^2 + \left(\hat{D}_0 H W_{out} - \psi_{IC}'\right)^2$$

Taking derivatives of the Loss Function with respect to W_{OUT} results in:

$$\frac{\partial \mathcal{L}_{DE}}{\partial W_{OUT}} = 2\left(\hat{D}_n' H\right)^T \left(\hat{D}_n' H W_{OUT} - f'(t)\right)$$

$$\frac{\partial \mathcal{L}_{IC}}{\partial W_{OUT}} = 2\left(\bar{D}_0 H\right)^T \left(\bar{D}_0 H W_{OUT} - \psi_{IC}'\right)$$

Setting

$$\frac{\partial \mathcal{L}_{DE}}{\partial W_{OUT}} + \frac{\partial \mathcal{L}_{IC}}{\partial W_{OUT}} = 0 \quad \Leftarrow \text{for loss to be a minimum}$$

results in

$$W_{OUT} = \left(H^T D_H'^T \hat{D}_H + H^T D_H'^T \bar{D}_H\right)^{-1} \left(H^T \bar{D}_0^T f'(t) + H^T \bar{D}_0^T \psi_{IC}'\right)$$

This result shows that for or any fixed hidden states $H(t)$at a fixed time $\hat{t}$ and if the ODE is linear W_{OUT} can be computed analytically.

ONE SHOT TRANSFER LEARNING IN PINNS: PDEs

A **general linear 2nd - order PDE** in the x - t domain:

$$\left(D^t + D^x + D^{xt} + V(t,x)\right)\psi(x,t) = f(x,t)$$

where

$$D^t\psi = \sum_{i=1}^{2} a_i(t,x)\psi_t^{(i)}$$

$$D^x\psi = \sum_{i=1}^{2} b_i(t,x)\psi_x^{(i)}$$

$$D^{xt}\psi = D^{tx}\psi = c(x,t)\psi_{xt}$$

where

a,b,c,f and V are continuous functions of x and t.

$$\psi_x^{(i)} = \frac{\partial^{(i)}\psi}{\partial x^{(i)}};\psi_{xt} = \frac{\partial^2\psi}{\partial x\partial t}$$

The **Total Loss Function** $\mathcal{L}$ is composed of the losses associated with the PDE, IC and BC i.e.:

$$\mathcal{L} = \mathcal{L}_{PDE} + \mathcal{L}_{IC} + \mathcal{L}_{BCs}$$

$$\mathcal{L} = \left(\hat{D}_n\psi - f(t,x)\right)^2 + \left(\psi(0,x) - g(x)\right)^2 + \sum_{\mu=L,R}\left(\psi(t,\mu) - B_\mu(t)\right)^2$$

where

$$\hat{D}_n = \left(D^t + D^x + D^{xt}\right) + V(t,x))$$

B_L and B_R : left $(\mu = L)$ and right $(\mu = R)$ boundary conditions

$g(x)$: initial condition at $t = 0$

ONE SHOT TRANSFER LEARNING IN PINNS-PDEs

ReX

For the output solution $\psi = HW_{OUT}$

$$\frac{\partial \mathcal{L}}{\partial W_{OUT}} = 0 \Leftarrow \text{for minimising the loss}$$

Considering each component of the Loss:

$$\frac{\partial \mathcal{L}_{PDE}}{\partial W_{OUT}} = 2H^T \hat{D}^T \left(\hat{D}HW_{OUT} - f(t,x)\right)$$

$$\frac{\partial \mathcal{L}_{IC}}{\partial W_{OUT}} = 2H_0^T \left(H_0 W_{OUT} - g(0,x)\right)$$

where $H_0 = H(0,x)$

$$\frac{\partial \mathcal{L}_{BC}}{\partial W_{OUT}} = \sum_{\mu=L,R} 2H_\mu^T \left(H_\mu W_{OUT} - B_\mu(t)\right)$$

where $H_\mu = H(t,\mu)$

assuming Dirichelet BCs here

It can be shown that $\frac{\partial \mathcal{L}}{\partial W_{OUT}} = 0$ results in

$$W_{OUT} = \left(H^T \hat{D}^T \hat{D}H + \sum_{\mu=0,L,R} H_\mu^T H_\mu\right)^{-1} \left(H^T \hat{D}^T f(t,x) + \sum_{\mu=0,L,R} H_\mu^T Q_\mu(t,x)\right)$$

$Q_0 = g(x) \Leftarrow$ initial condition

$Q_L = B_L(t) \Leftarrow$ Left Boundary Condition

$Q_R = B_R(t) \leftarrow\Leftarrow$ Right Boundary Condition

For linear 2nd Order PDEs
it is possible to determine the
weight analyically and hence the solution to the PDE

ONE SHOT TRANSFER LEARNING IN PINNS for PDEs

Consider a given PDE:

$\mathcal{D}(\mathbf{u},\mathbf{x}) = f(\mathbf{x}), \mathbf{x} \in \Omega \subset \mathbb{R}^d$

$\mathcal{B}(\mathbf{u},\mathbf{x}) = g(\mathbf{x}), \mathbf{x} \in \partial\Omega$

$\mathcal{D}:$ Differential Operator in Ω

$\mathcal{B}:$ Boundary Operator on $\partial\Omega$

Neural Network Surrogate for $\mathbf{u}$:

$\phi(\mathbf{x};\theta) = Y = \sigma_3 \bar{W}^3 \sigma_2\left(\bar{W}^2 \sigma_1\left(\bar{W}^1 X\right)\right)$

$\theta = \left\{\bar{W}^3, \bar{W}^2, \bar{W}^1\right\}$

with activation functions $\sigma_3 = \sigma_2 = \sigma_1 = \sigma(x)$

Interior Training Samples: $\left\{\mathbf{x}_i\right\}_{i=1}^{n_I}$

Initial/Boundary Samples: $\left\{\tilde{\mathbf{x}}_i\right\}_{i=1}^{n_b}$

Loss Function: $\mathcal{L} = \frac{\lambda}{2n_I}\sum_{i=1}^{n_I}\left\|\mathcal{D}\left[\phi(\mathbf{x}_i;\theta),\mathbf{x}_i\right] - f(\mathbf{x}_i)\right\|_2^2$

$$+\frac{1}{2n_b}\sum_{j=1}^{n_b}\left\|\mathcal{B}\left[\phi(\tilde{\mathbf{x}}_j;\theta),\tilde{\mathbf{x}}_j\right] - g(\tilde{\mathbf{x}}_j)\right\|_2^2$$

$\lambda > 0:$ hyperparameter for balancing the 2 contributions

$\min_{\theta} \mathcal{L}$

Network solves a single PDE,
i.e. one Neural Network to one PDE
even if PDEs may be similar and shared information with base pre-trained NN exists.

ONE-SHOT TRANSFER LEARNING IN PINNS FOR ODEs and PDEs

Consider a class of PDEs with different f_ε and g_ε :

i.e.,

$$\mathcal{D}(\mathbf{u},\mathbf{x}) = \{f_\varepsilon(\mathbf{x})\}_\varepsilon, \mathbf{x} \in \Omega \subset \mathbb{R}^d$$

$$\mathcal{B}(\mathbf{u},\mathbf{x}) = \{g_\varepsilon(\mathbf{x})\}_\varepsilon, \mathbf{x} \in \partial\Omega$$

Note: ε is a parameter which changes for different PDEs

Approximate PINN solution $\phi(\mathbf{x};\theta_\varepsilon)$ for a specific ε shares some of the network parameters i.e., $\{\bar{\mathbf{W}}^2, \bar{\mathbf{W}}^1, \mathbf{b}^3\}$ from a pre-trained model $\phi(\mathbf{x};\theta_\varepsilon)$ for a given ε keeping $\bar{\mathbf{W}}^2, \bar{\mathbf{W}}^1$ and the bias $\mathbf{b}^3$ associated with $\bar{\mathbf{W}}^3$ **frozen** and leaving only the weight $\mathbf{W}^3$ associated with $\bar{\mathbf{W}}^3$ to be **trained** for other ε.

This is what **one-shot transfer learning** accomplishes.

