AI for Scientific Computing: Theory and Applications

2024.11.14

International Workshop on AI for Theoretical Sciences

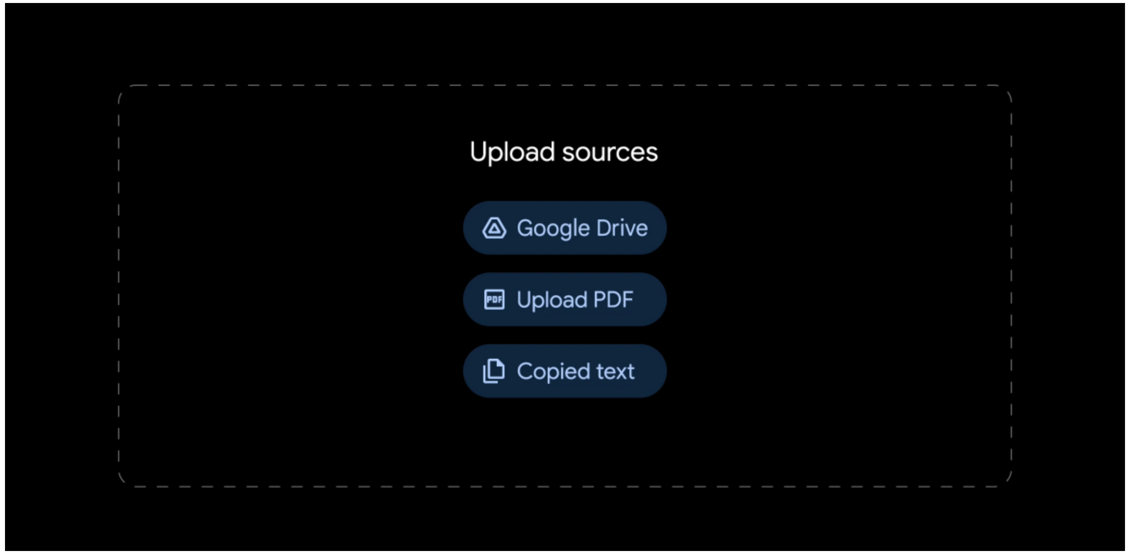
Youngjoon Hong
KAIST

Great success in AI and ML

Open AI: Sora

Prompt: A movie trailer featuring the adventures of the 30 year old space man wearing a red wool knitted motorcycle helmet, blue sky, salt desert, cinematic style, shot on 35mm film, vivid colors.

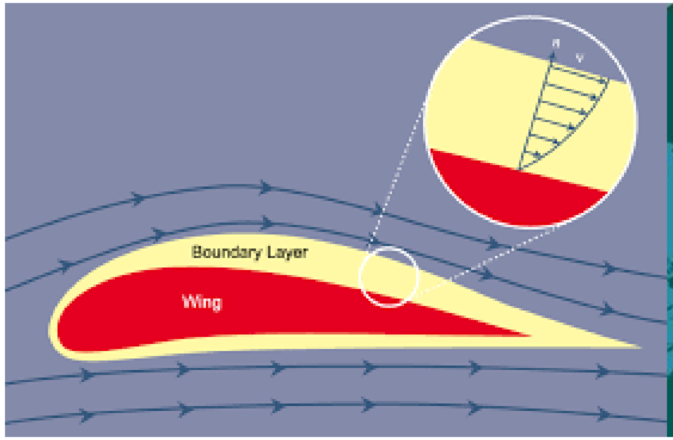
Google: Notebook LM

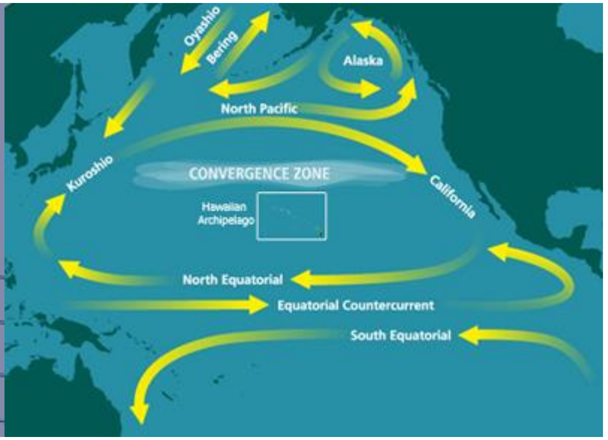


Although I am talking about the AI,
I am a mathematician:

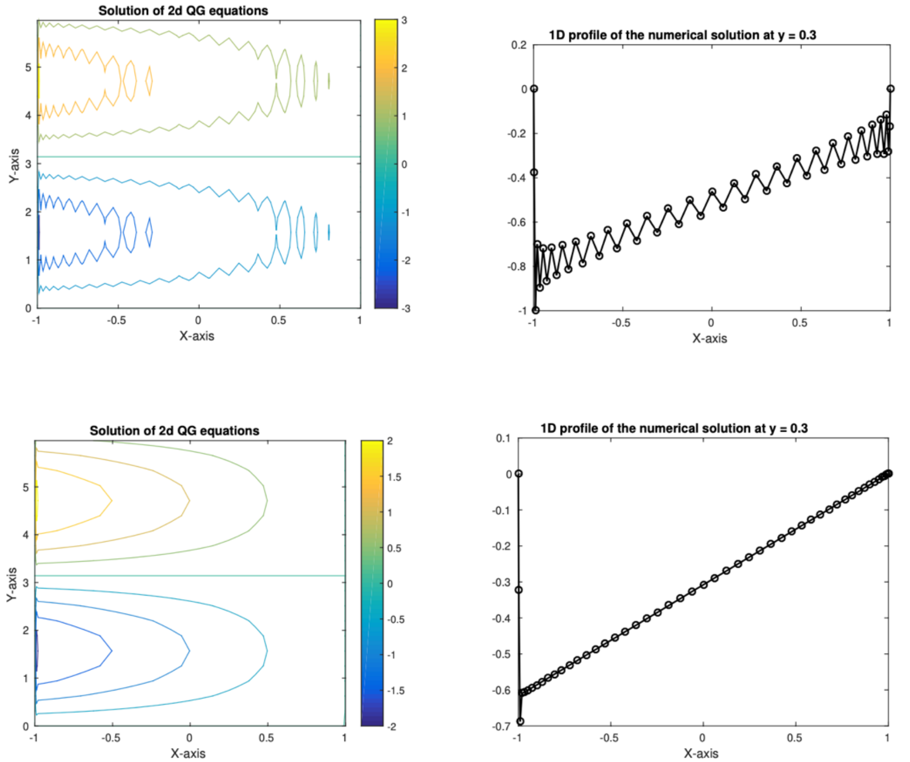
Singular Perturbation and Boundary layer

A major problem in mathematical and engineering fluid mechanics is the study of the boundary layer for the Navier-Stokes equations at small viscosity.





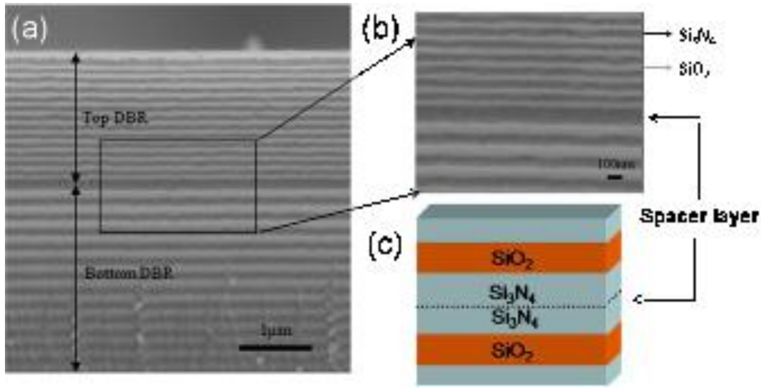
In general, due to the sharp transition inside boundary layers, a careful numerical treatment is required.

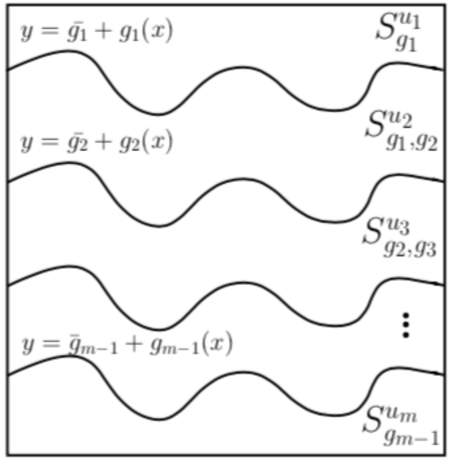


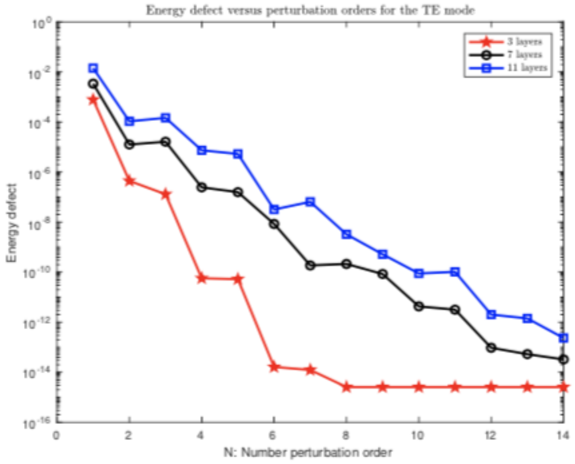
$$V_N := \left\{ u_N = \sum_{i=1}^{N} c_i \varphi_i(x,y) \right\} \subset H_0^1(D),$$

$$\bar{V}_N := \left\{ \bar{u}_N = \sum_{i=1}^{N} c_i \varphi_i(x,y) + \sum_{j=1}^{M} d_j \varphi_0(r,\eta_j)\psi_j(\eta) \right\} \subset H_0^1(D).$$

Electromagnetic Waves (layered structures)







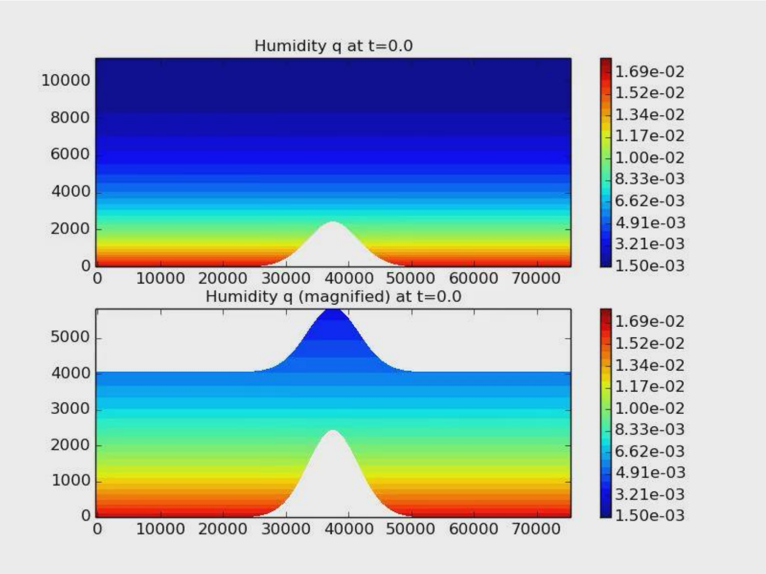
(a) A depiction of a multiply layered grating structure.

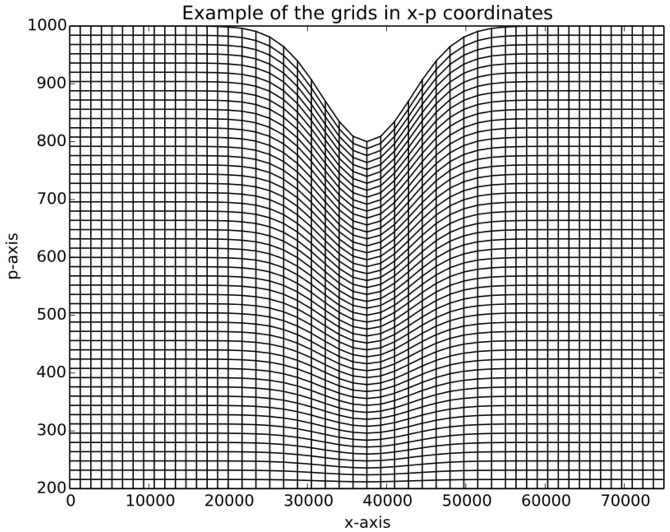
(b) Scattering solution with different numbers of layers.

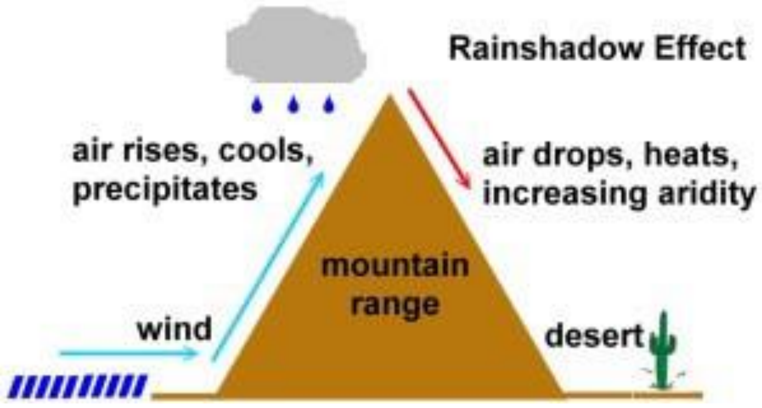
Numerical simulations are very useful since building device is complicated and expensive.

Geophysical Fluid Dynamics

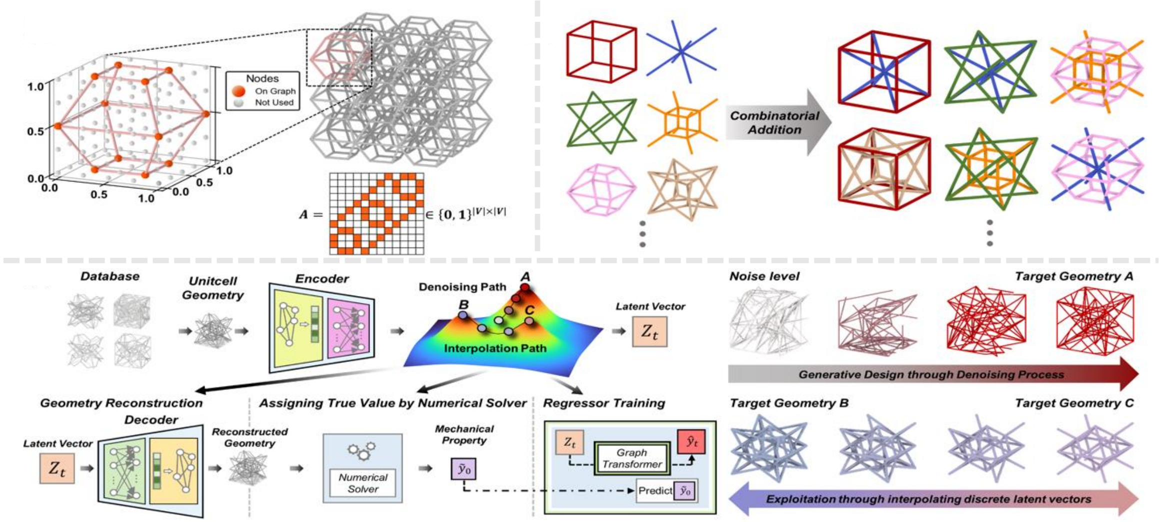
$$
\begin{cases}
\dfrac{\partial T}{\partial t} + u\dfrac{\partial T}{\partial x} + \omega\dfrac{\partial T}{\partial p} = \dfrac{\omega}{p}\left(\dfrac{RT}{C_p} - \delta\dfrac{\mathcal{L}F}{C_p}\right), \\
\dfrac{\partial q}{\partial t} + u\dfrac{\partial q}{\partial x} + \omega\dfrac{\partial q}{\partial p} = \delta\dfrac{F}{p}\omega, \\
\dfrac{\partial u}{\partial t} + u\dfrac{\partial u}{\partial x} + \omega\dfrac{\partial u}{\partial p} + \phi_x = 0, \\
\dfrac{\partial \omega}{\partial p} + \dfrac{\partial u}{\partial x} = 0, \\
\dfrac{\partial \phi}{\partial p} = -\dfrac{RT}{p}, \\
\phi = zg, \quad z = z(x,p,t).
\end{cases}
$$







My recent interests:



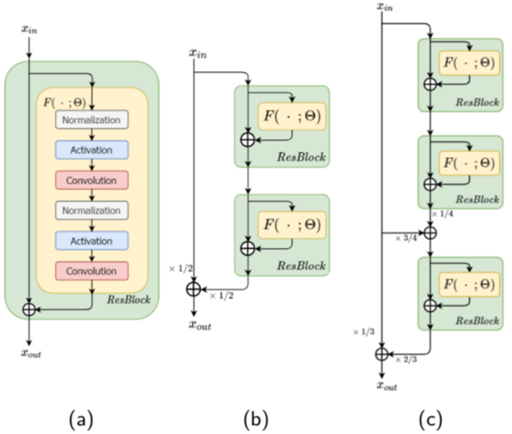


Figure: Network modules with *ResBlock* and SSP blocks. (a): *ResBlock*. (b): *SSP2-block*. (c): *SSP3-block*.

