# SVD-PINNs: Transfer Learning of Physics-Informed Neural Networks via Singular Value Decomposition

Yihang Gao

The University of Hong Kong

June 5, 2023

# Transfer Learning of PINNs

- For a given PDE

$$\begin{aligned} \mathcal{D}[\mathbf{u}, \mathbf{x}] &= \mathbf{f}(\mathbf{x}), \quad \mathbf{x} \in \Gamma \subset \mathbb{R}^d, \\ \mathcal{B}[\mathbf{u}, \mathbf{x}] &= \mathbf{g}(\mathbf{x}), \quad \mathbf{x} \in \partial\Gamma, \end{aligned} \tag{1}$$

  where $\mathcal{D}$ and $\mathcal{B}$ are differential operators in the interior and on the boundary respectively, $\Gamma$ is an open set of our interest and $\partial\Gamma$ is its boundary.
- We adopt a neural network $\phi(\mathbf{x}; \theta)$ as a surrogate to the solution $\mathbf{u}(\mathbf{x})$ [Raissi et al., 2019]. Here, $\phi(\mathbf{x}; \theta)$ is a fully connected 2-hidden-layers neural network parameterized by $\theta$:

$$\phi(\mathbf{x}; \theta) = \mathbf{W}_2 \cdot (\sigma(\mathbf{W}_1 \cdot \sigma(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1)) + \mathbf{b}_2, \tag{2}$$

  and

$$\theta = \{\mathbf{W}_2, \mathbf{W}_1, \mathbf{W}_0, \mathbf{b}_2, \mathbf{b}_1, \mathbf{b}_0\},$$

  with activation function $\sigma(x) = \text{ReLU}^q(x)$ or $\sigma(x) = \tanh(x)$.