Example PDE:

$$\frac{\partial u(t,\mathbf{x})}{\partial t} + \frac{\partial}{\partial \mathbf{x}} a(\mathbf{x}) \frac{\partial u(\mathbf{x},t)}{\partial \mathbf{x}} = f_\varepsilon(t,x) \text{ in } (0,1) \cup \Omega$$

$$u(t,\mathbf{x}) = g_\varepsilon(t,\mathbf{x}) \text{ on } (0,1) \cup \partial\Omega$$

$$u(0,\mathbf{x}) = h_\varepsilon(\mathbf{x}) \text{ in } \Omega$$

$$a(\mathbf{x}) = 1 + \frac{1}{2}\|\mathbf{x}\|_2$$

$$u_{EXACT} = u_\varepsilon(t,\mathbf{x}) = e^{\left(\|\mathbf{x}\|_2 \sqrt{1-t} + \varepsilon(1-t)\right)}$$

$u_\varepsilon, f_\varepsilon, g_\varepsilon$ and h_ε are differentiable w.r.t. ε

Set $f_\varepsilon, g_\varepsilon$ and h_ε to be $u_\varepsilon(t,\mathbf{x}) = e^{\left(\|\mathbf{x}\|_2 \sqrt{1-t} + \varepsilon(1-t)\right)}$

Pre-train the neural network with $\varepsilon=0$

Then use transfer learning for other $\varepsilon = 0.5, 2.0, \ldots$

Reference:

Paper: Shaan Desai, Marios Mattheakis, Hayden Joy, Pavlos Protopapas, and Stephen Roberts., "One-shot transfer learning of physics-informed neural Networks". *ICML AI4Science Workshop*, 2022. https://arxiv.org/abs/2110.11286

Codes: https://github.com/shaandesai1/TransferDE

SVD TRANSFER LEARNING IN PINNS FOR ODEs and PDEs

Consider a class of PDEs with different f_ε and g_ε :

i.e.,

$$\mathcal{D}(\mathbf{u},\mathbf{x})=\{f_\varepsilon(\mathbf{x})\}_\varepsilon,\mathbf{x}\in\Omega\subset\mathbb{R}^d$$

$$\mathcal{B}(\mathbf{u},\mathbf{x})=\{g_\varepsilon(\mathbf{x})\}_\varepsilon,\mathbf{x}\in\partial\Omega$$

A finite difference or a finite volume discretization of a linear PDE will result in a system of algebraic equations of the form $\mathbf{Ax}=\mathbf{b}$ where matrix $\mathbf{A}$ is the discretization of the operators $\mathcal{D}$ and $\mathcal{B}$

The SVD of matrix $\mathbf{A}$ is: $\mathbf{A}=\mathbf{U\Sigma V^T}$

$\mathbf{x}=\mathbf{A}^\dagger\mathbf{b}=\mathbf{V\Sigma U^T b}$

$\mathbf{A}^\dagger$: pseudo-inverse of $\mathbf{A}$

$\mathbf{U}$ and $\mathbf{V}$: bases of $\mathbf{A}$ $\simeq$ $\bar{\mathbf{W}}^2=\mathbf{U\Sigma V^T}$

$\mathbf{\Sigma}$: Diagonal matrix of Singular Values of $\bar{\mathbf{W}}^2$

Freeze $\{\mathbf{U},\mathbf{V}\}$ for $\bar{\mathbf{W}}^2$

including the bias term associated with $\bar{\mathbf{W}}_3$

Trainable parameters: $\{\mathbf{\Sigma},\bar{\mathbf{W}}_3,\bar{\mathbf{W}}_1,\mathbf{b}_3\}$

SVD TRANSFER LEARNING IN PINNS FOR ODEs and PDEs

Consider a 3 Hidden Layer Fully Connected Neural Network with 2 hidden layers

$$z^1 = \bar{W}^1 X$$

$$z^2 = \bar{W}^2 \sigma_1\left(z^1\right) = \bar{W}^2 \sigma_1\left(\bar{W}^1 X\right)$$

$$z^3 = \bar{W}^3 \sigma_2\left(z^2\right) = \bar{W}^3 \sigma_2\left(\bar{W}^2 \sigma_1\left(\bar{W}^1 X\right)\right)$$

$$Y = z^3 = \bar{W}^3 \sigma_2\left(\bar{W}^2 \sigma_1\left(\bar{W}^1 X\right)\right)$$

Freeze $\{\mathbf{U}, \mathbf{V}\}$ for $\bar{\mathbf{W}}^2$ based on output θ_ε including the bias term associated with $\bar{\mathbf{W}}_3$

Trainable parameters:

$\{\boldsymbol{\Sigma}, \bar{\mathbf{W}}_3, \bar{\mathbf{W}}_1, \mathbf{b}_3\}$

$\boldsymbol{\Sigma}$: singular values of $\bar{\mathbf{W}}_2$

Instead of training $\bar{\mathbf{W}}_2$, train its singular values

During training (optimization of loss function) a constraint is imposed to ensure that singular values of $\boldsymbol{\Sigma}$ are always > 0.

Hence training of singular values become a constrained optimization problem.

Reference:

Y. Gao, K. C. Cheung and M. K. Ng, "SVD-PINNs: Transfer Learning of Physics-Informed Neural Networks via Singular Value Decomposition," *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, Singapore, Singapore, 2022, pp. 1443-1450, doi: 10.1109/SSCI51031.2022.10022281

MULTI-HEAD TRANSFER LEARNING IN PINNS FOR ODEs and PDEs: L-HYDRA

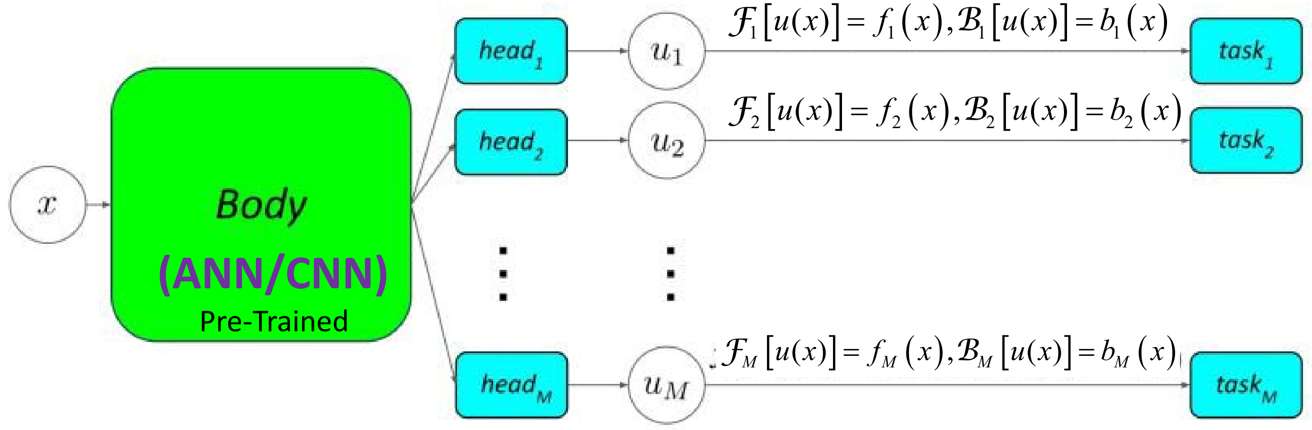
ODE / PDE : $\mathcal{F}_k\left[u_k(x)\right] = f_k(x), x \in \Omega_k$

Boundary/Initial Condition: $\mathcal{B}_k\left[u_k(x)\right] = b_k(x), x \in \partial\Omega_k$

x : spatio-temporal coordinates of D_u dimensions

$\mathcal{F}_k$: General Differential Operators

$\mathcal{B}_k$: General Boundary/Initial Condition terms



M Tasks: $\{T_k\}_{k=1}^{M}$
PINN regression problems (solution of ODEs/PDEs) associated with
M Data: $D_k, k = 1, \ldots, M$

Image Source From the Paper: Zongren Zou, George Em Karniadakis, "L-HYDRA: Multi-Head Physics-Informed Neural Networks", http://dx.doi.org/10.48550/arXiv.2301.02152

Neural Networks can be split into a body and multiple heads as shown resulting in Multi-Head PINNs or MH-PINNs

Body could be the *nonlinear portion of the feed forward neural network* while the **head** could be the *last linear layer of the network or a few last layers of the network.*

Body could be a *convolutional neural network* and the **head** ca *feed forward neural network.*

MULTI-HEAD TRANSFER LEARNING IN PINNS FOR ODEs and PDEs: L-HYDRA

Body : $\Phi: \mathbb{R}^{D_x} \to \mathbb{R}^L$

D_x : spatial-temporal dimensions

$\Phi(x) = \left[\phi^1(x), \phi^2(x), \ldots, \phi^L(x)\right]^T$

$\phi : \mathbb{R}^{D_x} \to \mathbb{R}$

function parametrized by the neural network

with parameter θ

and

Head : $H_k \in \mathbb{R}^{L+1}$,

$H_k = \left[h_k^0, h_k^1, \ldots, h_k^L\right]^T$

is the number of neurons in the last layer of the Body.

The approximate solution is then given as:

$\hat{u}_k = h_k^0 + \sum_{l=1}^{L} h_k^l \phi^l(x)$ for all $x \in \Omega$

Conventional PINN solves each task independently of each other. Hence the M solutions are uncorrelated.