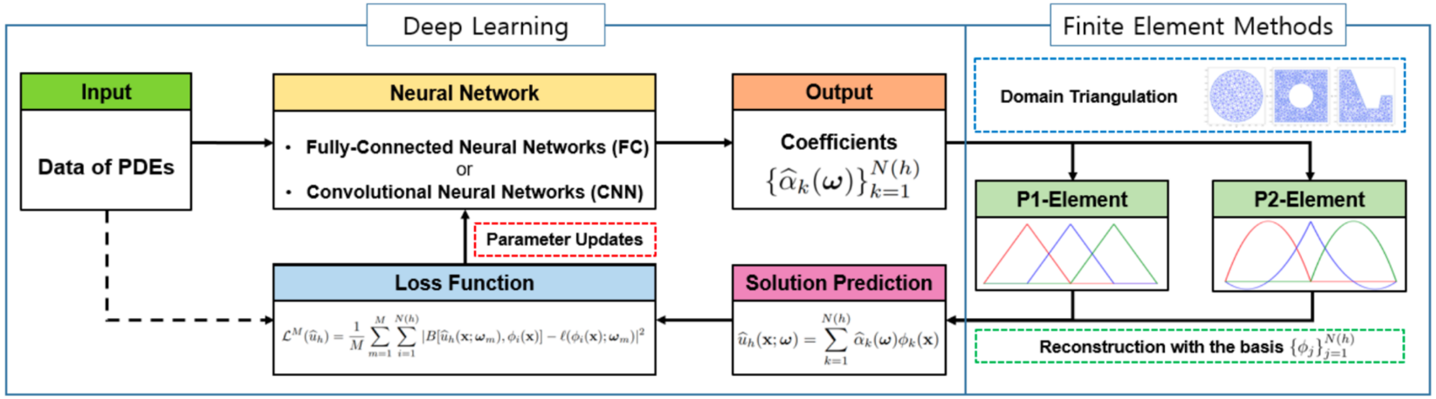
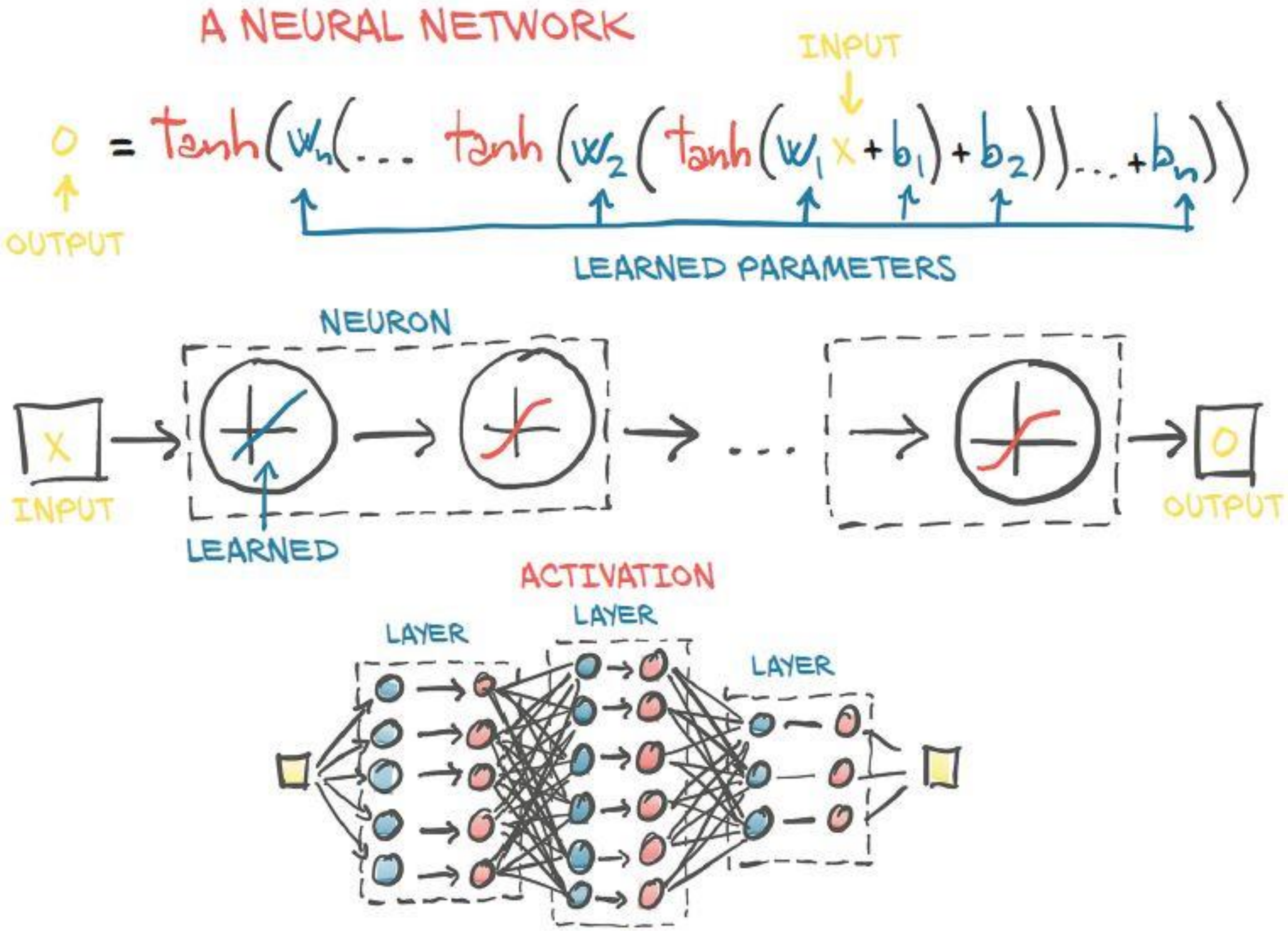


Figure: Schematic diagram of FEONet

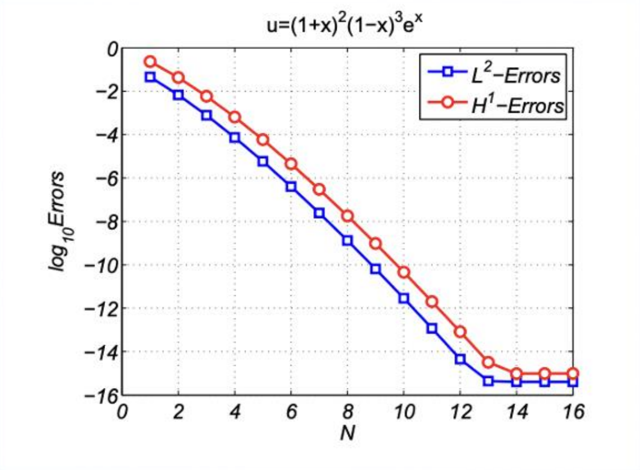
Neural Networks?



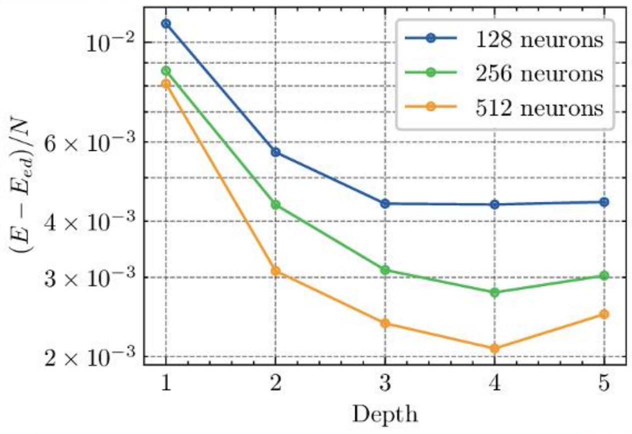
Cartoon by Girijesh

Theoretical Aspects

- Over the past decade, **Machine Learning** (ML) and **Neural Networks** (NN) have achieved remarkable breakthroughs. Despite their outstanding performance, many aspects of ML remain a "**black box**" to us.
- There is a significant **lack of understanding** regarding when and why ML systems work well, and how to improve them or address their failures.



numerical methods



ML methods

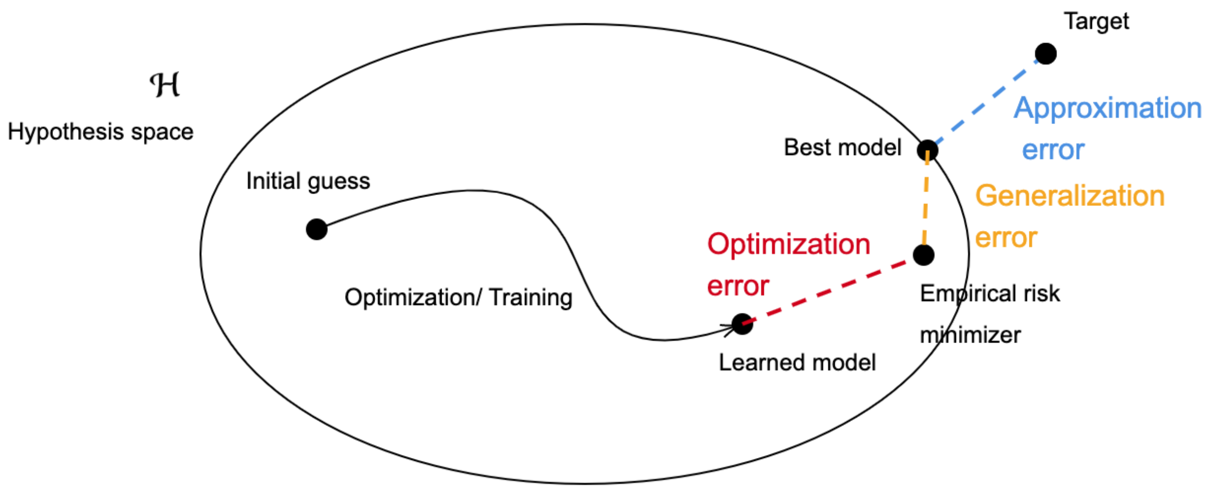
Theoretical Aspects

The main goal is to show the convergence of the approximation:

$$\|u - u_{n,M,N}\| \to 0 \text{ as } n, M, N \to 0$$

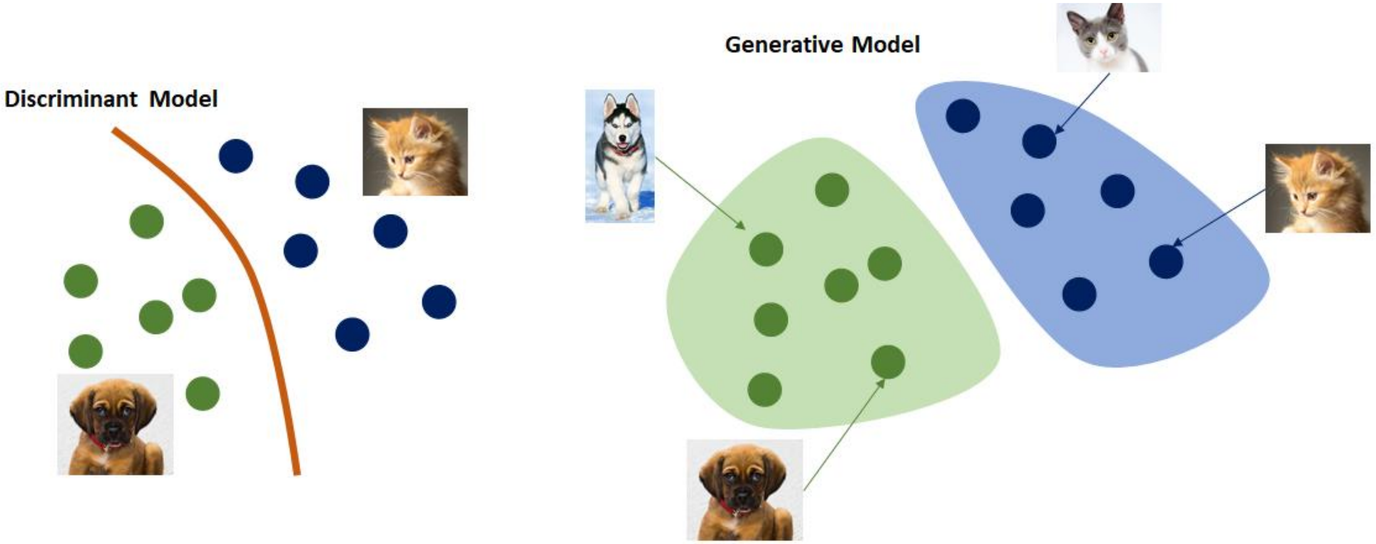
This is achieved by decomposing the error:

$$u - u_{n,M,N} = \underbrace{(u - u_n)}_{\textit{approx. error}} + \underbrace{(u_n - u_{n,M})}_{\textit{gen. error}} + \underbrace{(u_{n,M} - u_{n,M,N})}_{\textit{opt. error}}$$



Deep Generative Model

A generative model describes how a dataset is generated, in terms of a probabilistic model. By sampling from this model, we are able to generate new data.



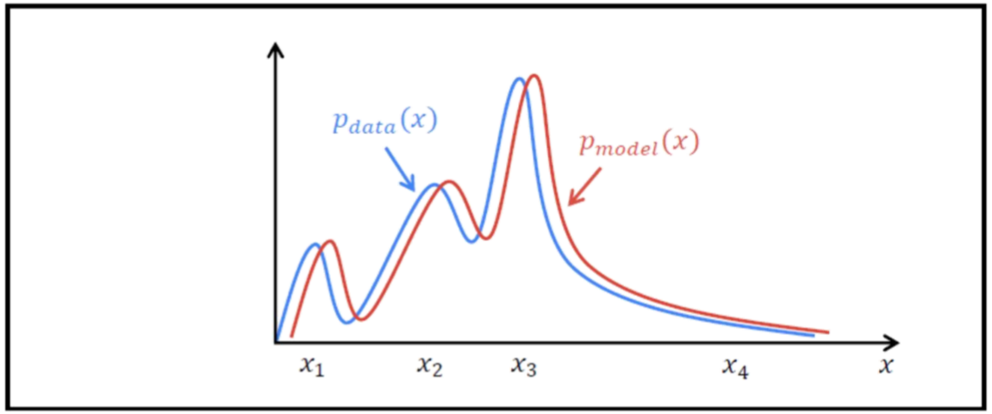
Example of the Progression in the Capabilities of deep generative model from 2014 to 2018.

Deep Generative Model

Distribution of images generated by the model

The goal of the generative model is to find a $p_{model}(x)$ that approximates $p_{data}(x)$ well.

Distribution of actual images



Then, generate new samples from $p_{model}(x)$

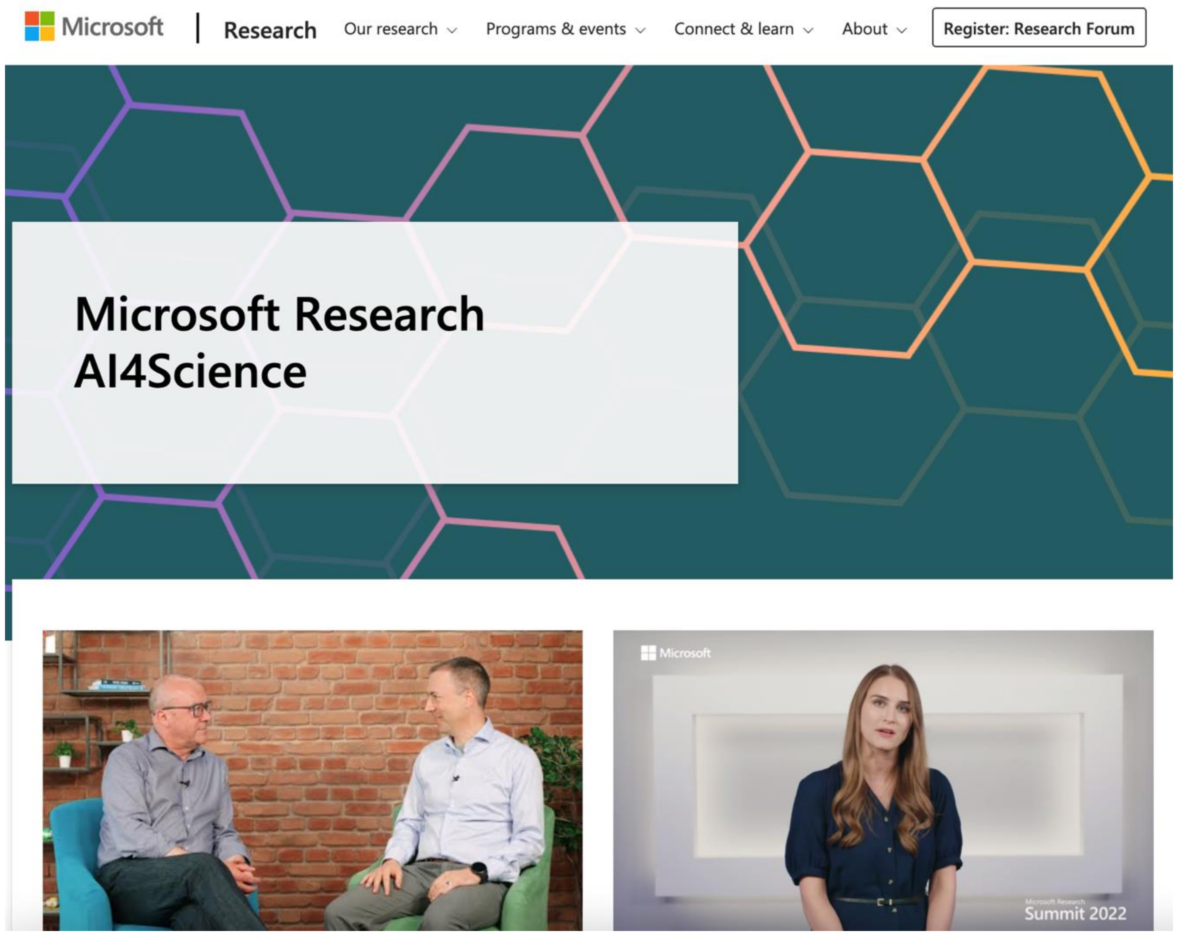
Addresses density estimation, a core problem in unsupervised learning