# Transfer Learning of PINNs

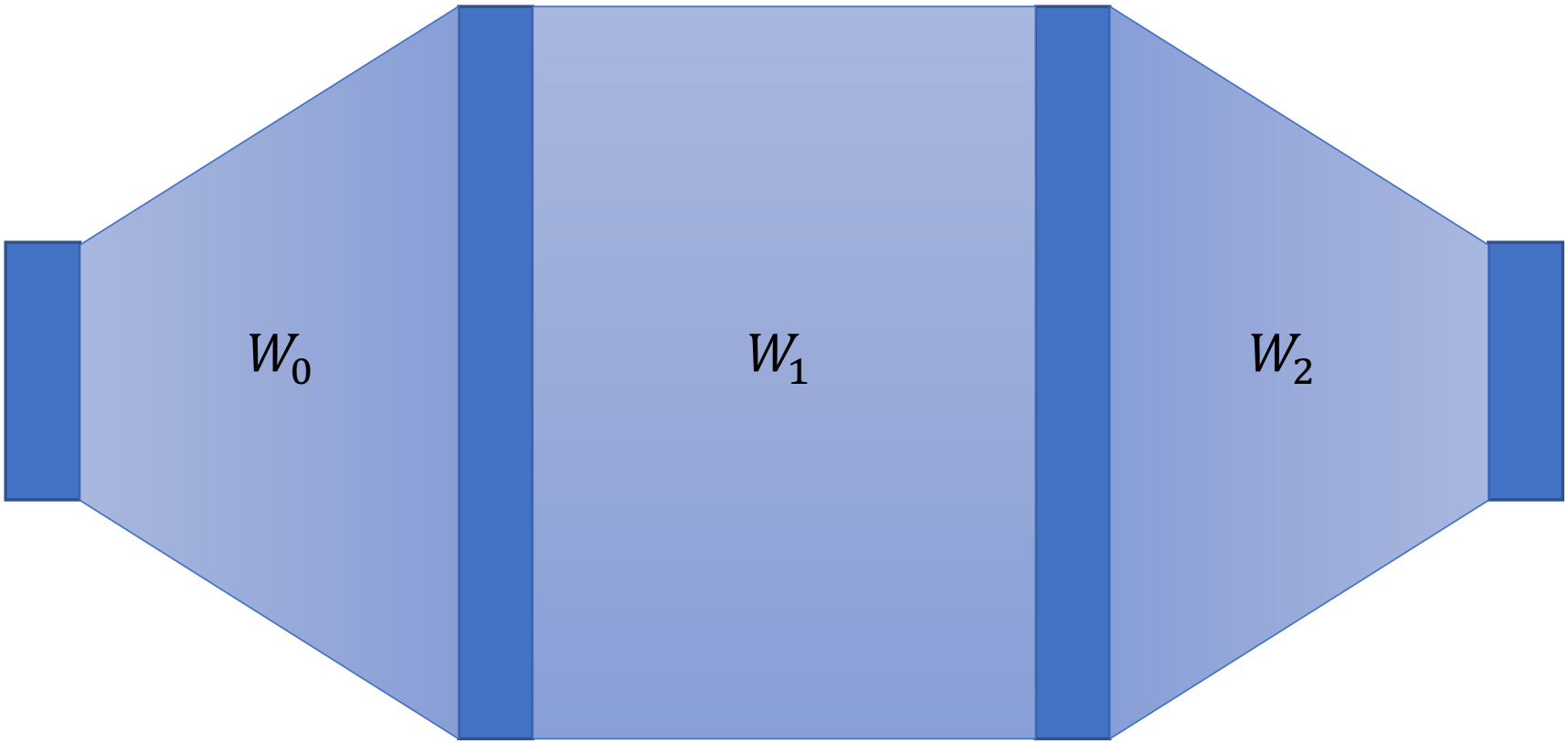


Figure: Architecture of neural networks

# Transfer Learning of PINNs

- With interior training samples $\{\mathbf{x}_i\}_{i=1}^{n_1}$ and initial/boundary samples $\{\tilde{\mathbf{x}}_j\}_{j=1}^{n_2}$, the loss function of PINNs is formulated as

$$\begin{aligned}\min_{\theta} \quad & \nu \cdot \frac{1}{n_1} \sum_{i=1}^{n_1} \frac{1}{2} \left\| \mathcal{D}[\phi(\mathbf{x}_i;\theta), \mathbf{x}_i] - \mathbf{f}(\mathbf{x}_i) \right\|_2^2 \\ & + \frac{1}{n_2} \sum_{j=1}^{n_2} \frac{1}{2} \left\| \mathcal{B}[\phi(\tilde{\mathbf{x}}_j;\theta), \tilde{\mathbf{x}}_j] - \mathbf{g}(\tilde{\mathbf{x}}_j) \right\|_2^2 ,\end{aligned} \tag{3}$$

  where the hyperparameter $\nu > 0$ balances the interior and initial/boundary conditions.

# Transfer Learning of PINNs

- The method above solves a single PDE, which implies that one neural network corresponds to one PDE, even if several PDEs have similarities and shared information.

- DeepONet [Lu et al., 2021], FNO [Li et al., 2020], Trasnformer [Cao, 2021]: data-driven methods, learning differential operator, high requirements for (solution) data.

# Transfer Learning of PINNs

- The deep Galerkin's idea: $\boldsymbol{u} = \sum_{i=1}^{m} a_i \varphi_i(\mathbf{x})$, where $\varphi_i(\mathbf{x})$ is some base function.

- For a class of PDEs with the same differential operators $\mathcal{D}$ and $\mathcal{B}$ but different right-hand side functions $\{\mathbf{f}_\epsilon(\mathbf{x})\}_\epsilon$ and $\{\mathbf{g}_\epsilon(\mathbf{x})\}_\epsilon$, the corresponding approximate solution $\phi(\mathbf{x}; \theta_\epsilon)$ shares some parameters $\{\mathbf{W}_1, \mathbf{W}_0, \mathbf{b}_2, \mathbf{b}_1, \mathbf{b}_0\}$ but only $\mathbf{W}_2$ is trainable.

$$\phi(\mathbf{x}; \theta) = \mathbf{W}_2 \cdot (\sigma(\mathbf{W}_1 \cdot \sigma(\mathbf{W}_0 \mathbf{x} + \mathbf{b}_0) + \mathbf{b}_1)) + \mathbf{b}_2.$$

# Motavition

- Improve the capability of the base function.
- Do not change the base functions much.
- Intuitively, for PDEs with different but close right-hand side functions, the corresponding approximate solution $\phi(\mathbf{x}; \theta_\epsilon)$ may share the bases of $\mathbf{W}_1$, i.e., $\mathbf{U}$ and $\mathbf{V}$ are frozen with $\mathbf{W}_1 = \mathbf{U} * \mathbf{D} * \mathbf{V}^{\mathrm{T}}$ for different $\epsilon$.