If the M Tasks: $\{T_k\}_{k=1}^{M}$ are treated together and connected to the MH-PINNs then solutions are correlated as the body provides a set of basis functions for u_k

Loss Function:

$$L\left(\left\{D_k^f\right\}_{k=1}^{M};\theta,\left\{H_k\right\}_{k=1}^{M}\right)=\frac{1}{M}\sum_{k=1}^{M}L_k\left(D_k;\theta,H_k\right)$$

where

$$D_k=\left\{D_k^f,D_k^b,D_k^u\right\}$$

$$D_k^f=\left\{x_k^i,f_k^i\right\}_{i=1}^{N_k^f},D_k^b=\left\{x_k^i,b_k^i\right\}_{i=1}^{N_k^b}$$

$D_k^u=\left\{x_k^i,u_k^i\right\}_{i=1}^{N_k^u}$ are the sparse sensor data that is available for T_k

$$L_k\left(D_k;\theta,H_k\right)=\frac{\omega_k^f}{N_k^f}\sum_{i=1}^{N_k^f}\left\|F_k\left(\hat{u}_k\left(x_k^i\right)\right)-f_k^i\right\|^2+\frac{\omega_k^b}{N_k^b}\sum_{i=1}^{N_k^b}\left\|B_k\left(\hat{u}_k\left(x_k^i\right)\right)-b_k^i\right\|^2$$

$$+\frac{\omega_k^u}{N_k^u}\sum_{i=1}^{N_k^u}\left\|\hat{u}_k\left(x_k^i\right)-u_k\left(x_k^i\right)\right\|^2+R\left(\theta,H_k\right)$$

$R\left(\theta,H_k\right)$: Regularisation to mitigate overfitting

$\omega_k^f;\omega_k^b$ and ω_k^u are the weights to balance the various loss terms

$\|\bullet\|$: norm such as 2-norm or any other.

ONE SHOT TRANSFER LEARNING IN PINNS FOR ODEs and PDEs

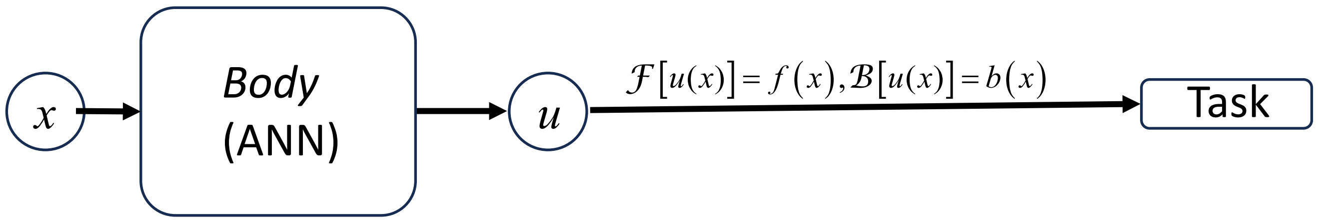
ODE / PDE : $\mathcal{F}\left[u(x)\right] = f(x), x \in \Omega$

Boundary/Initial Condition: $\mathcal{B}\left[u(x)\right] = b(x), x \in \partial\Omega$

x : spatio-temporal coordinates of D dimensions

$\mathcal{F}$: General Differential Operators

$\mathcal{B}$: General Boundary/Initial Condition terms



The **one-shot transfer learning in PINNs** could be viewed as a MH-PINN in the sense that

the **heads were used to pre-train the PINN with similar ODEs/PDEs , BCs and Ics and after that the heads were abandoned**

and

the **body with all the pre-trained information is retained and used in transfer learning for new unseen but similar** ODEs/PDEs with different BCs/ICs

TRANSFER LEARNING IN PINNS

Example :

Consider Solving a 2D Poisson Equation in a unit square domain i.e.,

$$\nabla^2 u(x,y) = f(x,y); (x,y) \in [0,1]\times[0,1]$$

with Dirichelet Boundary Conditions

$$u(0,y) = u(1,y) = u(x,0) = u(x,1) = 0$$

$\tilde{u}$: PINN solution

$$\tilde{u} = \left[\bar{W}^{L}\sigma_{L-1}(\bar{W}^{L-1}\sigma_{L-2}(\cdots\sigma_1(\bar{W}^{1}\mathbf{x}))\right]$$

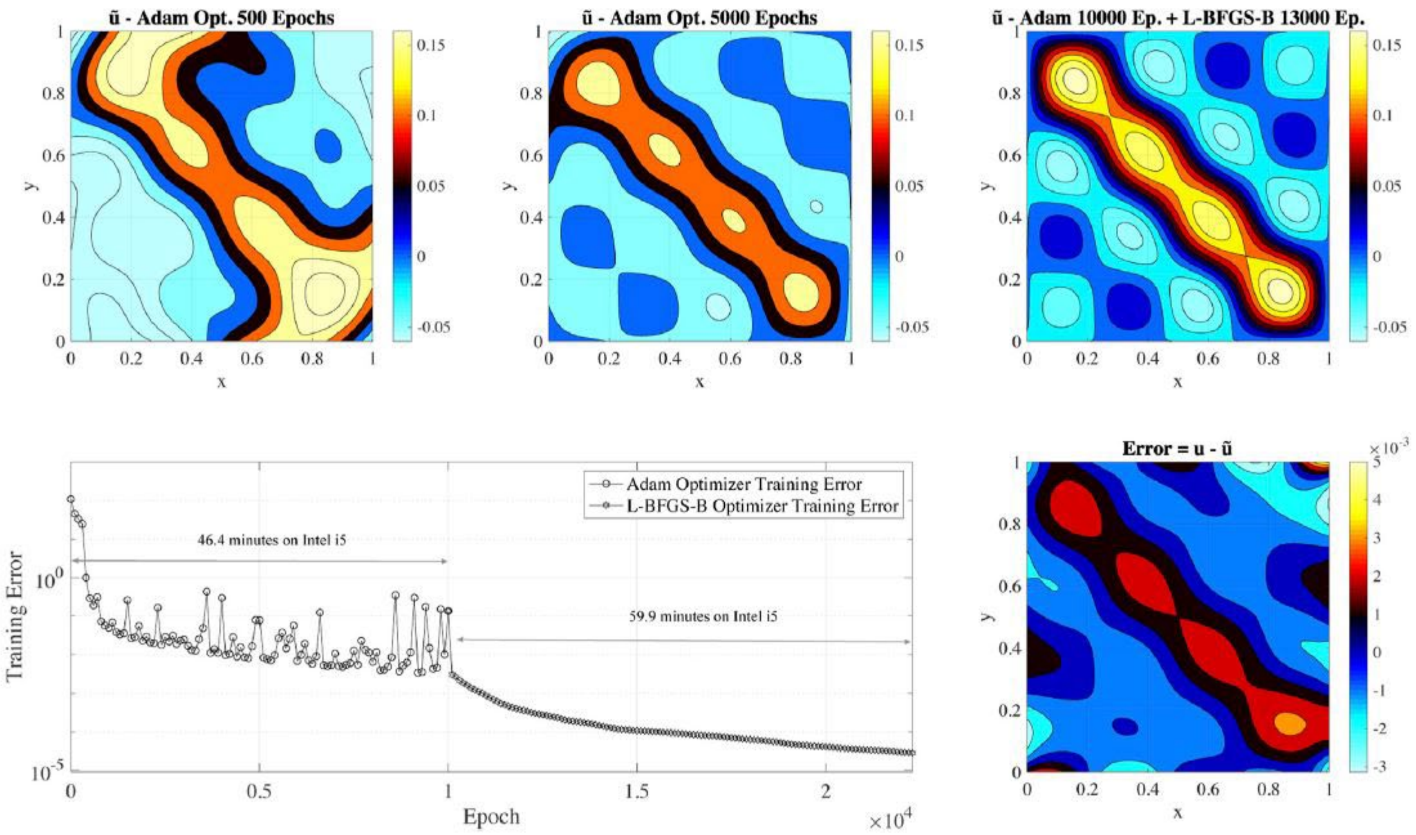
Residual: $r = \nabla^2\tilde{u}(x,y) - f(x,y)$

Loss Function: $\mathcal{L} = \frac{1}{N_{x_i,y_i}}\sum_{i=1}^{N_{x_i,y_i}} \left\| r(x_i,y_i) \right\|^2$

Network has 4 Hidden Layers with 50 neurons per layer and σ is the tanh activation function. 128×128 collocation points and 4000 points on the boundary

Optimise the Loss Function to obtain the solution for a given $f(x,y)$

Consider a smooth function: $f(x,y) = \frac{1}{4}\sum_{k=1}^{4}(-1)^{k+1}2k\sin(k\pi x)\sin(k\pi y)$



Images Source and **Paper:** Markidis S (2021) The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers? Frontiers in Big Data 4:669097. https://doi.org/10.3389/fdata.2021.669097

TRANSFER LEARNING IN PINNS FOR ODEs and PDEs

Consider another smooth function:

$$f(x,y) = 10x(x-1)(y-1) - 2\sin(\pi x)\sin(\pi y) + 5(2\pi x\sin(\pi y))$$

Transfer Learning from the PINN of the earlier smooth function PINN to the current case.