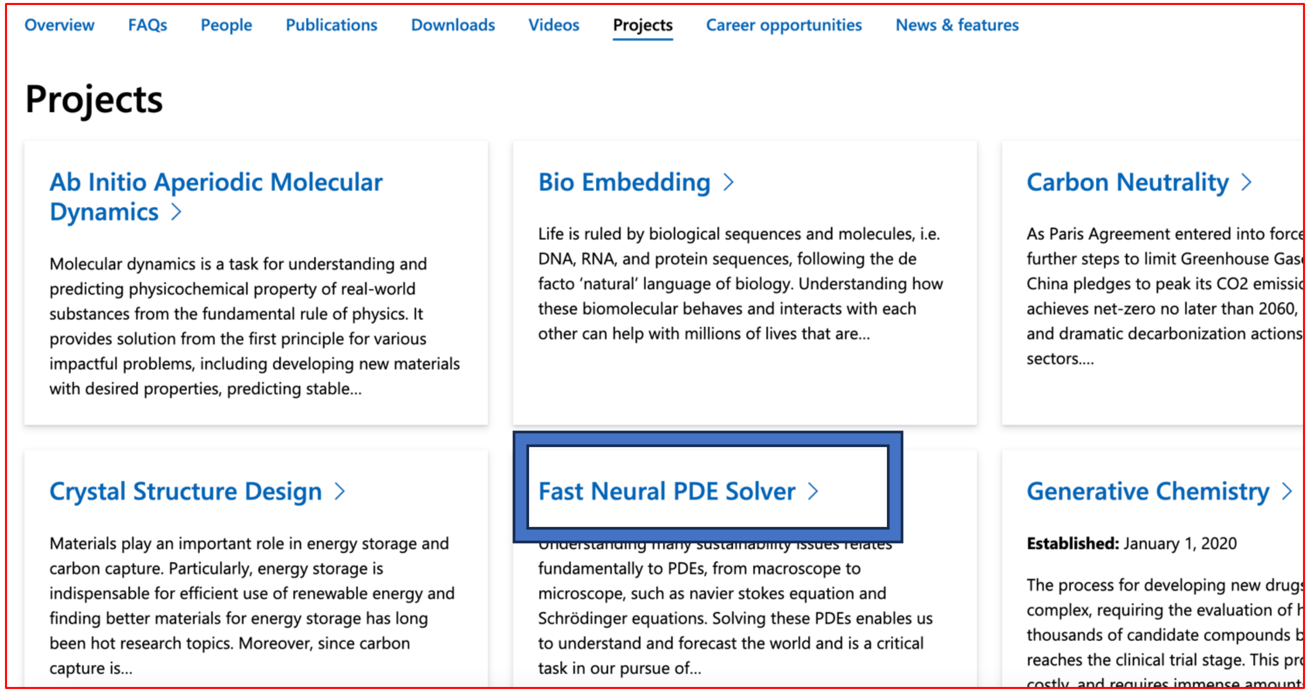
Several flavors:

- Explicit density estimation: explicitly define and solve for $p_{model}(x)$
- Implicit density estimation: learn model that can sample from $p_{model}(x)$ w/o explicitly defining it

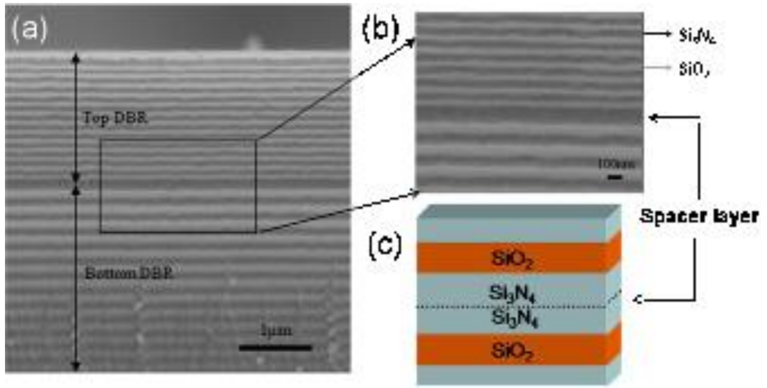
II. AI for Science

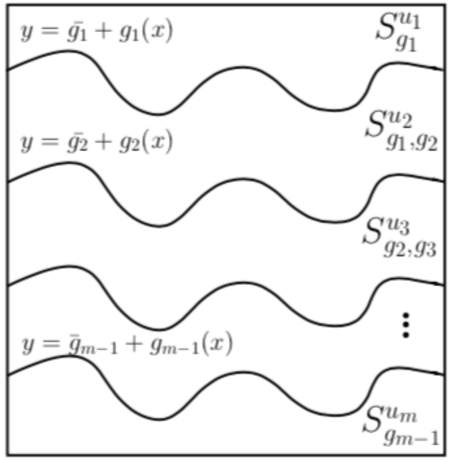
AI for Science



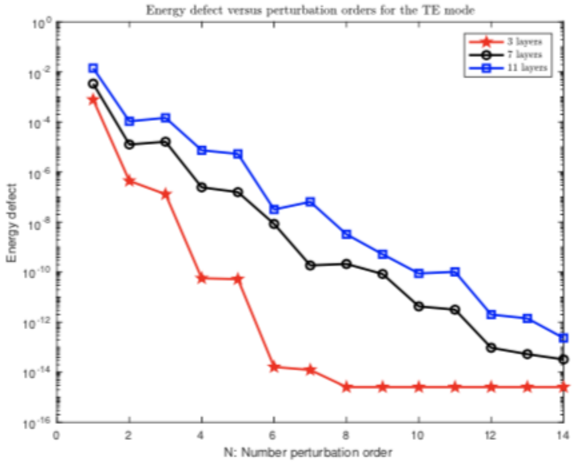


Revisits: my old friends





(a) A depiction of a multiply layered grating structure.



(b) Scattering solution with different numbers of layers.

Numerical simulations are very useful since building device is complicated and expensive.

Electromagnetic Waves (layered structures)

1. Flatten the interface using change of variables

⟹ ugly terms appear

2. Keep the Helmholtz operator in l.h.s., and source terms (nonlinear terms with g(x) and h(x)) in r.h.s.

3. Introduce ansats w.r.t. ε and rearrange the equations

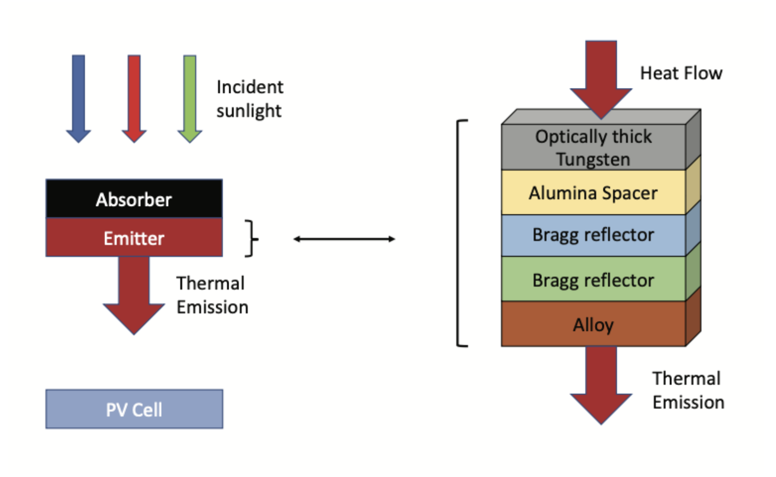
$$u = \sum_{n=-\infty}^{\infty} u_n(x,y)\varepsilon^n$$

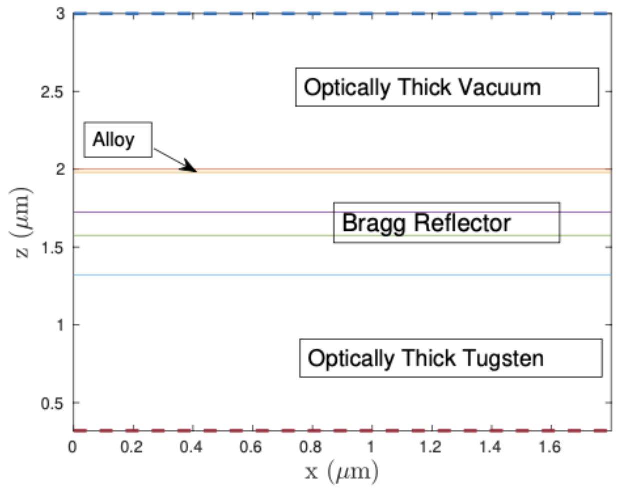
⟹ At each ε-order, inhomogeneous Helmholtz equations are deduced

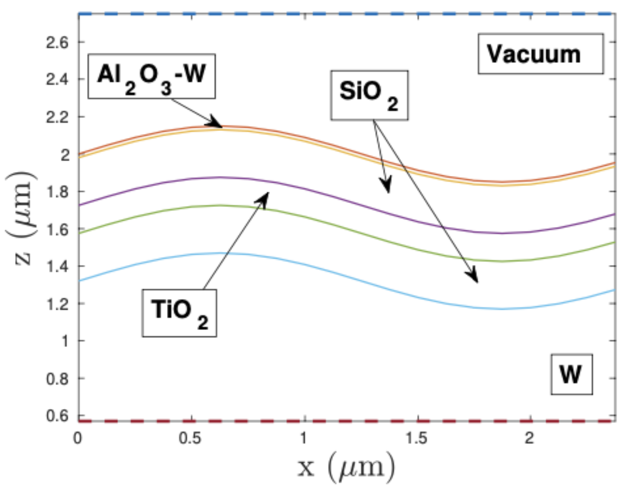
4. Solve the problem recursively

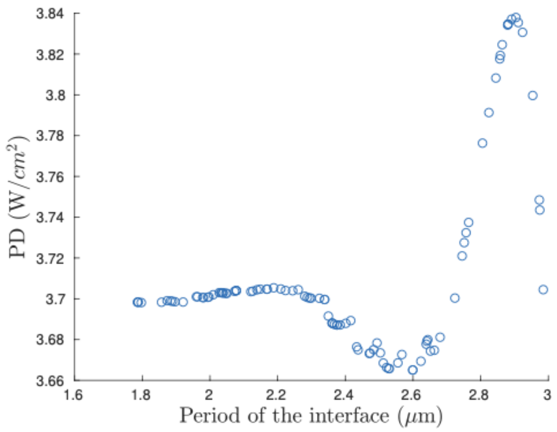
Hong et al., JCP (2017a, 2017b, 2018), APL (2019), SINUM (2021), Optics Express (2022)

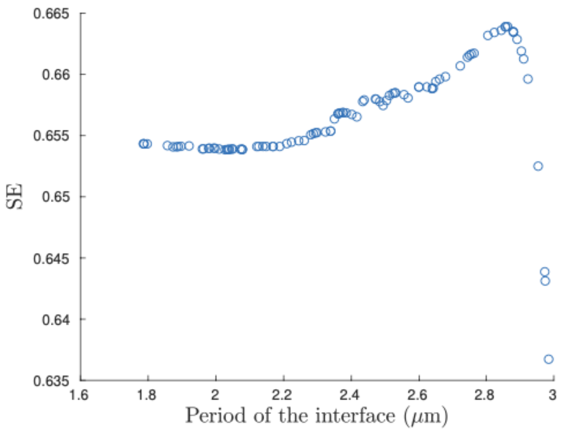
Solar Thermophotovoltaic (STPV)

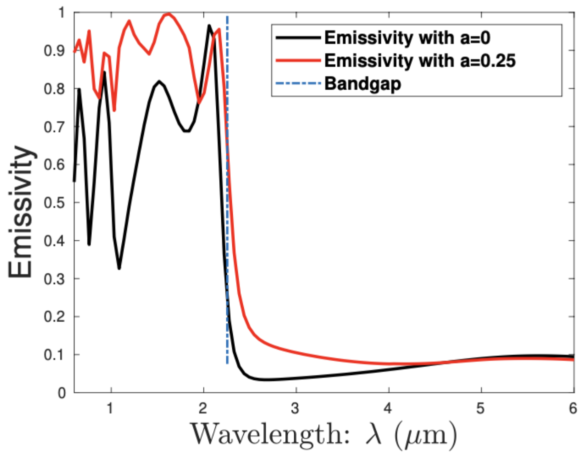








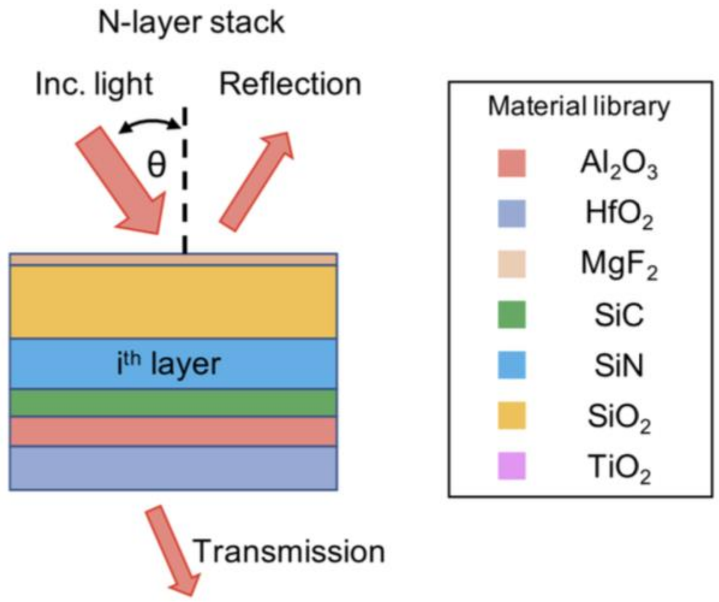




Hong et al., JCP (2017a, 2017b, 2018), APL (2019), SINUM (2021), Optics Express (2022)

Connection to the ML

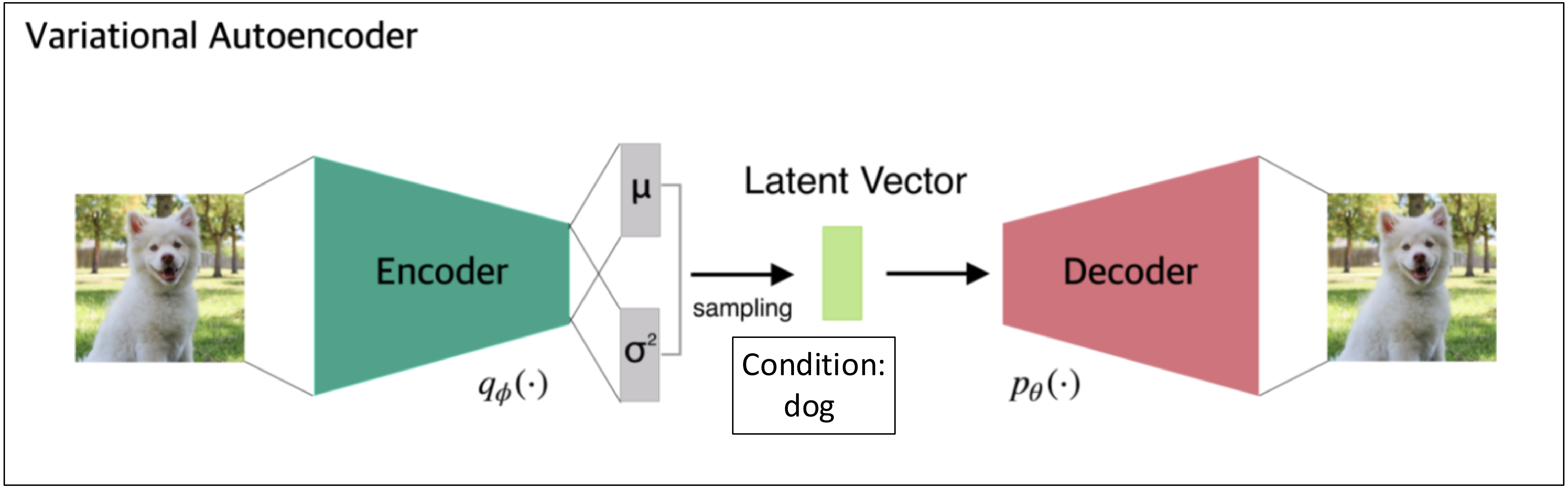
Problem description: Evaluation and design of photonic devices