# PINNs

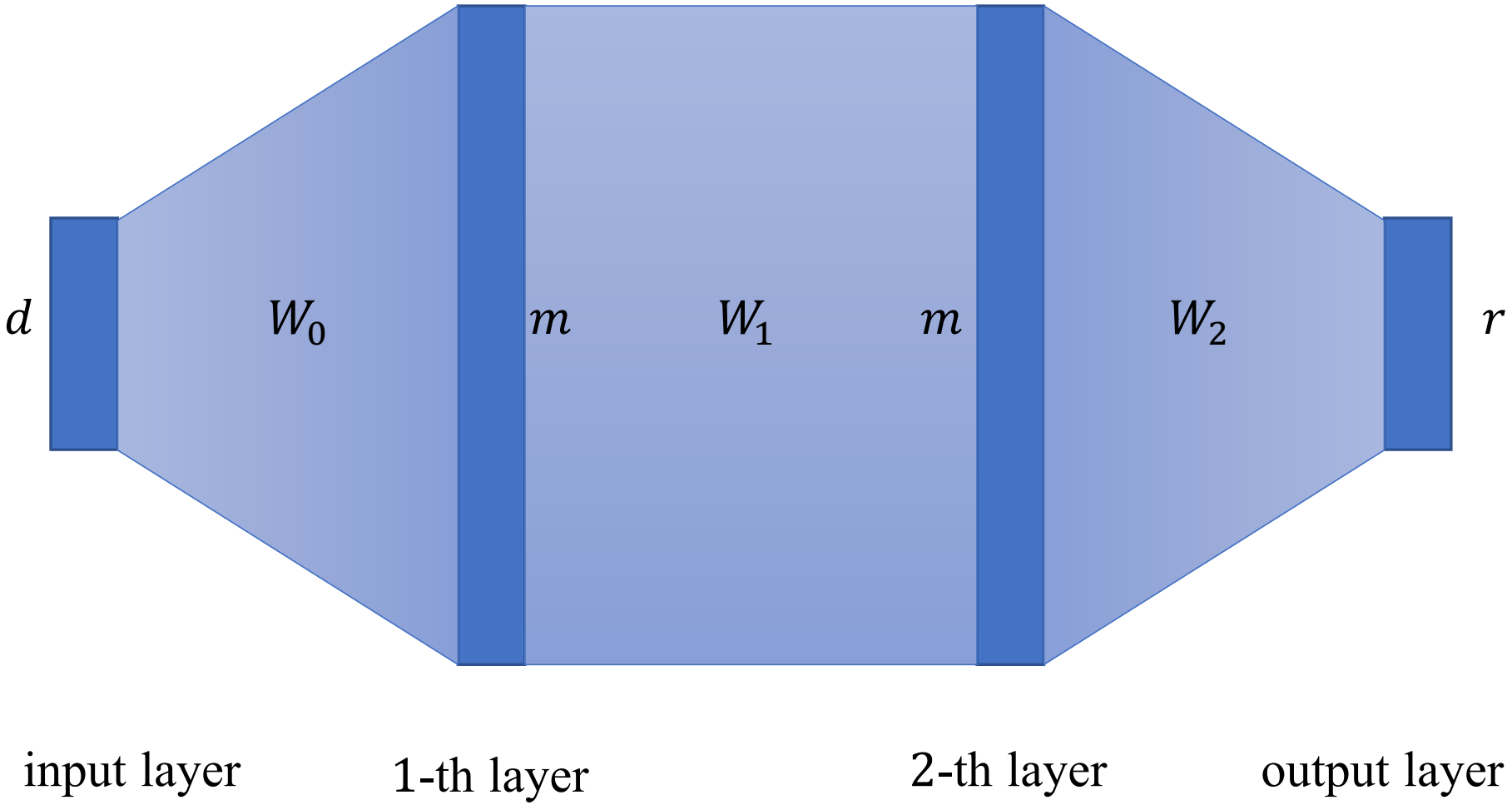


Figure: Architecture of neural networks

**Algorithm** SVD-PINNs

**Input: samples** $\{\mathbf{x}_i\}_{i=1}^{n_1}$, $\{\tilde{\mathbf{x}}_j\}_{j=1}^{n_2}$ $\{\mathbf{f}_\epsilon(\mathbf{x})\}_\epsilon$ **and** $\{\mathbf{g}_\epsilon(\mathbf{x})\}_\epsilon$ in
**Output:** $\{\theta_\epsilon\}_\epsilon$ out

*Pretraining* :

1: Using gradient-based methods to optimize the loss (3) with $\mathbf{f}_0$ and $\mathbf{g}_0$, we have the output variables $\theta_0 = \{\mathbf{W}_2, \mathbf{W}_1, \mathbf{W}_0, \mathbf{b}_2, \mathbf{b}_1, \mathbf{b}_0\}$.
2: Applying the SVD to $\mathbf{W}_1 = \mathbf{U} * \mathbf{D} * \mathbf{V}^{\mathrm{T}}$ and $\{\mathbf{U}, \mathbf{V}\}$ are frozen for all $\theta_\epsilon$. $\{\mathbf{D}, \mathbf{W}_2, \mathbf{W}_0, \mathbf{b}_2, \mathbf{b}_1, \mathbf{b}_0\}$ are trainable variables.
3: Transfer Learning of PINNs for a given $\epsilon$:
4: $\theta_\epsilon(0) = \theta_0$.
5: **for** $t = 1$ to $T$ **do**
6: Updating variables $\{\mathbf{W}_2, \mathbf{W}_0, \mathbf{b}_2, \mathbf{b}_1, \mathbf{b}_0\}$.
7: Updating singular values $\mathbf{D}$.
8: We have the current variables $\theta_\epsilon(t)$.
9: **end for**
10: $\theta_\epsilon = \theta_\epsilon(T)$
11: **return** $\{\theta_\epsilon\}_\epsilon$

# Advantages

- If we aim to solve $n$ PDEs with the same differential operators but different right-hand side functions $\mathbf{f}_\epsilon$ and $\mathbf{g}_\epsilon$:
  - PINNs method without transfer learning is required to optimize and store $n$ different neural networks, where the total storage is of $n \cdot \left(m^2 + (r + d + 1) \cdot m + r\right)$ parameters.
  - For the proposed SVD-PINNs, $n \cdot ((r + d + 2) \cdot m + r) + 2m^2$ parameters are stored.