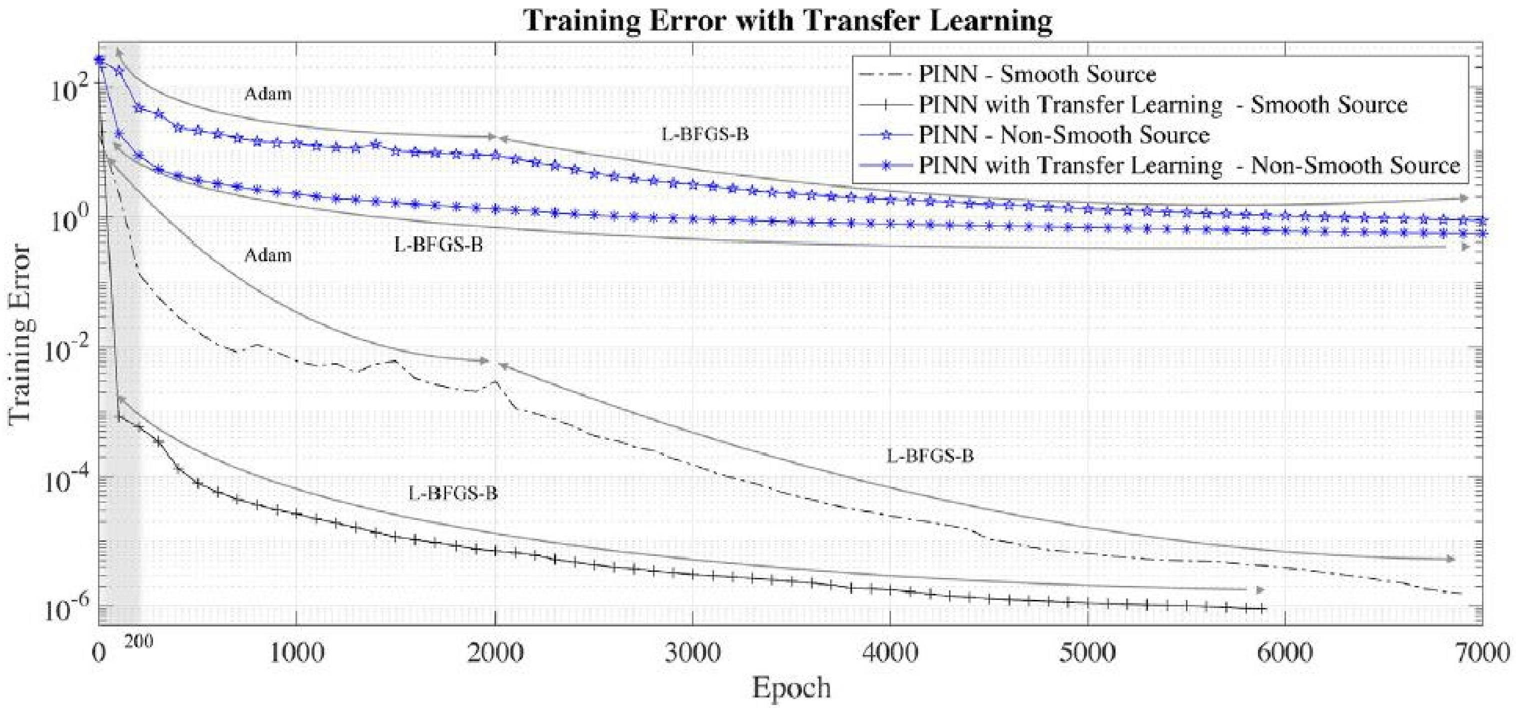
Consider a non-smooth function i.e.,

$$f(x,y) = \begin{cases} 0 & \text{everywhere} \\ 1 & (x-0.5)^2 + (y-0.5)^2 \le 0.2 \end{cases}$$

Consider another non-smooth function i.e.,

$$f(x,y) = \begin{cases} 0 & \text{everywhere} \\ -10 & (x-0.7)^2 + (y-0.7)^2 \le 0.1 \end{cases}$$

Transfer Learning from the PINN of the earlier non-smooth function PINN to the current case



Images Source and **Paper:** Markidis S (2021) The Old and the New: Can Physics-Informed Deep-Learning Replace Traditional Linear Solvers? Frontiers in Big Data 4:669097. https://doi.org/10.3389/fdata.2021.669097

Euler-Implicit Transfer Learning

Consider the 1 - D Viscous Burger's Eqn

$$u_t + uu_x = \nu u_{xx}, (t, x) \in (0, t_f) \times (0,1)$$

$$u(0, x) = u_0(x), x \in [0,1]$$

$$u(t,0) = u(t,1) = 0, t \in [0, t_f]$$

$\hat{u}(x,t)$: Solution Using PINN

Most of the attempts for the PINN solution scattered the neurons in the entire x - t domain of interest and obtained the solution by optimisation of the PINN loss function.

Consider Backward Euler or Euler - Implicit Scheme:

Refresher for Finite Difference Schemes for 1-D PDEs: https://en.wikipedia.org/wiki/Backward_Euler_method

$$\hat{u}^{(0)}(x) = u_0(x) \quad \Leftarrow \text{initial condition}$$

$$\hat{u}^{(k)} = \hat{u}^{(k-1)} + h_t \left(\nu \hat{u}_{xx}^{(k)} - \hat{u}^{(k)} \hat{u}_x^{(k)} \right)$$

$$\hat{u}^{(k)}(0) = \hat{u}^{(k)}(1) = 0 \quad \Leftarrow \text{boundary conditions}$$

$u^{(k)}(x) \approx u\left(t^{(k)}, x\right)$ at each time $t^{(k)} = kh_t$,

where $k = 0,1,2,...n_t$ and time step size: $h_t = \dfrac{1}{n_t}$

Euler-Implicit Transfer Learning

Train PINN from Initial Condition :

$$\mathcal{L}_0 = \frac{1}{n_s}\sum_{s=1}^{n_s}\left\|\hat{u}^{(0)}(x_s) - u_0(x_s)\right\|^2$$

for n_s collocation points/samples $0 \le x_s \le 1$

Transfer the knowledge from PINN network $\mathcal{N}^{(k-1)}$ to **PINN Network** $\mathcal{N}^{(k)}$

$\mathcal{N}^{(k)} \leftarrow \mathcal{N}^{(k-1)}$

Then train the $\mathcal{N}^{(k)}$ by minimizing the loss function

$$\mathcal{L} = \frac{1}{n_s - 2}\sum\left\|\hat{u}^{(k)} - \hat{u}^{(k-1)} - h_t\left(\nu\hat{u}^{(k)}_{xx} - \hat{u}^{(k)}\hat{u}^{(k)}_x\right)\right\|^2 + \frac{1}{2}\left(\left\|\hat{u}^{(k)}(0)\right\|^2 + \left\|\hat{u}^{(k)}(1)\right\|^2\right)$$

Reset / Update index k

Repeat knowledge transfer i.e., $\mathcal{N}^{(k)} \leftarrow \mathcal{N}^{(k-1)}$ and minimization of Loss Function $\mathcal{L}$ for updated index k

Consider Backward Euler or Euler - Implicit Scheme:

$\hat{u}^{(0)}(x) = u_0(x) \quad \Leftarrow$ initial condition

$\hat{u}^{(k)} = \hat{u}^{(k-1)} + h_t\left(\nu\hat{u}^{(k)}_{xx} - \hat{u}^{(k)}\hat{u}^{(k)}_x\right)$