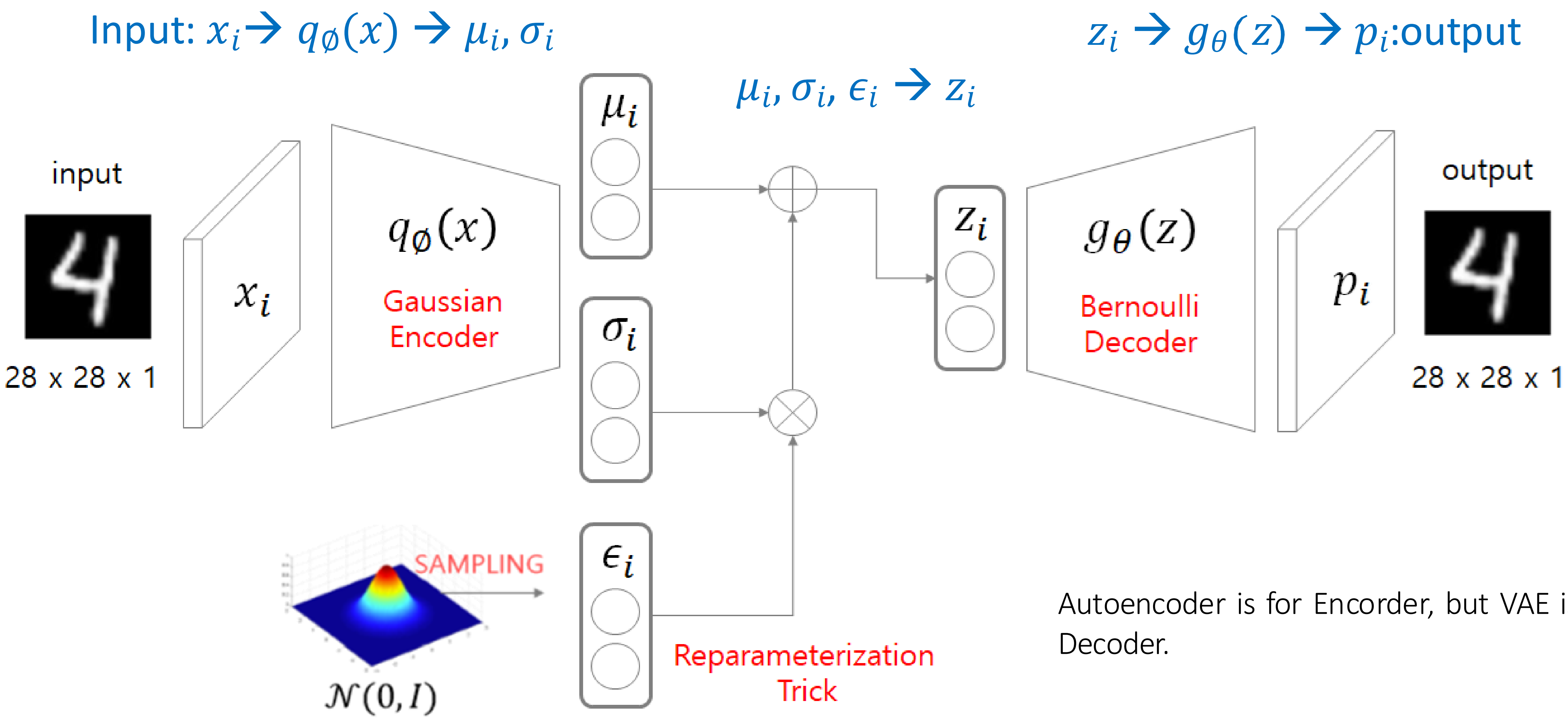
How to find an optimal design?

$$O(\mathbf{p}) = \frac{3}{\pi}\frac{1}{1200}\int_0^{\pi/3}\int_{400\text{ nm}}^{1600\text{ nm}} \mathcal{R}(\mathbf{p}, \lambda, \theta)\mathrm{d}\lambda,$$

Generative Model: Conditional Variational Autoencoder (CVAE)

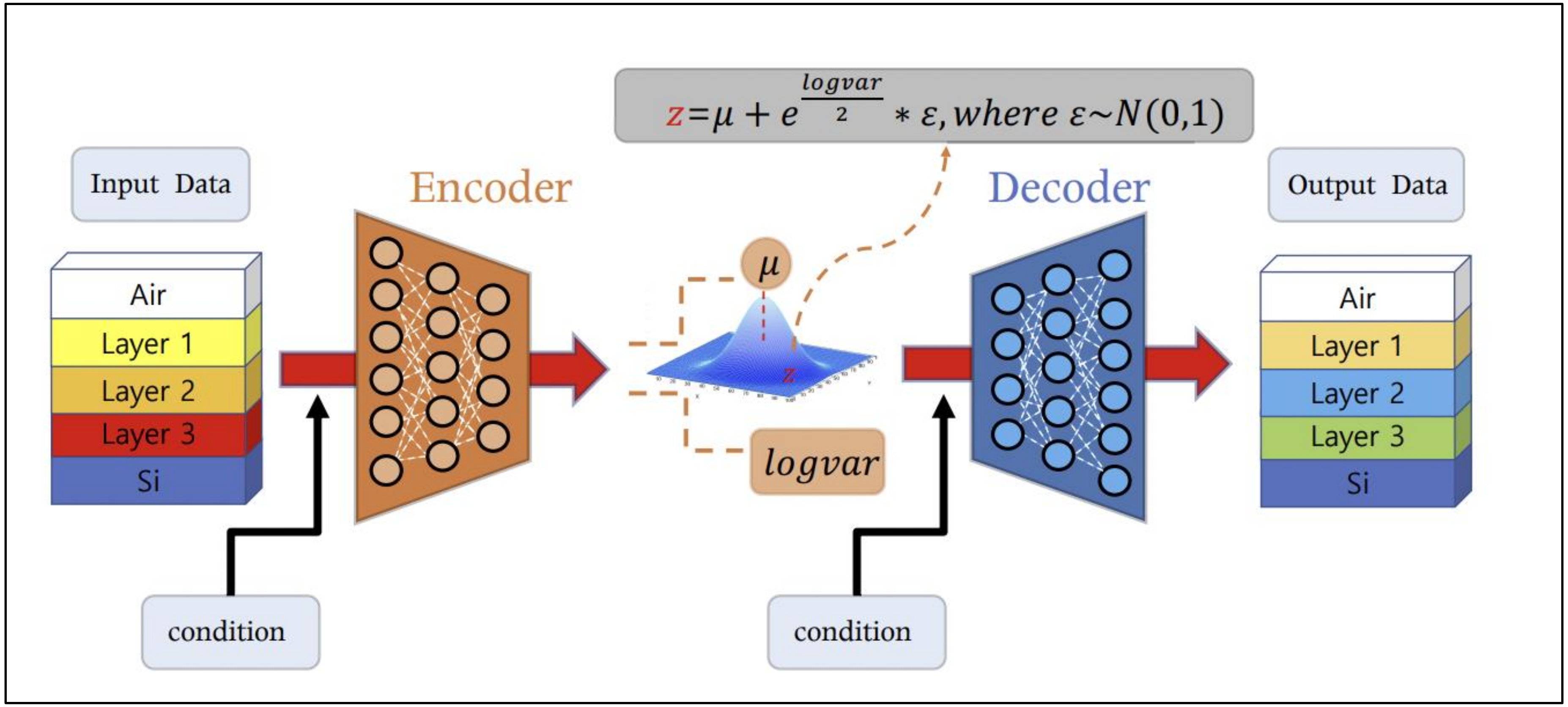


Variational Autoencoders (VAE)

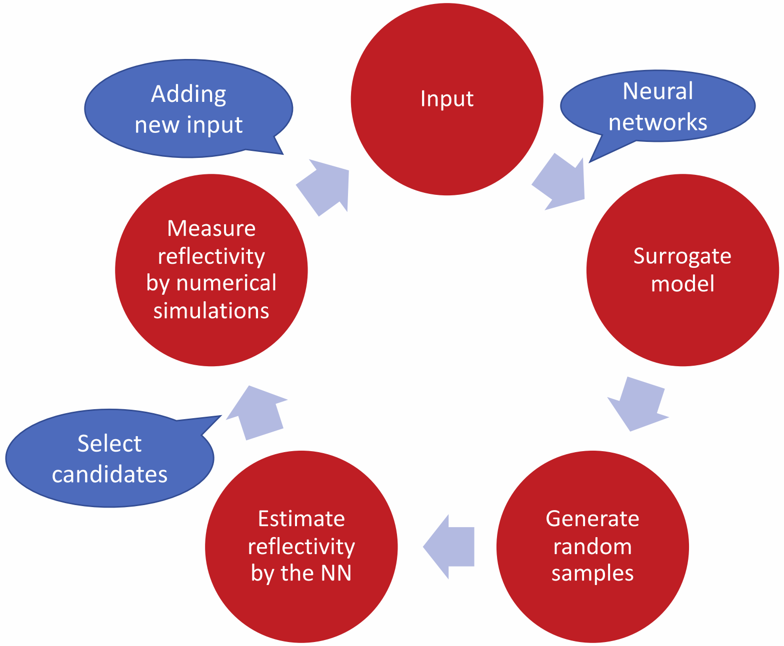


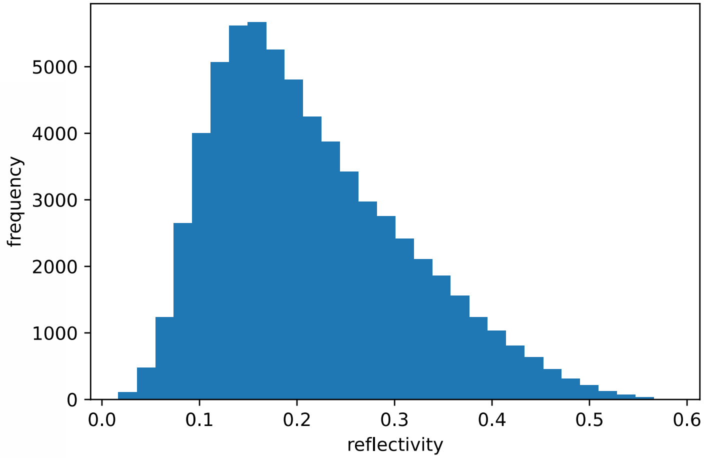
Autoencoder is for Encorder, but VAE is for Decoder.

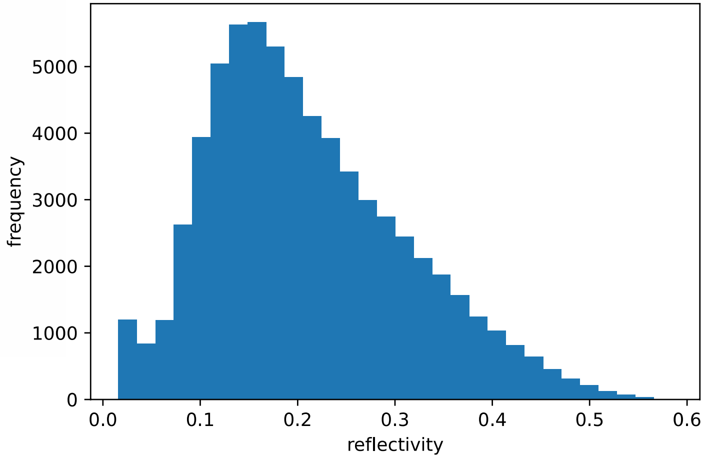
CVAE to the EM design



Active learning and results: Dataset







Active learning and results: Performance

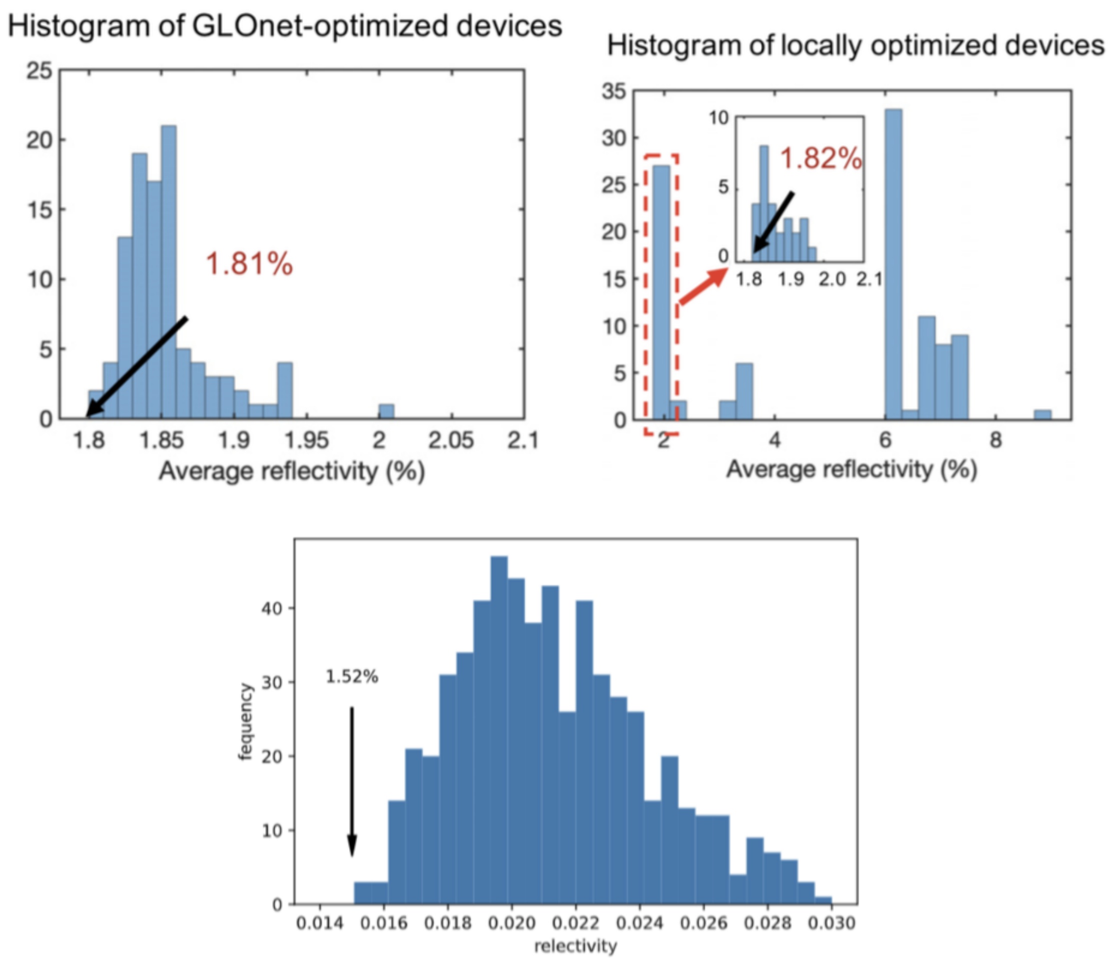
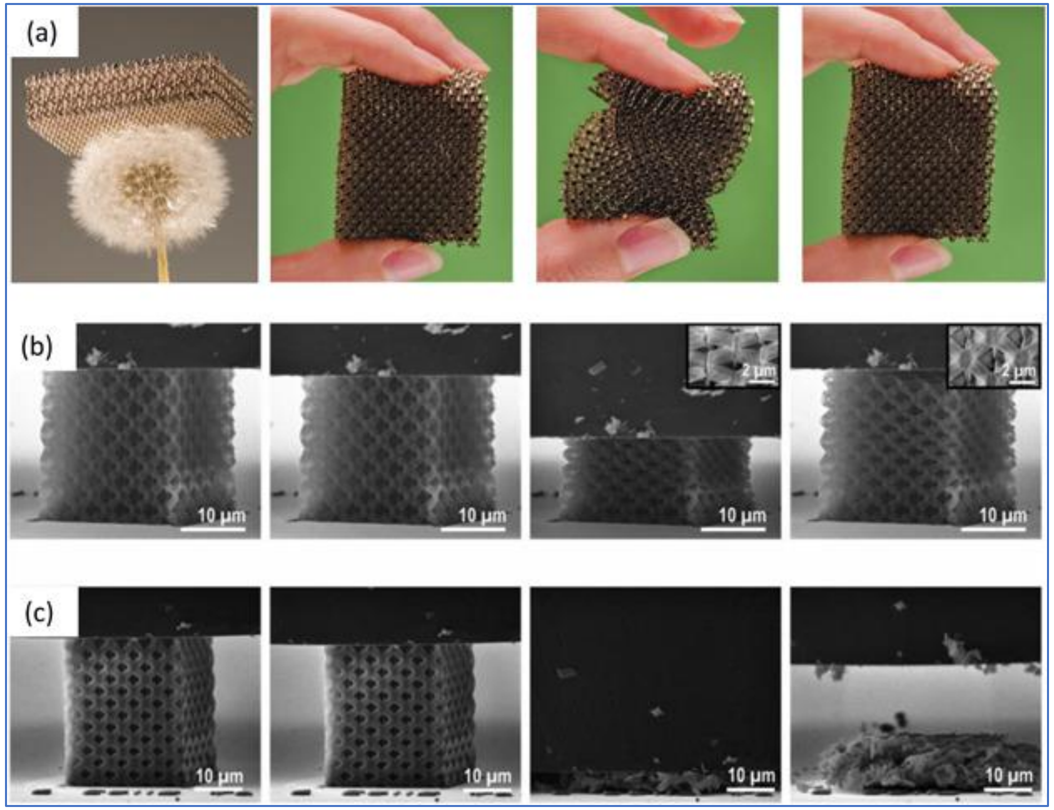
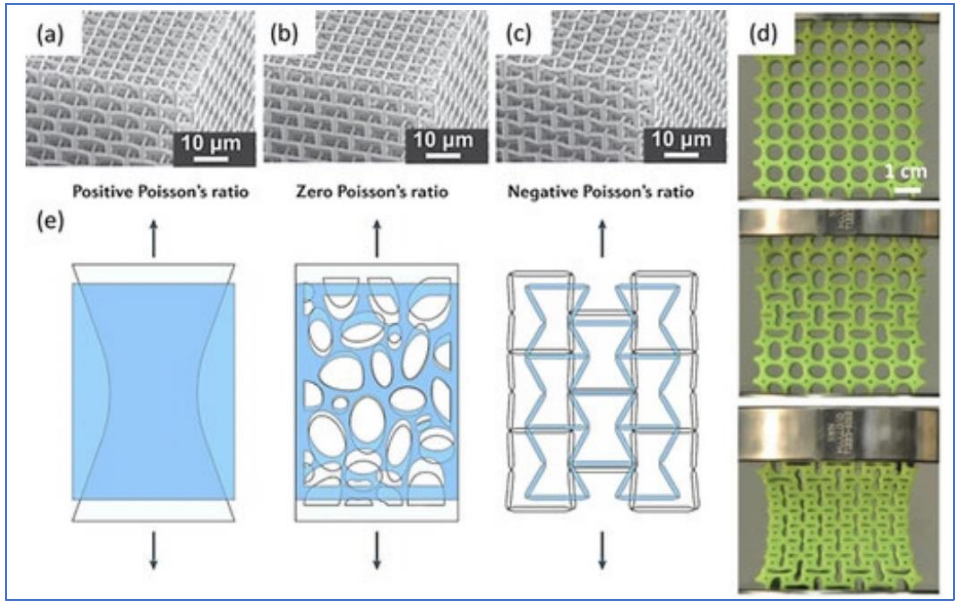
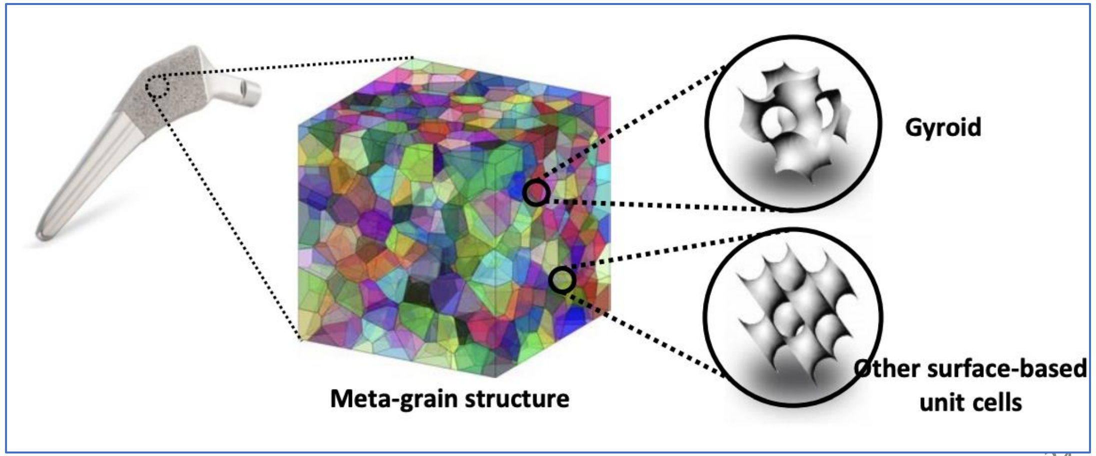