# Potential Limitations

- Nonconvex for parameters $\theta$.
- Singular values are non-negative: constrained optimization problem. Projection? E.g., Wasserstein GAN.

# Allen-Cahn Equations

We consider the following 10-dimensional nonlinear parabolic (Allen-Cahn) equation:

$$\begin{aligned} &\frac{\partial u}{\partial t}(t,\mathbf{x}) - \Delta_{\mathbf{x}} u(t,\mathbf{x}) - u(t,\mathbf{x}) + u^3(t,\mathbf{x}) = f_\epsilon(t,\mathbf{x}), \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{in } (0,1)^{10} \cup \Omega, \\ &u(t,\mathbf{x}) = g_\epsilon(t,\mathbf{x}), \text{ on } (0,1)^{10} \cup \partial\Omega \\ &u(0,\mathbf{x}) = h_\epsilon(\mathbf{x}), \text{ in } \Omega, \end{aligned} \tag{4}$$

with $\Omega = \{\mathbf{x} : \|\mathbf{x}\|_2 < 1\}$. Here, right-hand sides of the PDE (i.e., $f_\epsilon$, $g_\epsilon$ and $h_\epsilon$) are set by the exact solution

$$\begin{aligned} u_\epsilon(t,\mathbf{x}) = \exp(-t) \cdot \Big(&\sin\left(\frac{\pi}{2}(1 - \|\mathbf{x}\|_2)^{2.5}\right) \\ &+ \epsilon \cdot \sin\left(\frac{\pi}{2}(1 - \|\mathbf{x}\|_2)\right)\Big). \end{aligned} \tag{5}$$

# Allen-Cahn Equations

We first pre-train the model on the Allen-Cahn equation with $\epsilon = 0$, and then apply some transfer learning strategies to solve PDEs with $\epsilon = 50$.

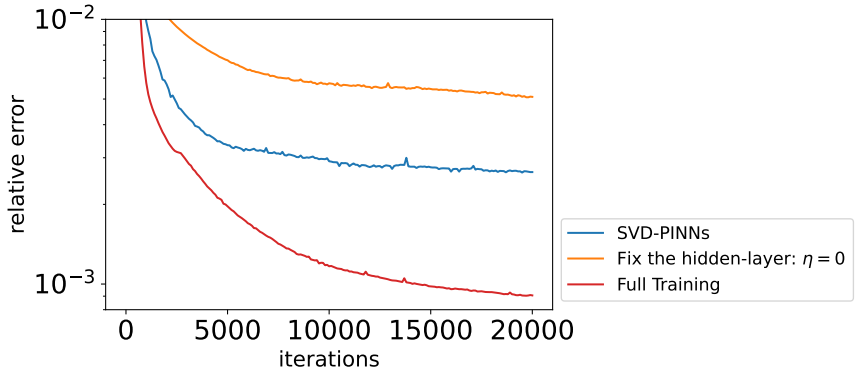


Figure: Trajectories of the relative error for the SVD-PINNs, the simple transfer learning method, and full training method in solving the 10-dimensional Allen-Cahn equation.

# Conclusion

- SVD-PINNs are more stable and powerful in numerical performances than some simple transfer learning methods.

# References


- Gao, Y., Cheung, K.C. and Ng, M.K., 2022. SVD-PINNs: Transfer learning of physics-informed neural networks via singular value decomposition. IEEE Symposium on Series on Computational Intelligence (SSCI).

# References I



Raissi, Maziar and Perdikaris, Paris and Karniadakis, George 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*.


Lu, Lu and Jin, Pengzhan and Pang, Guofei and Zhang, Zhongqiang and Karniadakis, George Em (2021)
Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators,
*Nature Machine Intelligence*.


Li, Zongyi and Kovachki, Nikola Borislavov and Azizzadenesheli, Kamyar and Bhattacharya, Kaushik and Stuart, Andrew and Anandkumar, Anima (2020)
Fourier Neural Operator for Parametric Partial Differential Equations,
*ICLR*.


Cao, Shuhao (2021)
Choose a transformer: Fourier or galerkin,
*NeurIPS*.

# The End

Thanks!