$\hat{u}^{(k)}(0) = \hat{u}^{(k)}(1) = 0 \quad \Leftarrow$ boundary conditions

$u^{(k)}(x) \approx u\left(t^{(k)}, x\right)$ at each time $t^{(k)} = kh_t$,

where $k = 0, 1, 2, \ldots n_t$ and time step size: $h_t = \dfrac{1}{n_t}$

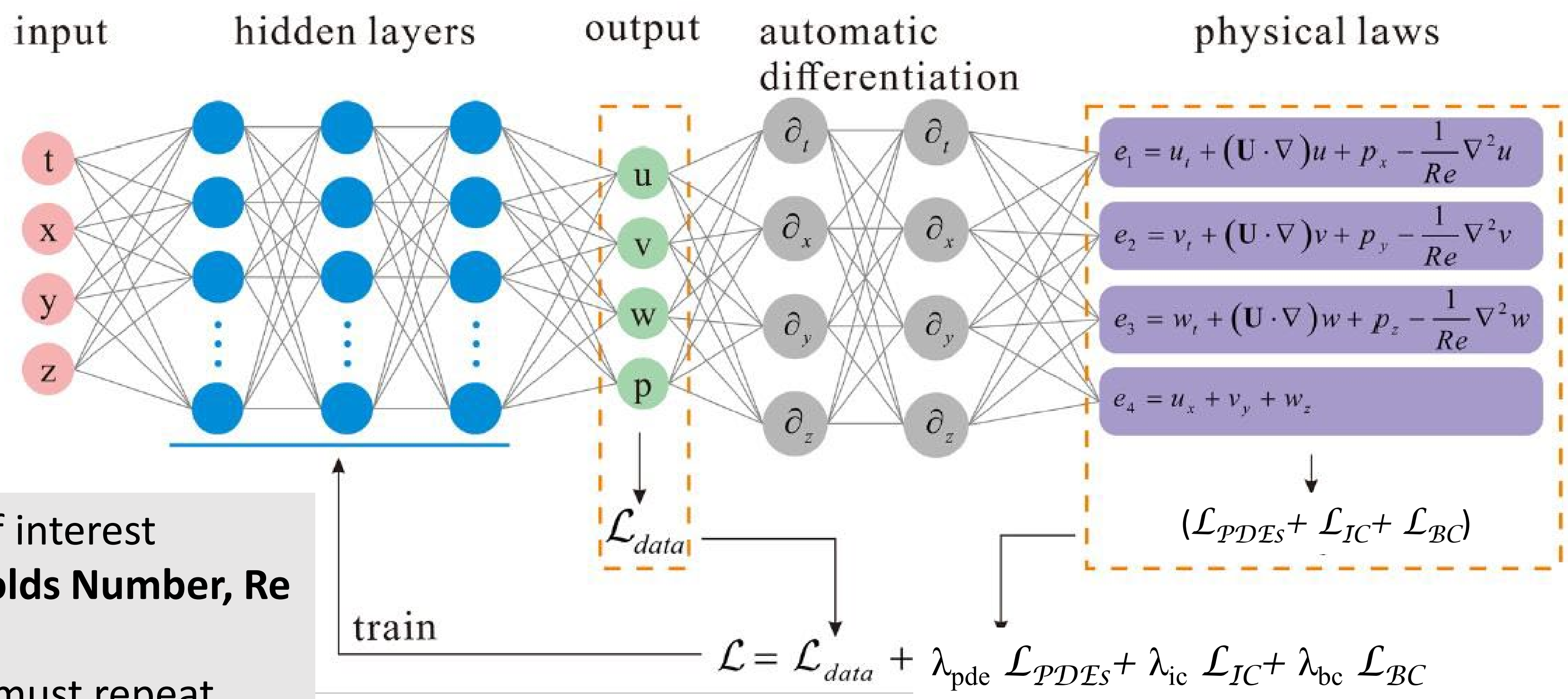
This approach results in a sequence of PINNs each giving an estimate of $\hat{u}^{(k)}(x) \approx u\left(t^{(k)}, x\right)$

Storage is reduced as only $\mathcal{N}^{(k)}$ and $\mathcal{N}^{(k-1)}$ needs to be stored.

Paper: Vitória Biesek, Pedro Henrique de Almeida Konzen, "Burgers' PINNs with Implicit Euler Transfer Learning" https://arxiv.org/abs/2310.15343
Conference paper XXVI ENMC/XIV ECTM 2023, Nova Friburgo, Brazil

R

PINN FOR 3D INCOMPRESSIBLE NAVIER-STOKES EQUATIONS



Estimate Physics Loss from all neurons, loss in Initial conditions and loss in Boundary Conditions

The parameter of interest here is **the Reynolds Number, Re**

For each **Re** one must repeat the same PINN prediction

Image Source: Hongping Wang, Yi Liu and Shizhao Wang, "Dense velocity reconstruction from particle image velocimetry/particle tracking velocimetry using a physics-informed neural network" (https://doi.org/10.1063/5.0078143) (Adapted abd Modified)

Question: *Can a pre-trained PINN for a task be used to generate predictions for a new task?*

PINN FOR 3D INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

The parameter of interest
here is **the Reynolds Number, Re**

For each **Re** one must repeat
the same PINN prediction.

Use **Transfer Learning**
to overcome this inflexibility

1. Set up Tasks for say
Re=10, 100, 500, 1000, 4000, 12000

2. Choose a **base task**, say Re=100, use
PINN from scratch and once converged
the PINN can be used to predict flow field for Re=100

3. Use the converged weights and biases from the pre-trained
Re=100 **base task** PINN wholly (all hidden layers) or partly (restricted to initial hidden layers) of the pre-trained Re=100 PINN for training
the new **target task** say Re=10 or Re=500

R

The Forward Propagation

$$z^1 = W^1 X + b^1$$
$$z^2 = W^2 \sigma_1\left(z^1\right) + b^2$$
$$\ldots$$
$$z^{L-1} = W^{L-1} \sigma_{L-2}\left(z^{L-2}\right) + b^{L-1}$$
$$z^L = W^L \sigma_{L-1}\left(z^{L-1}\right) + b^L$$
$$Y = z^L$$

$$z^1 = \overline{W}^1 X$$
$$z^2 = \overline{W}^2 \sigma_1\left(z^1\right)$$
$$\ldots$$
$$z^{L-1} = \overline{W}^{L-1} \sigma_{L-2}\left(z^{L-2}\right)$$
$$z^L = \overline{W}^L \sigma_{L-1}\left(z^{L-1}\right)$$
$$Y^L = z^L$$

Training of the network with known values of Y for a given X requires optimization of weights and biases or the elements of the augmented weight matrix.

TRANSFER LEARNING IN PINNS

$$\left.\begin{aligned} z^1 &= \bar{W}^1 X \\ z^2 &= \bar{W}^2 \sigma_1\left(z^1\right) \\ z^3 &= \bar{W}^3 \sigma_2\left(z^2\right) \\ &\vdots \\ z^{N_T} &= \bar{W}^{N_T} \sigma_{N_T-1}\left(z^{N_T-1}\right) \end{aligned}\right\}$$

$$\begin{aligned} &\vdots \\ z^{L-1} &= \bar{W}^{L-1} \sigma_{L-2}\left(z^{L-2}\right) \\ z^L &= \bar{W}^L \sigma_{L-1}\left(z^{L-1}\right) \\ Y^L &= \sigma_{L-1}\left(z^L\right) \end{aligned}$$

N_T layers of base transferred to N_T of target

Transfer pre-trained parameters (Weights and Biases) of 1st N_T **Base Task** starting from the 1st hidden layer for re-use as parameters for the 1st N_T hidden layers for the **Target Task.**

N_T : parameter for using part or whole of the pre-trained network parameters

TARGET TASK

$$\left\{\begin{aligned} z^1 &= \bar{W}^1 X \\ z^2 &= \bar{W}^2 \sigma_1\left(z^1\right) \\ z^3 &= \bar{W}^3 \sigma_2\left(z^2\right) \\ &\vdots \\ z^{N_T} &= \bar{W}^{N_T} \sigma_{N_T-1}\left(z^{N_T-1}\right) \end{aligned}\right.$$

$$\begin{aligned} z^{N_T+1} &= \bar{W}^{N_T+1} \sigma_{N_T}\left(z^{N_T}\right) \\ &\vdots \\ z^{L-1} &= \bar{W}^{L-1} \sigma_{L-2}\left(z^{L-2}\right) \\ z^L &= \bar{W}^L \sigma_{L-1}\left(z^{L-1}\right) \\ Y^L &= \sigma_{L-1}\left(z^L\right) \end{aligned}$$

TRANSFER LEARNING IN PINNS

BASE TASK

$$z^1 = \bar{W}^1 X$$
$$z^2 = \bar{W}^2 \sigma_1\left(z^1\right)$$
$$z^3 = \bar{W}^3 \sigma_2\left(z^2\right)$$
$$\vdots$$
$$z^{N_T} = \bar{W}^{N_T} \sigma_{N_T-1}\left(z^{N_T-1}\right)$$
$$\vdots$$
$$z^{L-1} = \bar{W}^{L-1} \sigma_{L-2}\left(z^{L-2}\right)$$
$$z^L = \bar{W}^L \sigma_{L-1}\left(z^{L-1}\right)$$
$$Y^L = \sigma_{L-1}\left(z^L\right)$$

Parameters of 1 to N_T hidden layers of the Target Task transferred from the base task **are kept frozen during the backpropagation / optimisation** of parameters of the target task

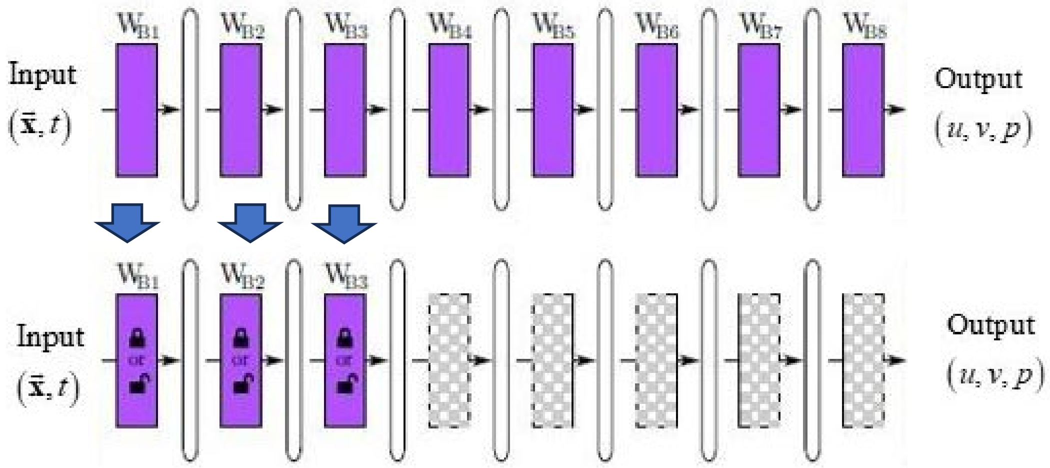
Parameters of N_{T+1} to L hidden layers for the Target Task are computed via backpropagation following gradient descent optimization while **freezing the parameters transfered from base task to target task**