Figure: (a) Azunre et al, *New J. Phys.* (2019); (b) . Jiang and Fan, *Nanophotonics* (2021); (c). H. and Nicholls (ours).

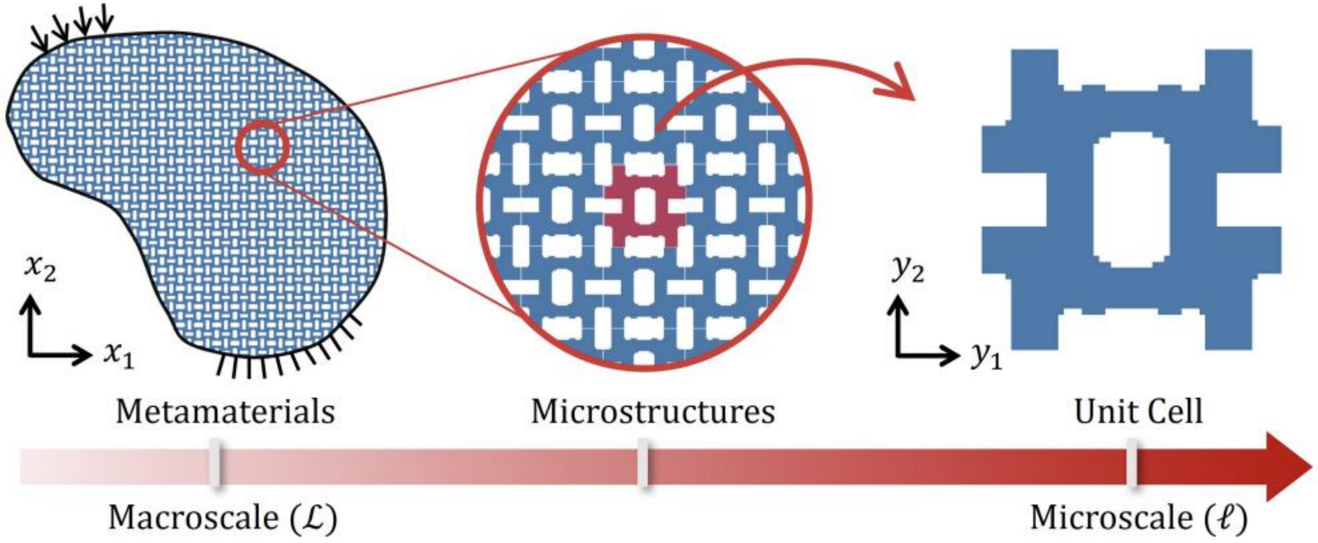
Further application: Metamaterial

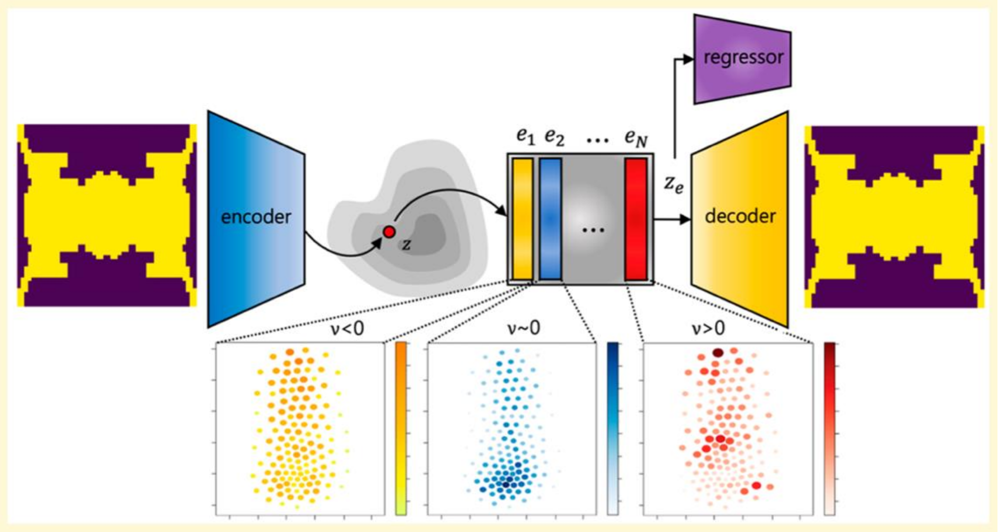


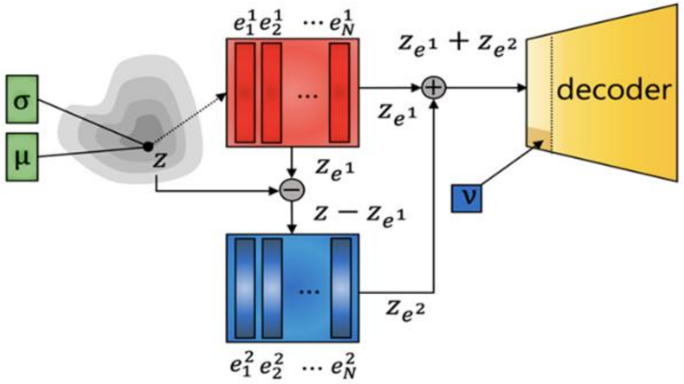




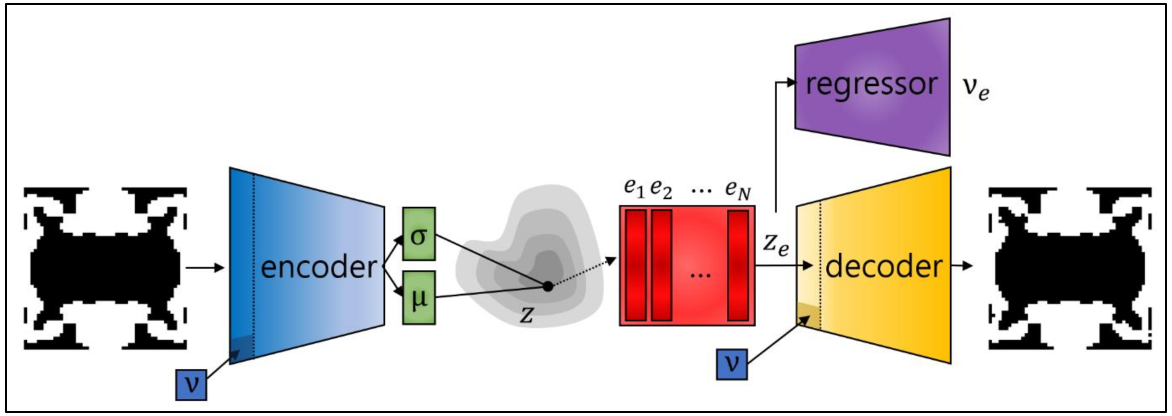
Metamaterial with Vq-CVAE



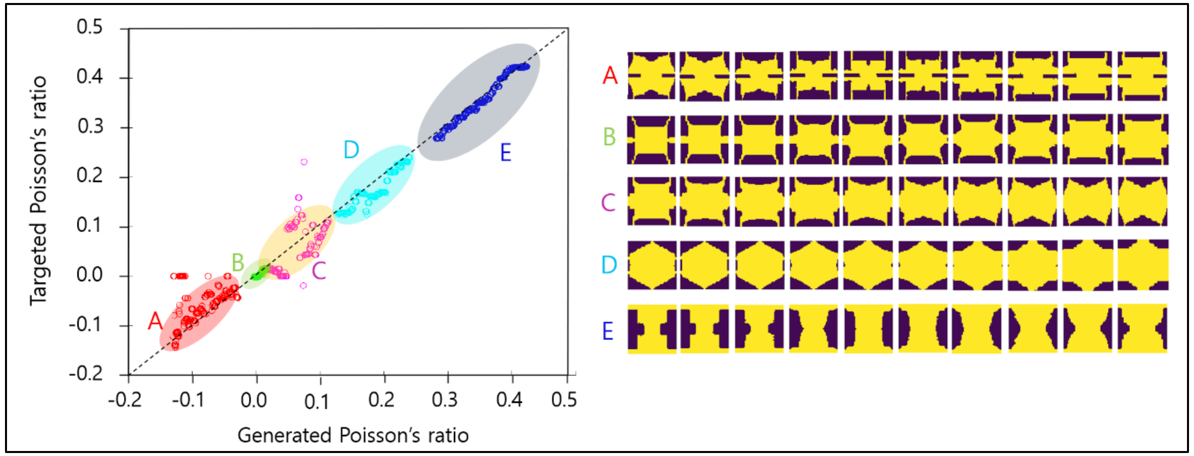




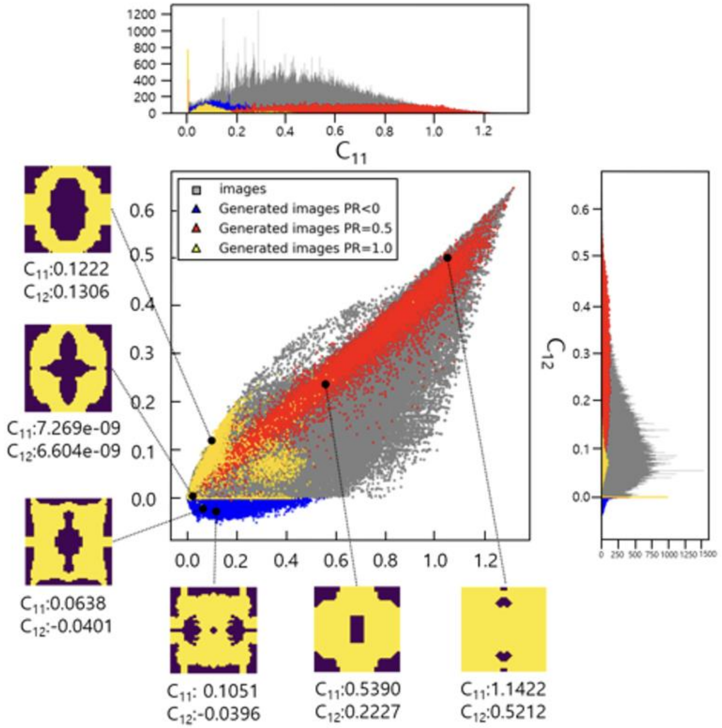
Metamaterial with Vq-CVAE



vq-cvae for metamaterial

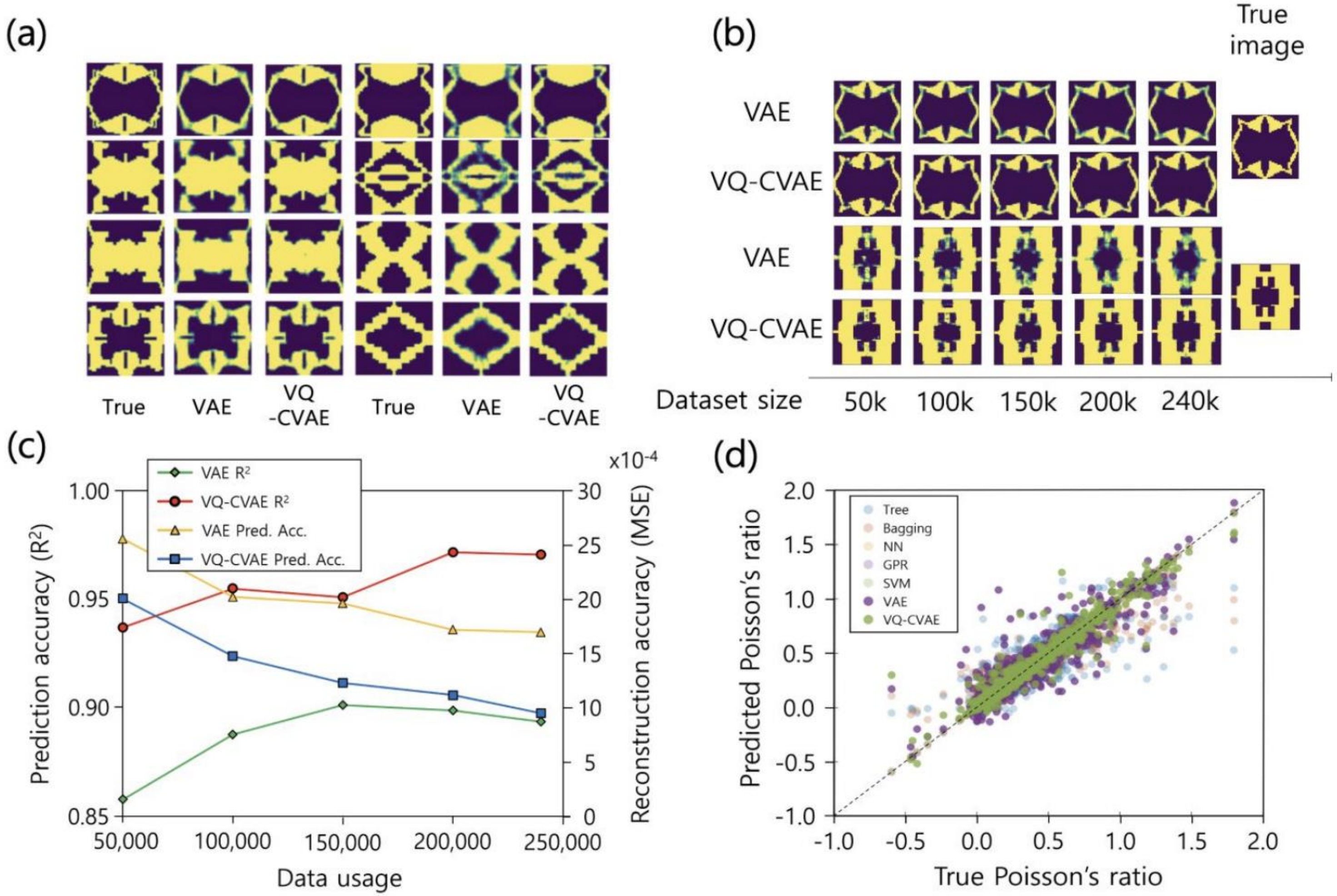


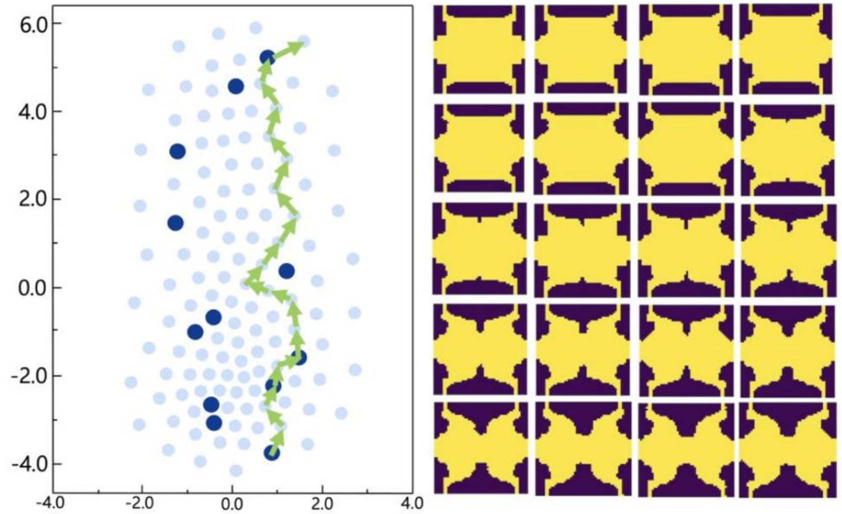
Walk in latent space



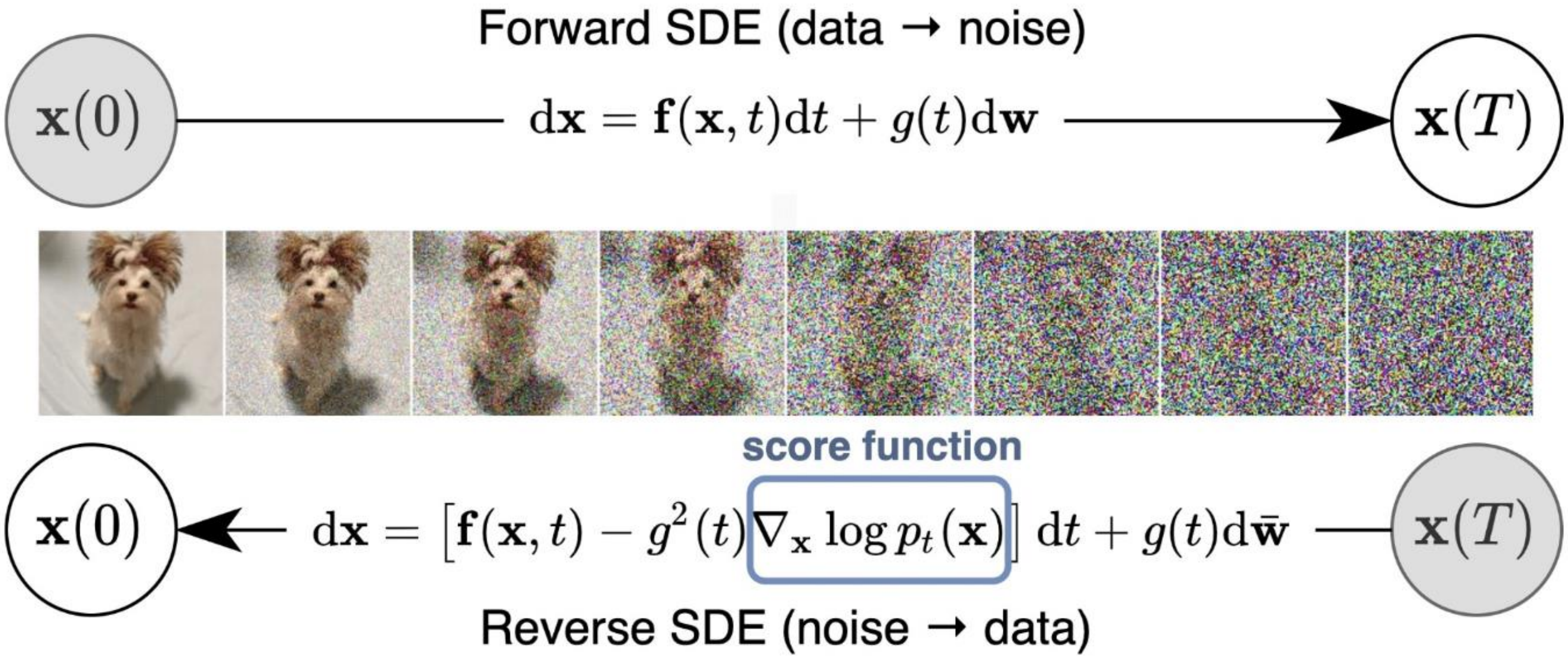
Frequency vs. Poisson's ratio

Metamaterial with Vq-CVAE

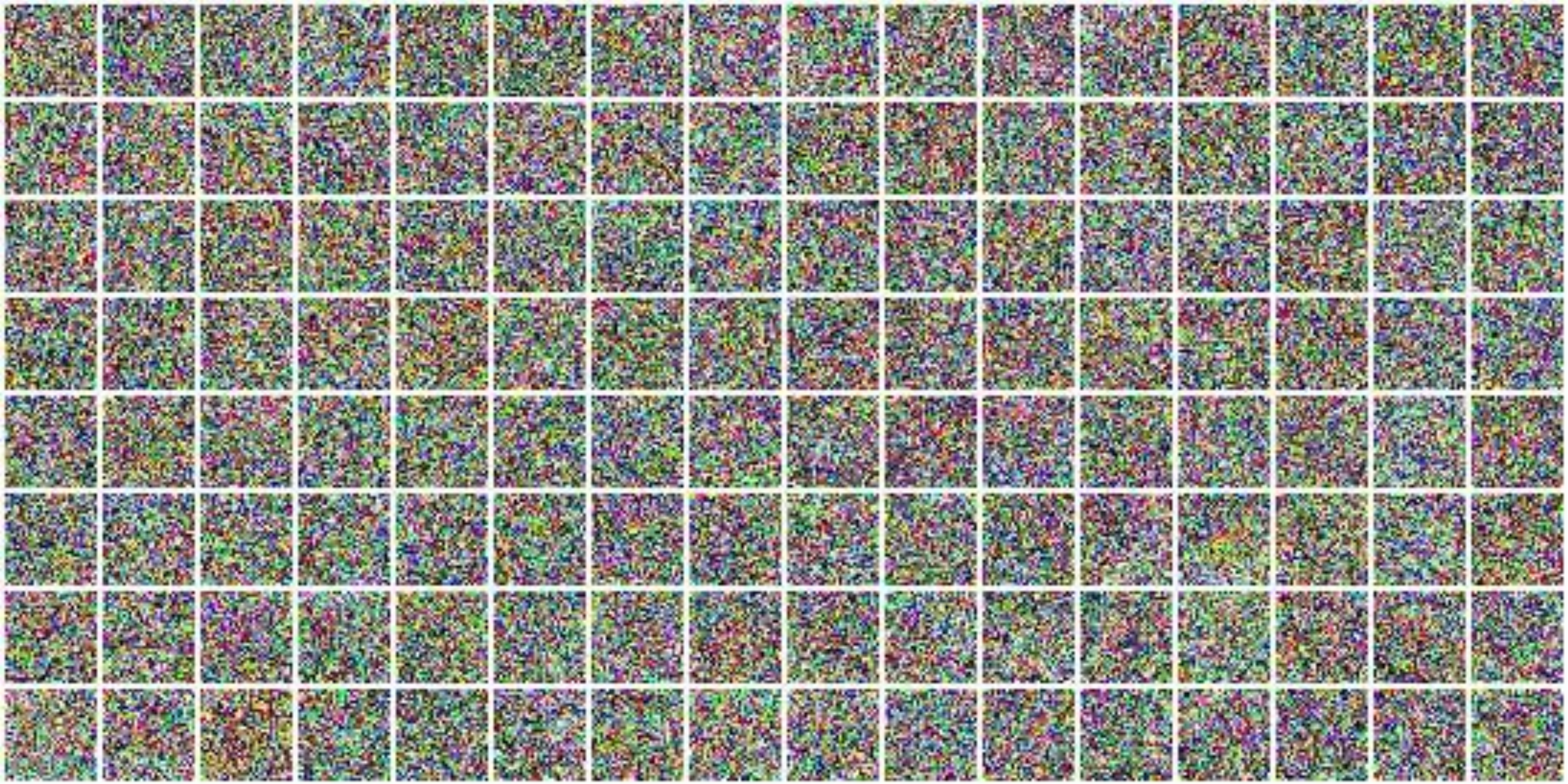




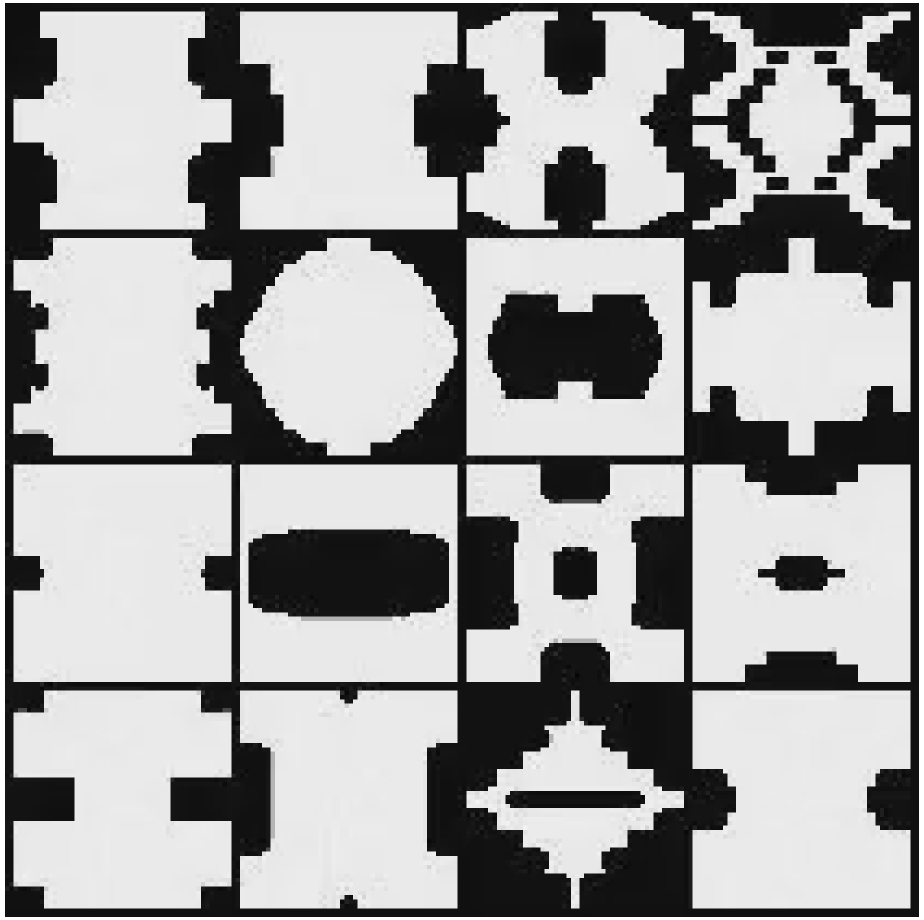
Diffusion models



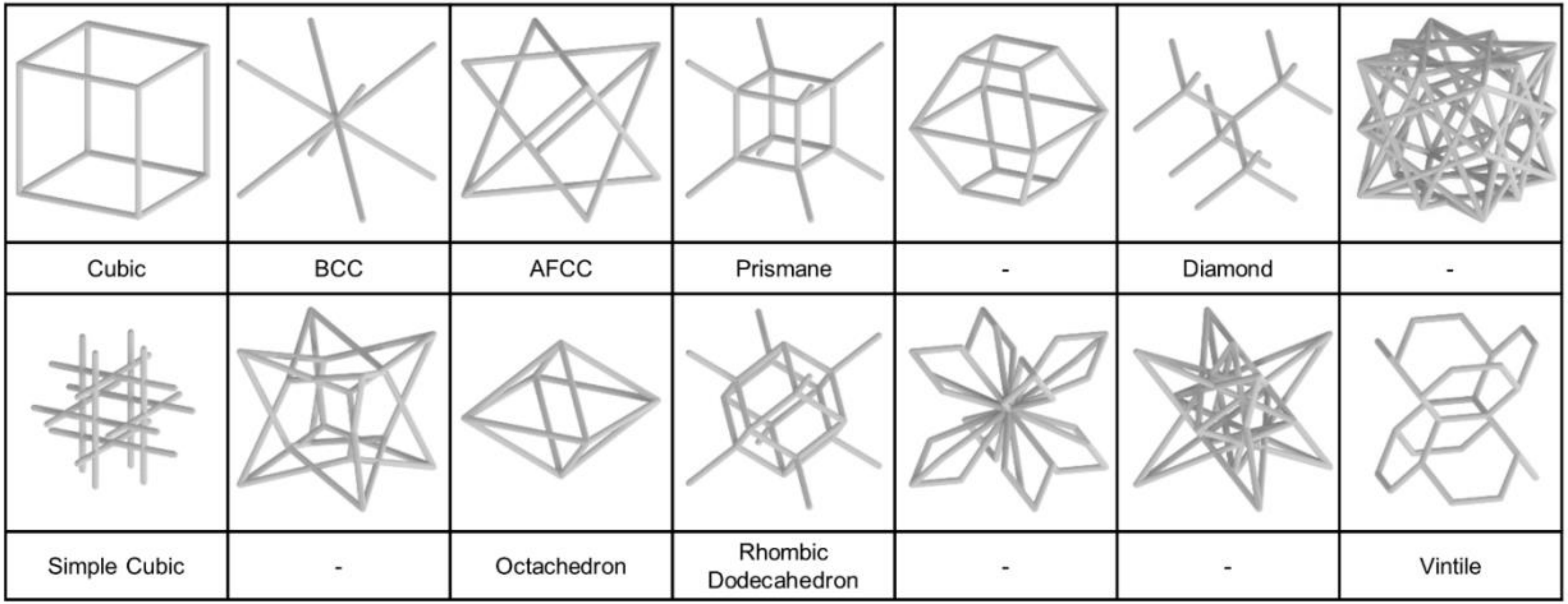
Diffusion models



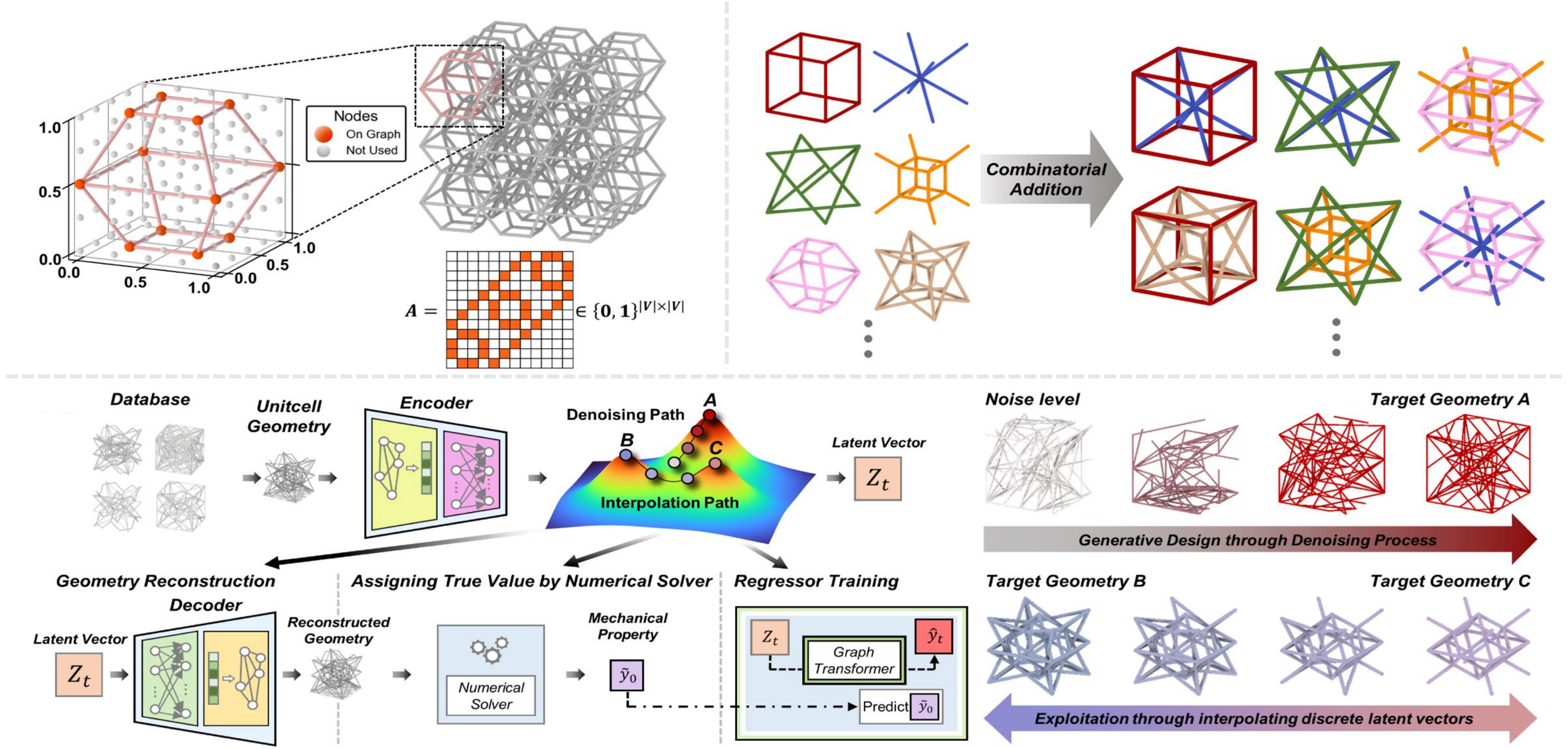
Metamaterial generation via the DM



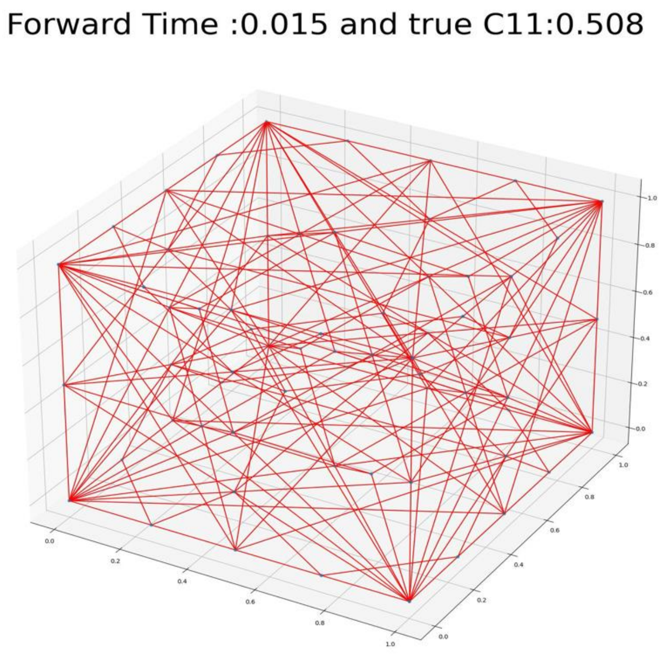
Graph Structured Database (3D metamaterials)