TARGET TASK

$$z^1 = \bar{W}^1 X$$
$$z^2 = \bar{W}^2 \sigma_1\left(z^1\right)$$
$$z^3 = \bar{W}^3 \sigma_2\left(z^2\right)$$
$$\vdots$$
$$z^{N_T} = \bar{W}^{N_T} \sigma_{N_T-1}\left(z^{N_T-1}\right)$$

$$z^{N_T+1} = \bar{W}^{N_T+1} \sigma_{N_T}\left(z^{N_T}\right)$$
$$\vdots$$
$$z^{L-1} = \bar{W}^{L-1} \sigma_{L-2}\left(z^{L-2}\right)$$
$$z^L = \bar{W}^L \sigma_{L-1}\left(z^{L-1}\right)$$
$$Y^L = \sigma_{L-1}\left(z^L\right)$$

TRANSFER LEARNING IN PINNS



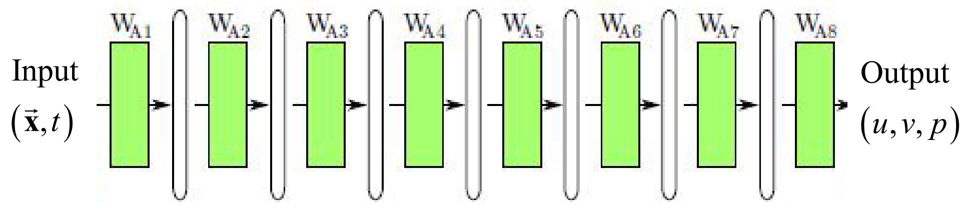
Base Task A: **FLOW PAST A CYLINDER**
Re=100 *Trained Using a PINN*

Target A: **FLOW PAST A CYLINDER**
Re=10 *Trained Using Transfer Learning from Base Task*

Parameters of 1 to N_T hidden layers of the **Target Task** transferred from the **Base task** are **kept frozen** during the backpropagation/optimisation of parameters of the **target task**

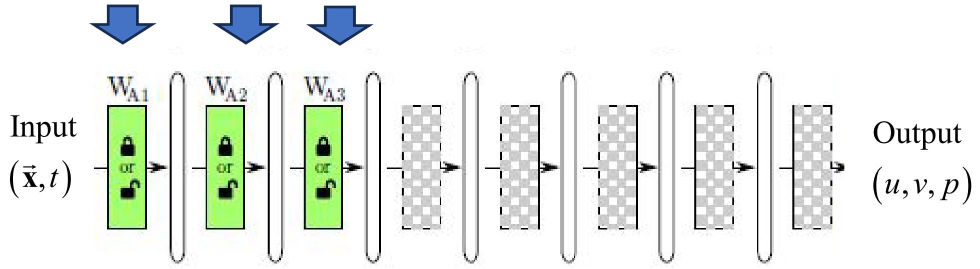
Parameters of N_{T+1} to L hidden layers for the **Target Task** are computed via backpropagation following gradient descent optimization while freezing the parameters transfered from base task to target task

TRANSFER LEARNING IN PINNS



Base Task A : Re=100

FLOW PAST AN AIRFOIL
Trained Using a PINN



Target Task B: Re=100

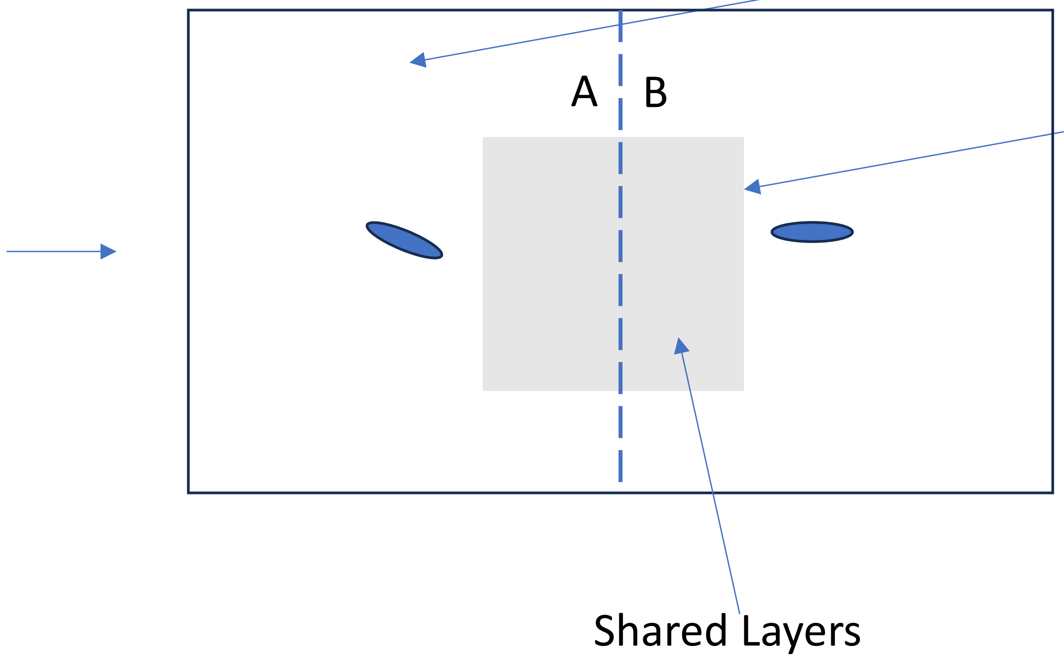
FLOW PAST A CYLINDER
Trained Using Transfer Learning from Base Task

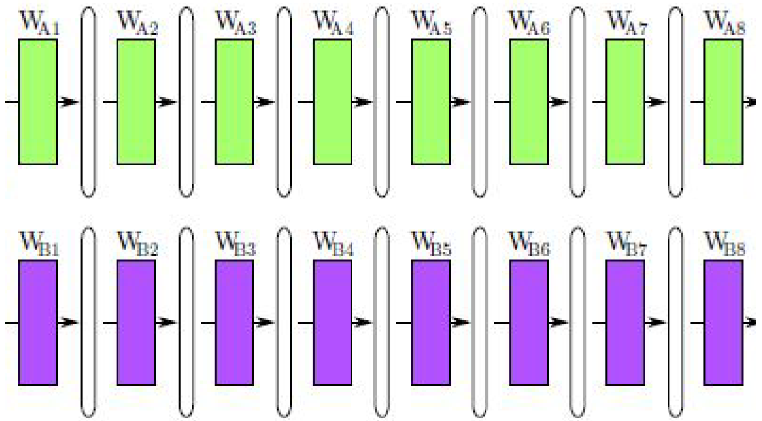
Parameters of 1 to N_T hidden layers of the **Target Task** i.e., (Flow Past a Cylinder) transferred from the **Base task** i.e., (Flow past an Airfoil) are kept frozen during the backpropagation/optimisation of parameters of the target task

Parameters of N_{T+1} to L hidden layers for the **Target Task** are computed via backpropagation following gradient descent optimization while freezing the parameters transfered from base task to target task. Alternative is to do fine-tuning transfer learning i.e., backpropagate to all layers