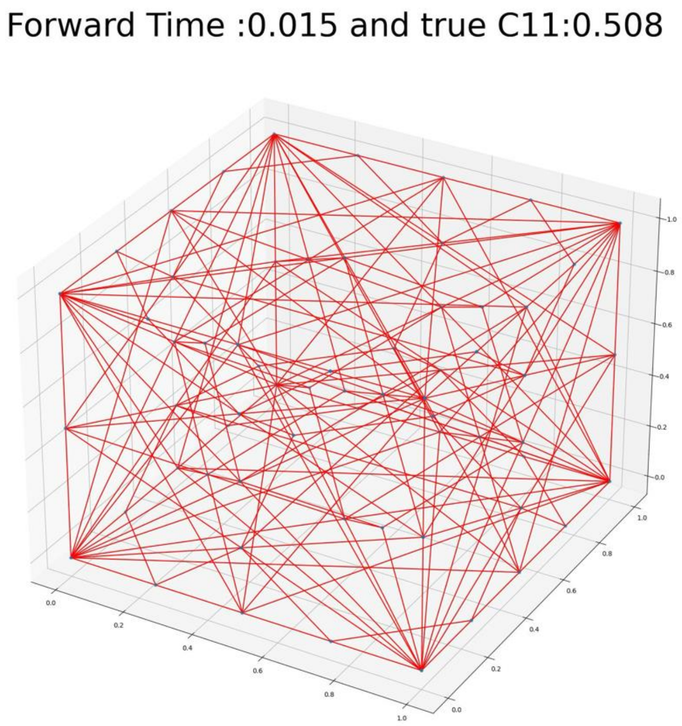
Metamaterial generation via the DM



Metamaterial generation via the DM

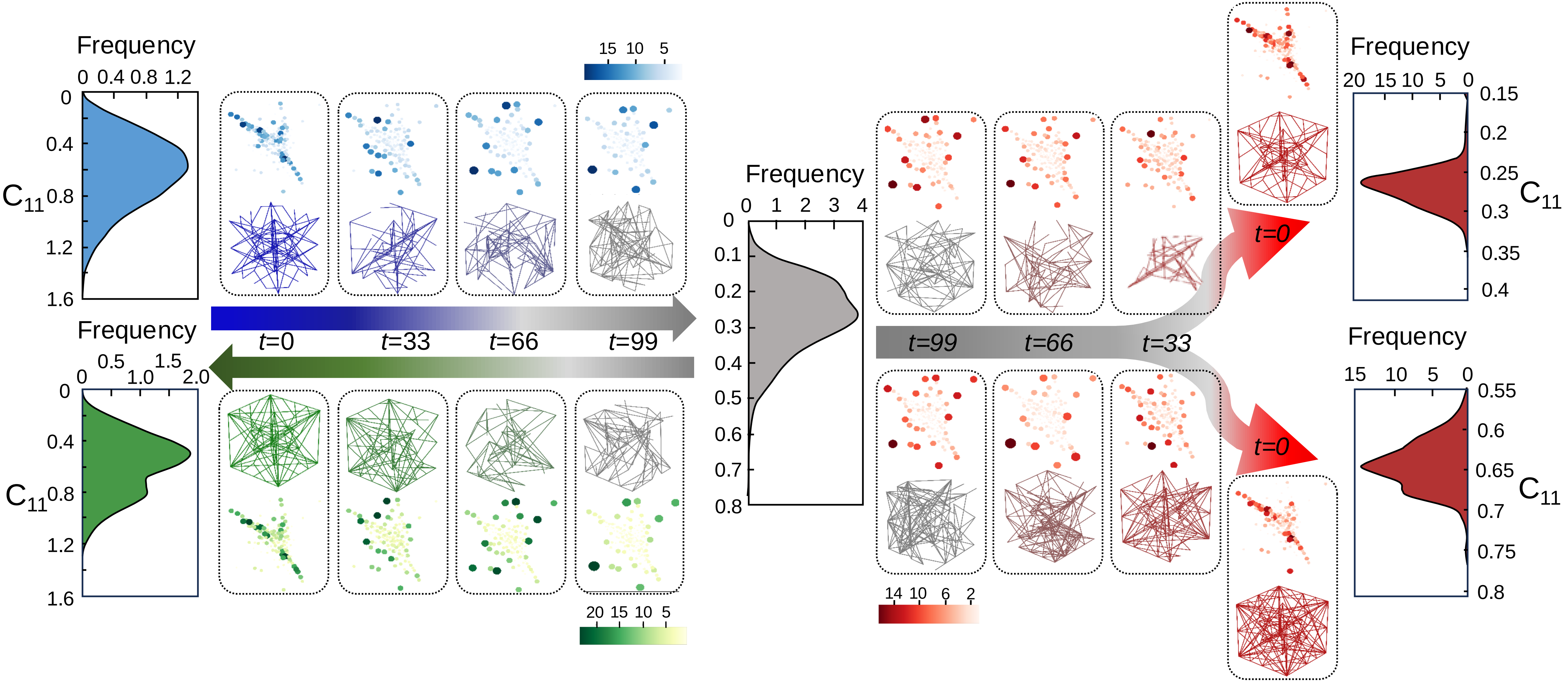


C11 = 0.85

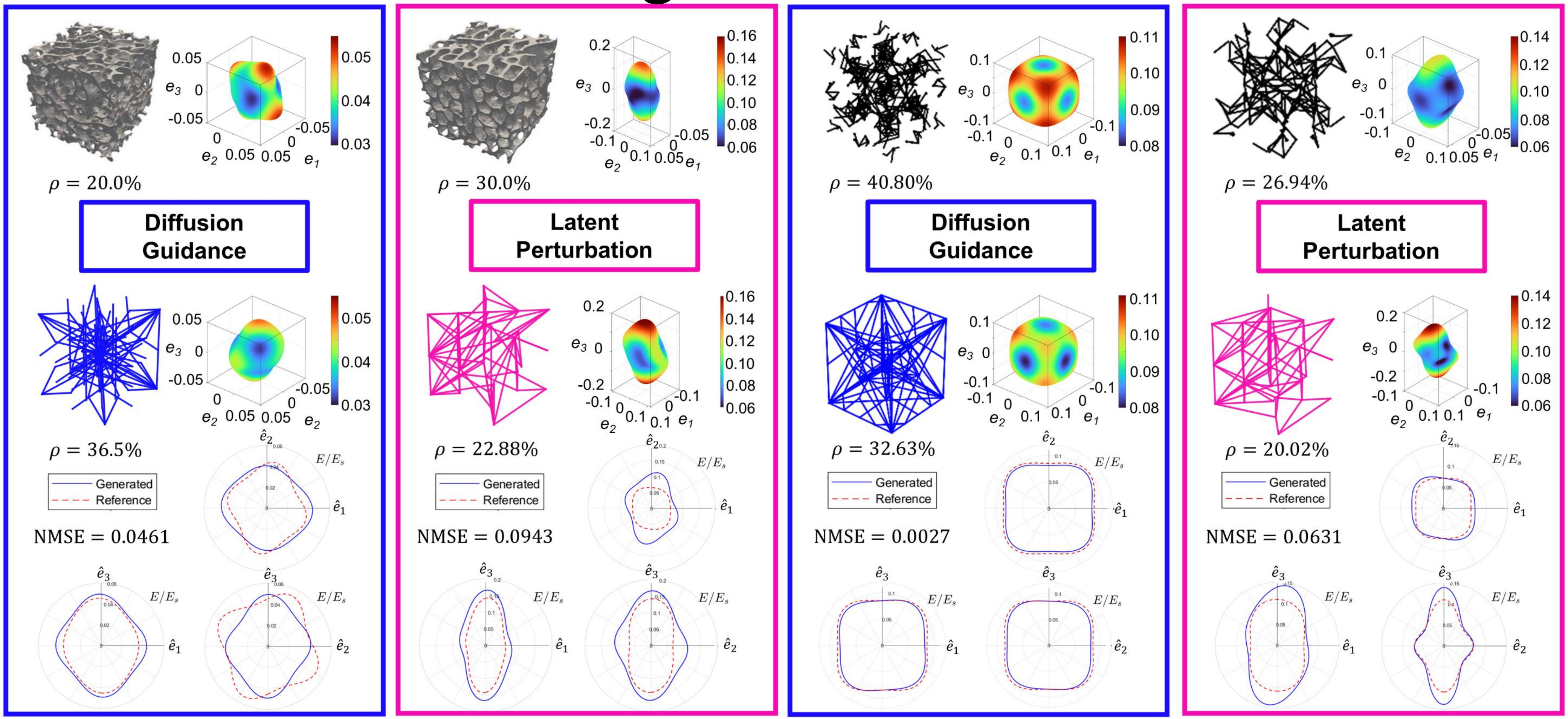


C11 = 0.42

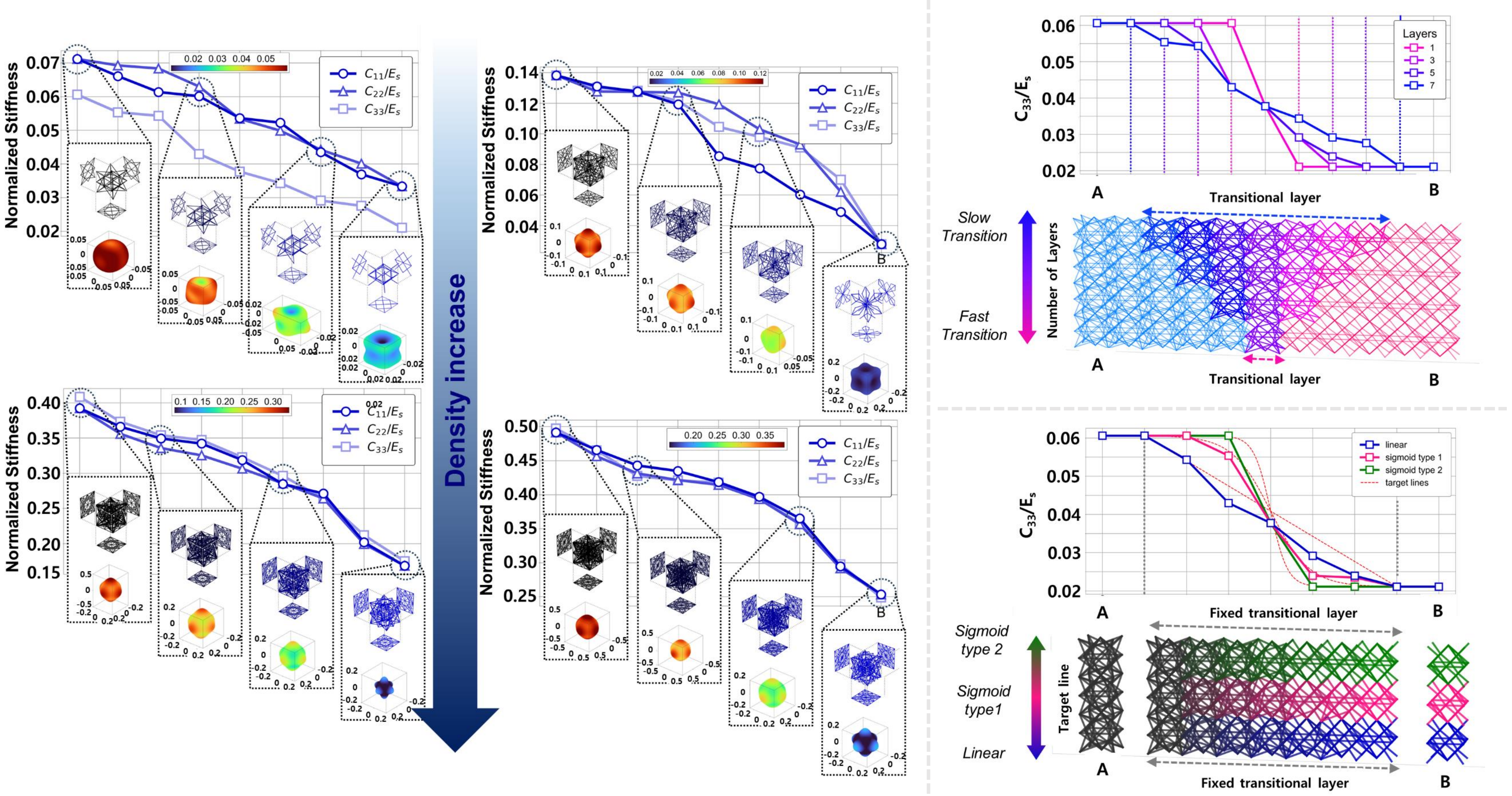
Results: dynamic evolution of graph structures



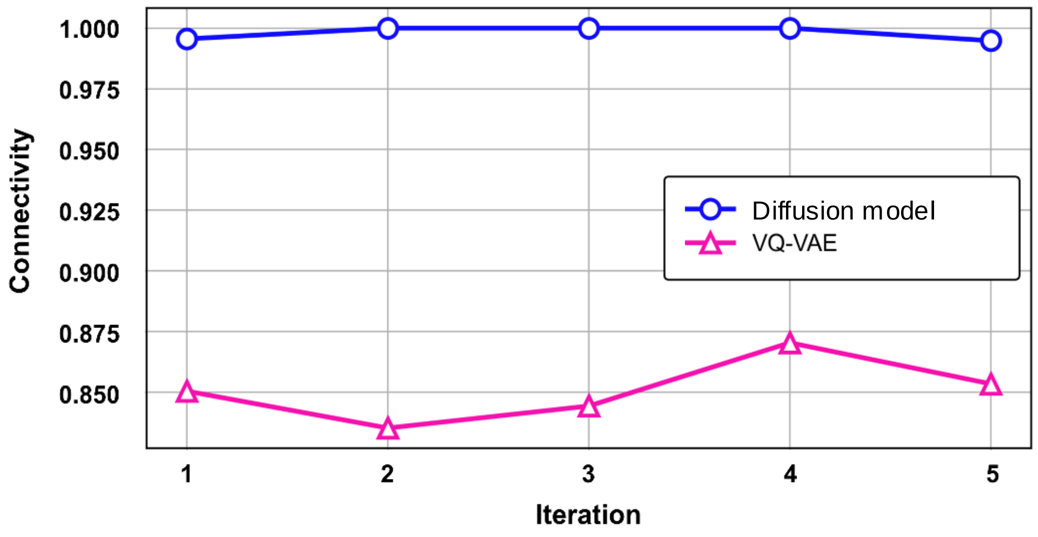
Results: inverse design



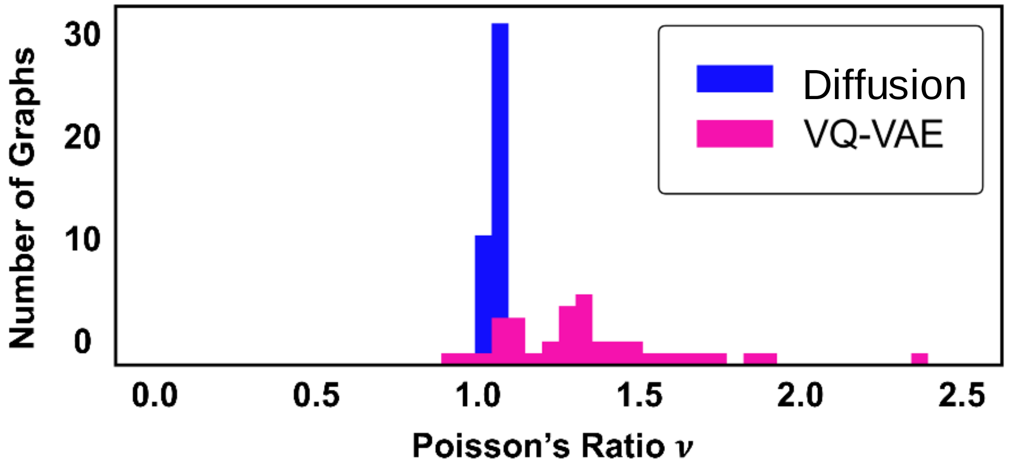
Results: functionally graded mechanical metamaterials