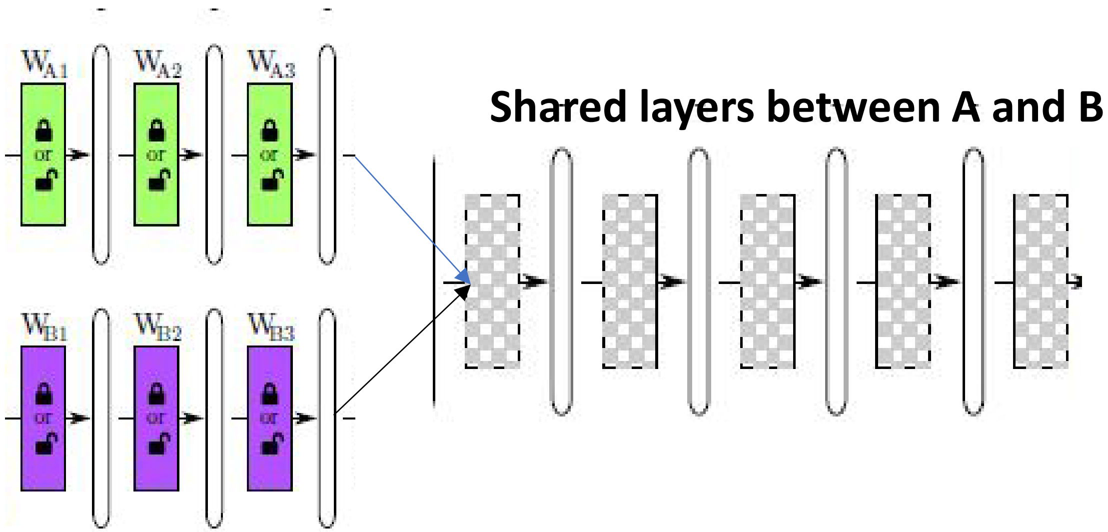
TRANSFER LEARNING IN PINNS

Considered **XPINNs/CPINNS/DPINNS** in Lecture PINN03

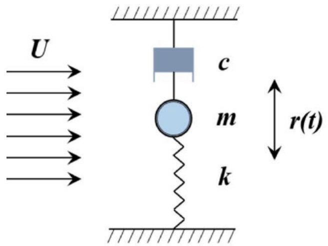


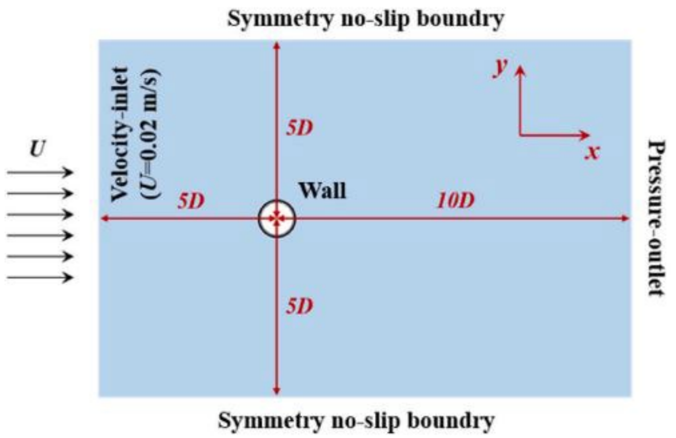


With interface boundary condition losses

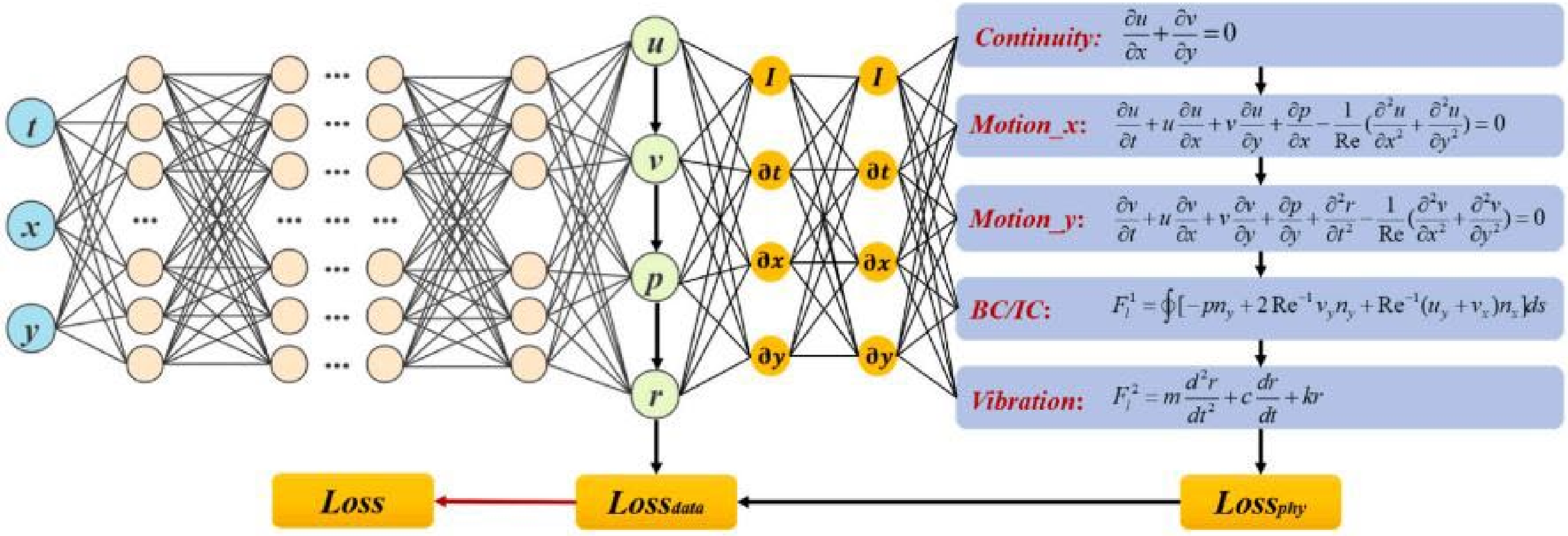


Shared layers between A and B



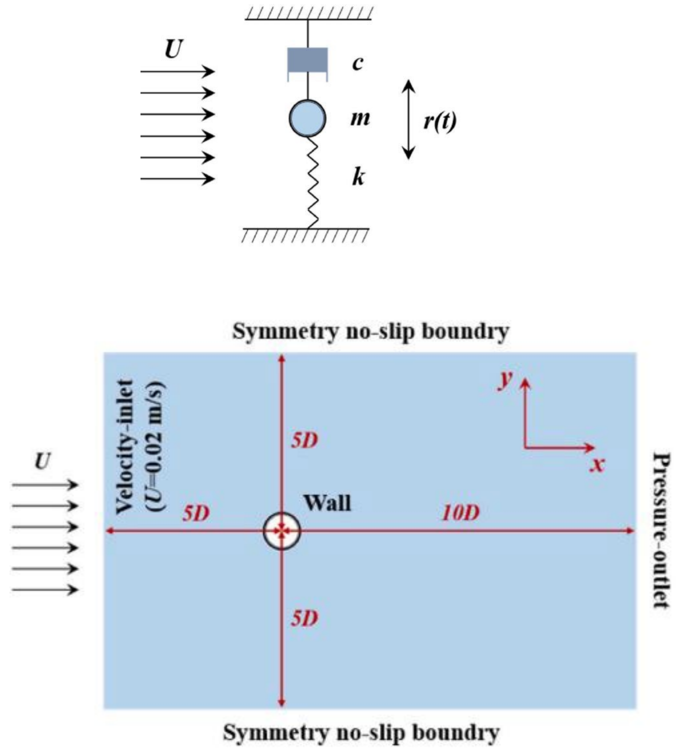


Flow Prediction of Vortex-Induced Vibration Using Conventional PINN Unsteady Incompressible Navier-Stokes Equations for Unsteady Flow past a Cylinder in Vertical Oscillation

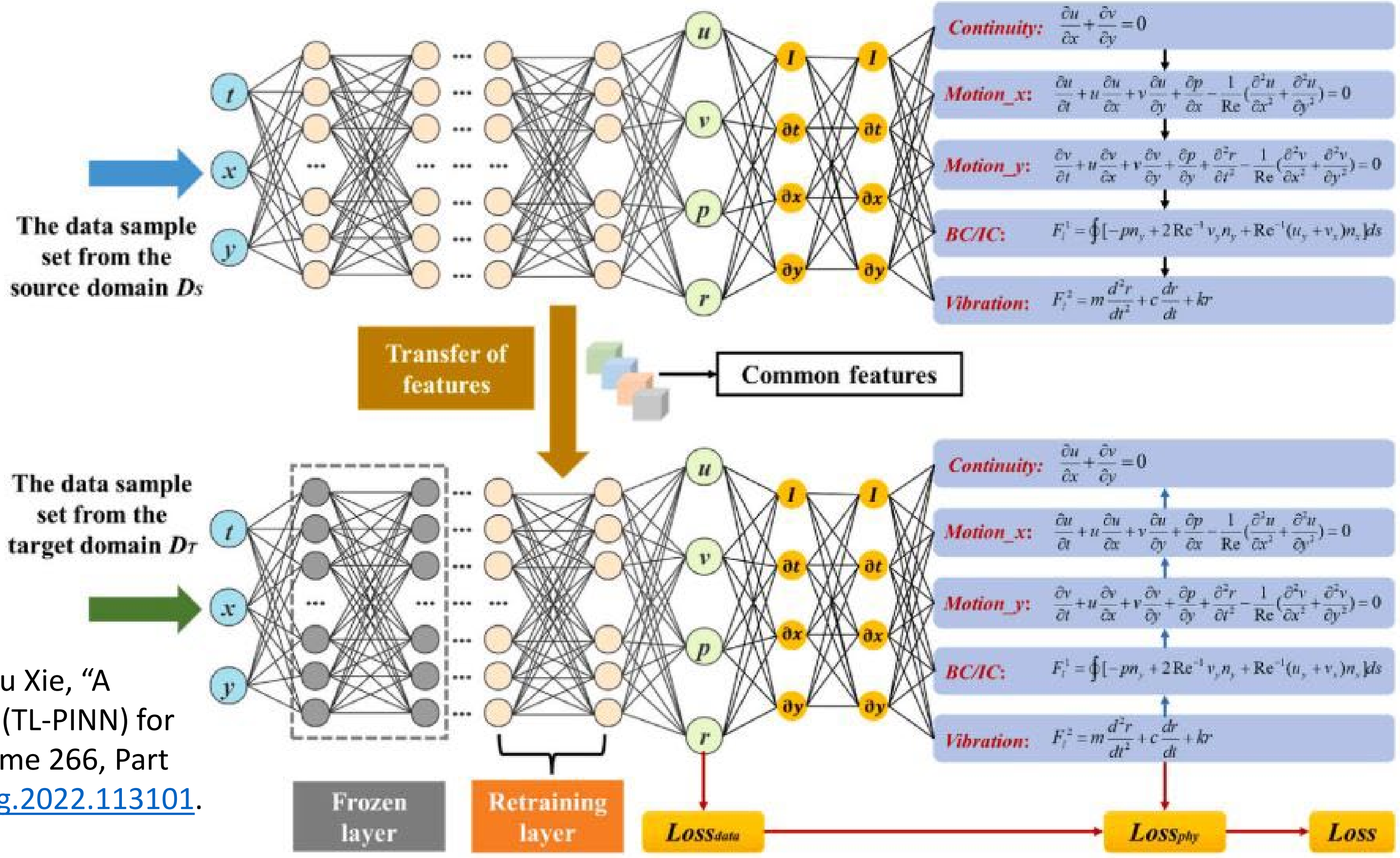


Source: Hesheng Tang, Yangyang Liao, Hu Yang, Liyu Xie, “A transfer learning-physics informed neural network (TL-PINN) for vortex-induced vibration”,*Ocean Engineering*, Volume 266, Part 4,2022,113101,https://doi.org/10.1016/j.oceaneng.2022.113101.

TRANSFER LEARNING IN PINNS FOR FLOW PREDICTION



Flow Prediction of Vortex-Induced Vibration Using Transfer Learning in a PINN



Source: Hesheng Tang, Yangyang Liao, Hu Yang, Liyu Xie, "A transfer learning-physics informed neural network (TL-PINN) for vortex-induced vibration",*Ocean Engineering*, Volume 266, Part 4,2022,113101,https://doi.org/10.1016/j.oceaneng.2022.113101.

TRANSFER LEARNING IN PINNS FOR FLOW PREDICTION

Note: Transfer Learning is used to reconstruct solutions on a sequence of samples halved sequentially from the PINN samples.

The settings of the 4 cases.

Case	The number of spatial points in the target domain	Training model
Case 1	14400 (1)	PINN
Case 2	7200 (1/2)	TL-PINN
Case 3	3600 (1/4)	
Case 4	1800 (1/8)	

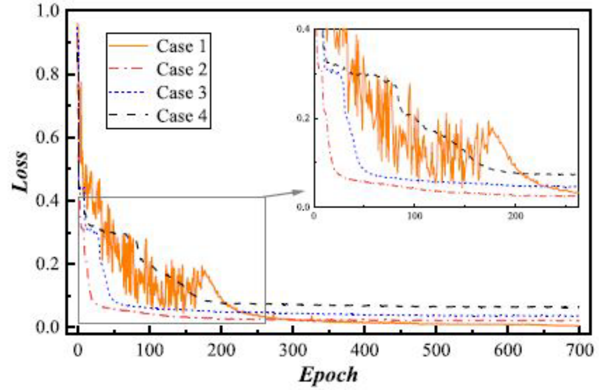


Fig. 17. The loss function of different cases during training.

Table 10
The training time of different cases.

Case	Epoch	Time (h)
Case 1	700	48.7
Case 2	700	22.9
Case 3	700	10.3
Case 4	700	5.4

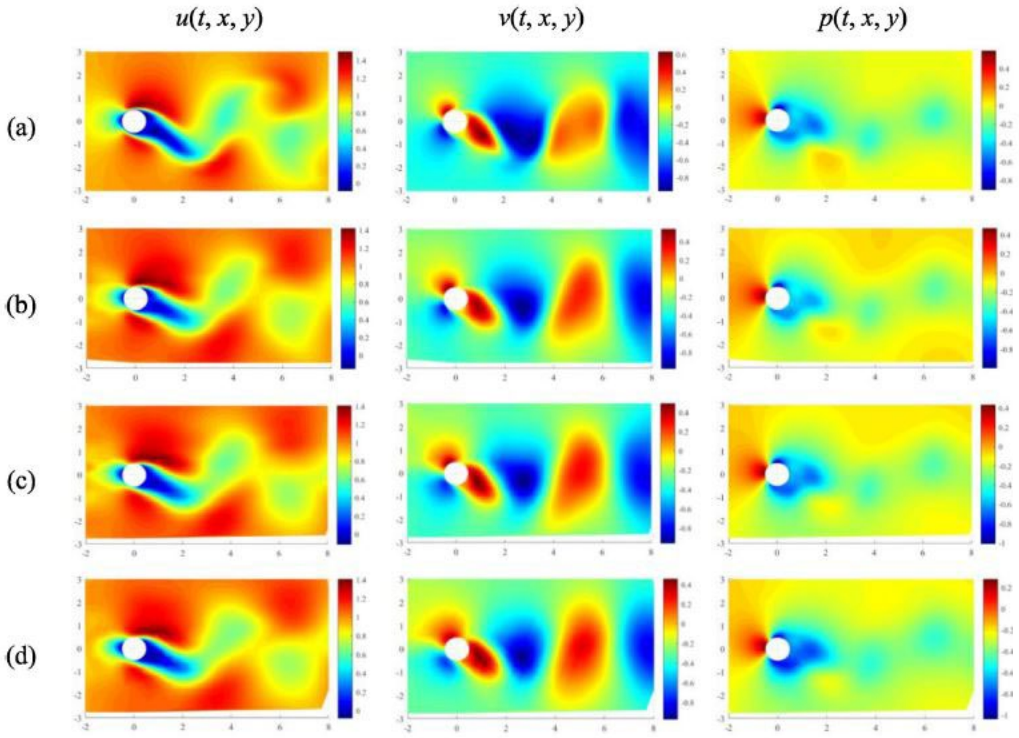


Fig. 18. The reproduced results of flow field information at 13 s by different cases. (a) Case 1; (b) Case 2; (3) Case 3; (4) Case 4.

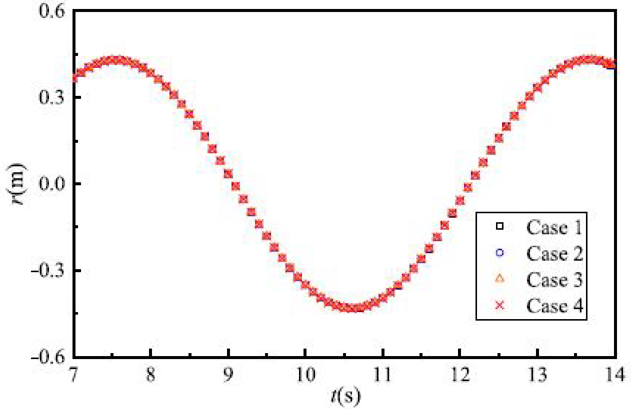


Fig. 19. The vibration displacement of the cylinder.

Transfer Learning generally improves the rate of convergence and reduces the computational effort and storage

Source: Hesheng Tang, Yangyang Liao, Hu Yang, Liyu Xie, “A transfer learning-physics informed neural network (TL-PINN) for vortex-induced vibration”,*Ocean Engineering*, Volume 266, Part 4,2022,113101,https://doi.org/10.1016/j.oceaneng.2022.113101.

RECENT REFERENCES ON APPLICATIONS OF TRANSFER LEARNING IN FLUID MECHANICS

Srihari M. and Balaji Srinivasan, Transfer physics informed neural network: a new framework for distributed physics informed neural networks via parameter sharing; July 2022, Engineering with Computers 39(1145/1390156) DOI:10.1007/s00366-022-01703-9

Zhao Zhang, Hao Yang and Xianfeng Yang(2023), " A Transfer Learning–Based LSTM for Traffic Flow Prediction with Missing Data," *ASCE Journal of Transportation Engineering, Part A: Systems*, Volume 149, Issue 10; https://doi.org/10.1061/JTEPBS.TEENG-7638

ReX

TRANSFER LEARNING IN PINNS FOR 3D INCOMPRESSIBLE NAVIER-STOKES USING THE L-HYDRA CONCEPT ?