Comparison study between VQ-VAE and diffusion models

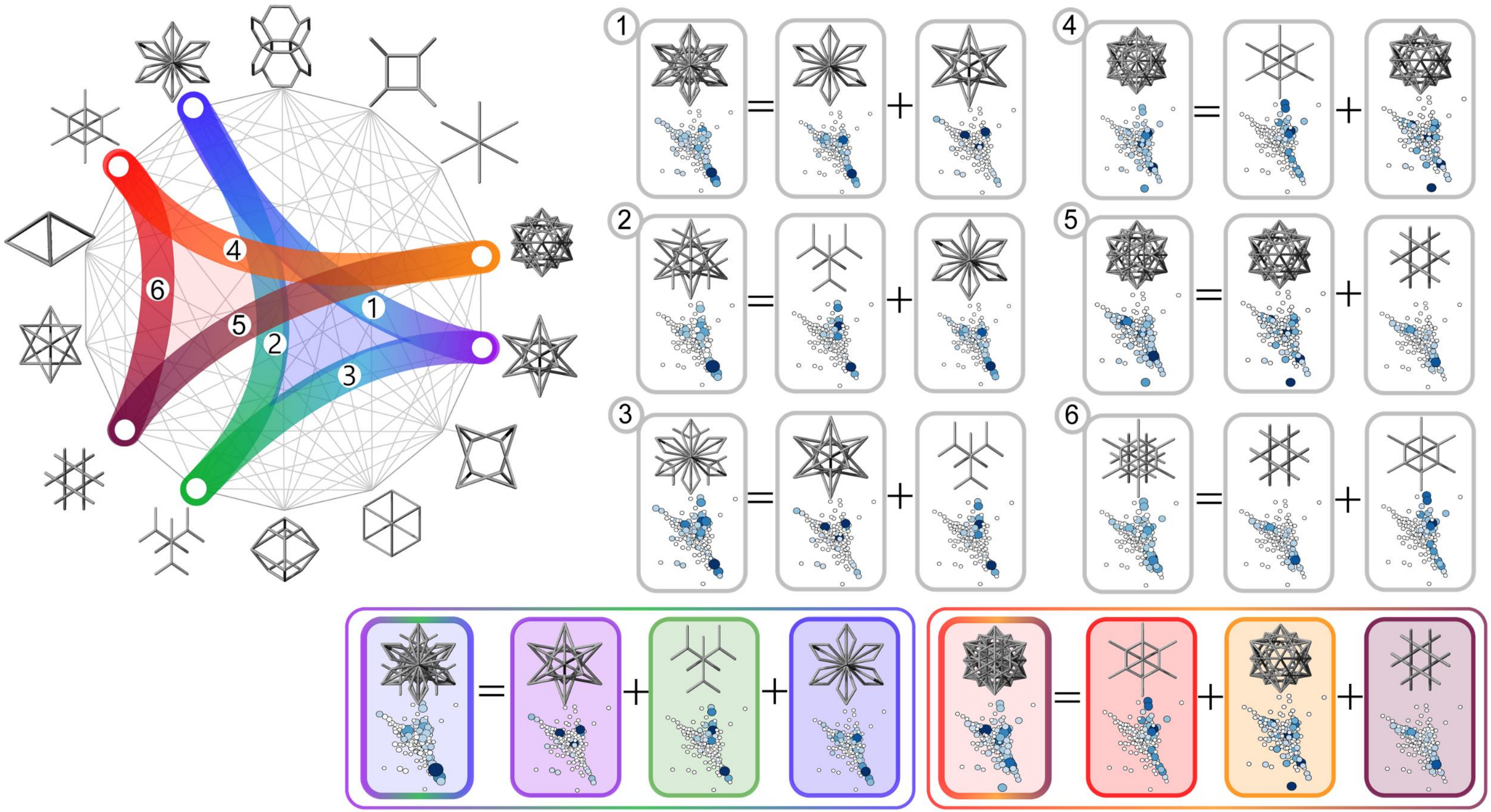


Diffusion model consistently outperforms VQ-VAE in maintaining perfect connectivity across all iterations, underscoring its robustness in generating structurally coherent samples.



Diffusion model generates graphs with Poisson's ratio values clustered around one, indicating similarity to the training dataset, VQ-VAE produces a wider distribution, reflecting its ability to create more diverse and novel graph structures.

Results: combinatorial synthesis of graph structures from basis latent



II. Scientific Machine Learning

Scientific Machine Learning

What is Scientific Machine Learning?

- **Scientific machine learning (SciML) is an emerging discipline** within the data science community. SciML **extract insights from scientific data sets** through innovative methodological solutions. SciML draws on tools from both machine learning and scientific computing to develop new methods for data analysis, and will be critical in driving the next wave of **data-driven scientific discovery** in the physical and engineering sciences.

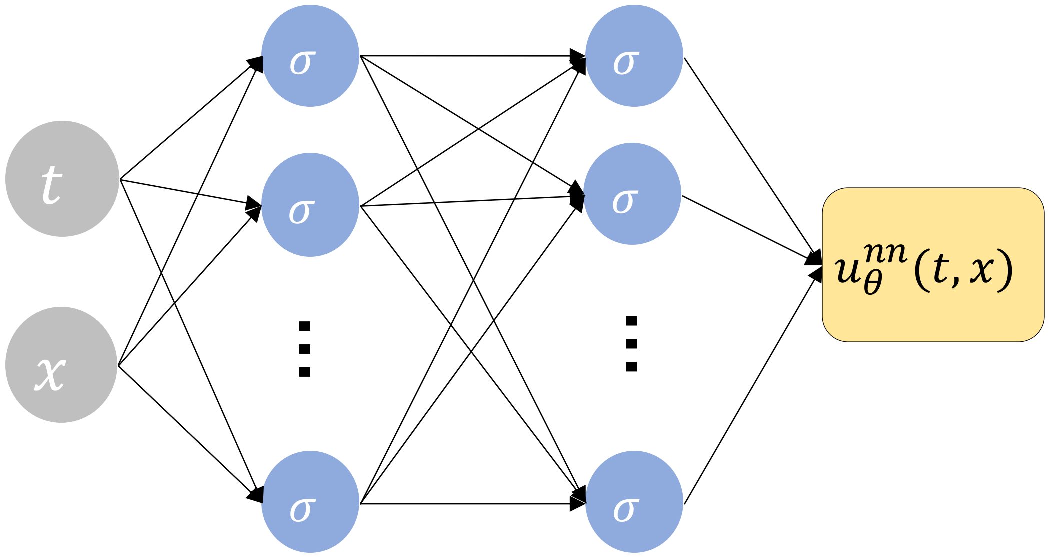
- Like scientific computing, SciML is **multidisciplinary** and leverages expertise from mathematics, computer science, and the physical sciences.

Two main streams on DNN for PDEs

- Physics-Informed Machine Learning
  - Using neural networks directly to parametrize the solution to PDEs.
  - Solve one instance of PDE at a time.
  - Models: unsupervised learning
    - **Physics Informed Neural Network (PINN), Deep Ritz Methods**

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 main streams on DNN for PDEs

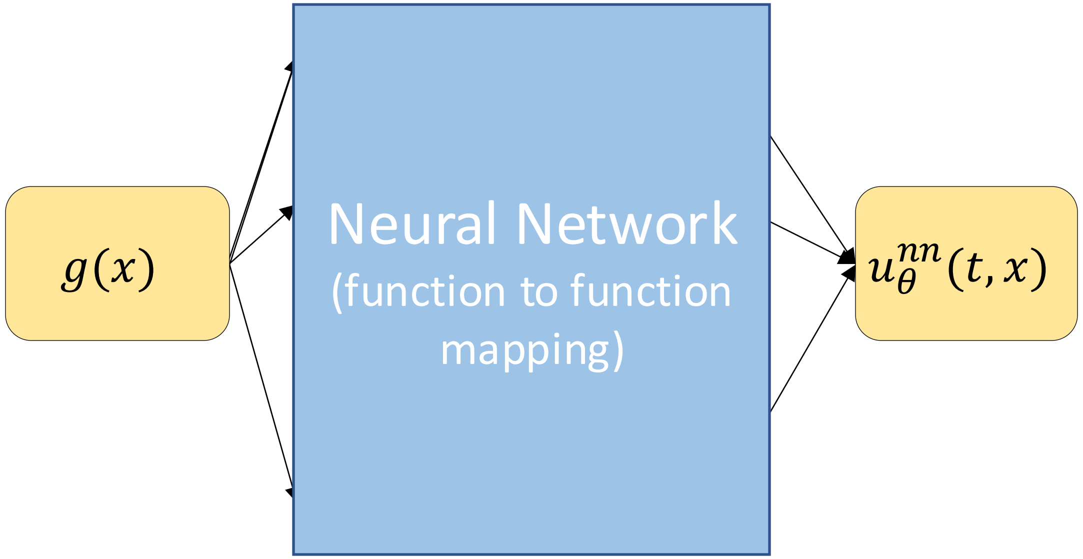
- Operator learning
  - Learning a mapping from the parameters (e.g. external force, initial, and boundary conditions) of the PDEs to the corresponding solution.
  - Learning a family of PDEs from data.
  - Models
    - Deep Operator Network (DeepONet)
    - Fourier Neural Operator (FNO)

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

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

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

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



Pros and Cons of major ML approaches

- PIMLs
  - (+) easy to implement, applicable to various domains and equations
  - (+) unsupervised learning
  - (-) predict only a single PDE instance
  - (-) Hard to impose a boundary condition

- Operator learning
  - (+) predict multiple PDE instances (parametric PDEs)
  - (+) can use the Computer Vision architectures
  - (-) supervised learning, so require a paired input-output dataset
  - (-) low accuracy on unseen data, boundary condition issue

Legendre-Galerkin Network

I will make use of this NPDE framework for my NN approximation!!

When applying the spectral element methods, we can obtain an accurate numerical solution:

$$u(x) \simeq \sum_{k=0}^{N-1} \alpha_k \varphi_k(x),$$

where N is the number of the global basis function. In practice, there are many feasible choices of the basis functions such as Fourier series $(\exp(ikx))$, Chebyshev polynomial $(T_k(x))$, or Legendre polynomial $(L_k(x))$.

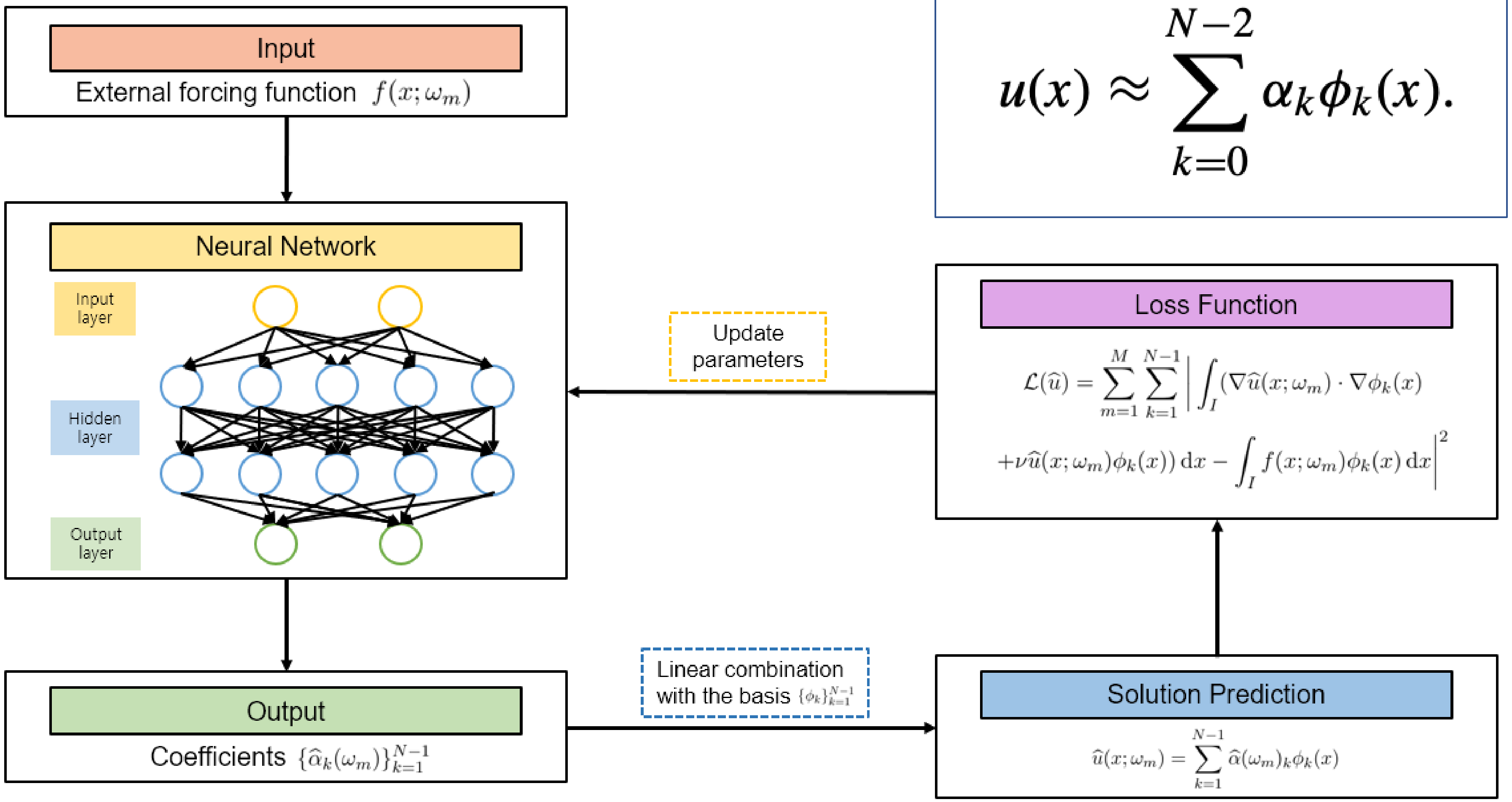
For given data (e.g., forcing terms, boundary conditions, etc.), our neural network predicts only the coefficients of the basis functions. Subsequently, we generate infinitely many datasets through reconstruction:

$$\alpha_k \implies u_N = \sum_{k=0}^{N-1} \alpha_k \varphi_k.$$

Unsupervised Operator Network

ML + Numerical PDEs

$$u(x) \approx \sum_{k=0}^{N-2} \alpha_k \phi_k(x).$$



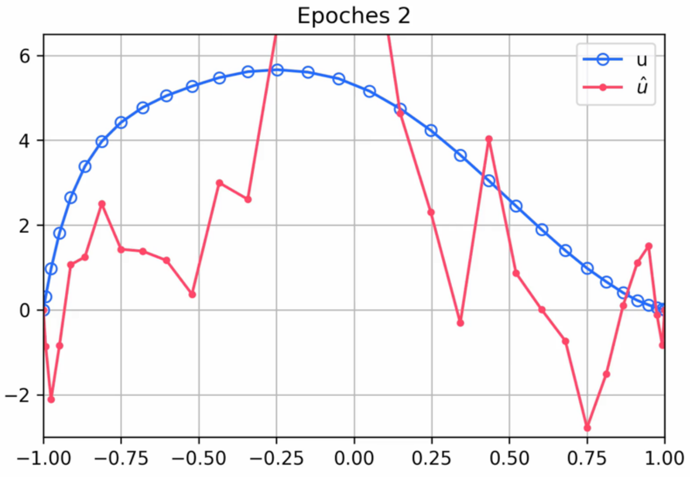
Hong et al., IEEE Access (2023), Int. J. Numer. Anal. Model. (2024)

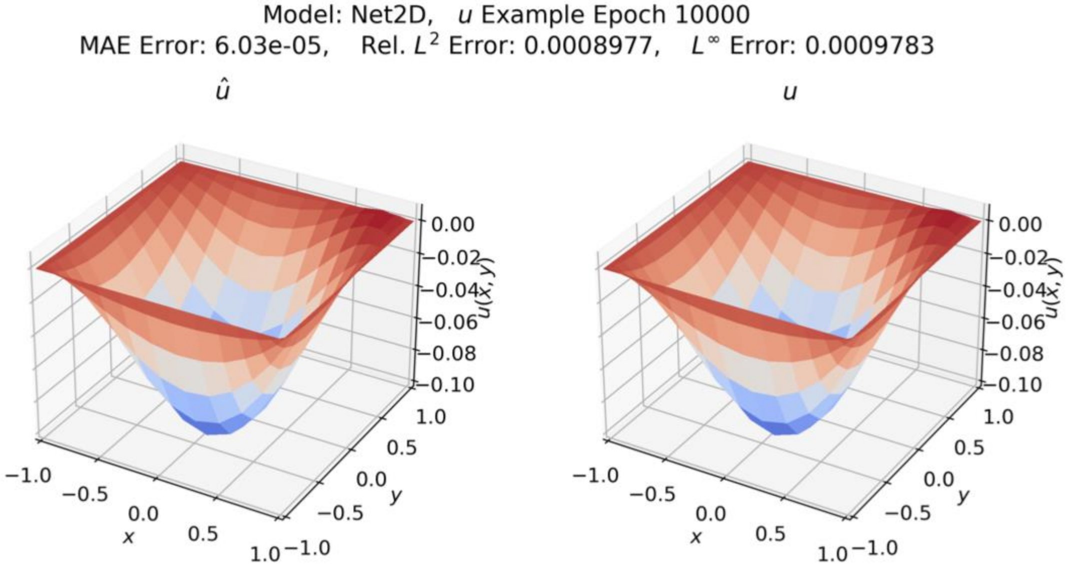
Numerical Results

$$\begin{cases} -\varepsilon u_{xx} - u_x = f, \\ u(-1) = u(1) = 0. \end{cases}$$

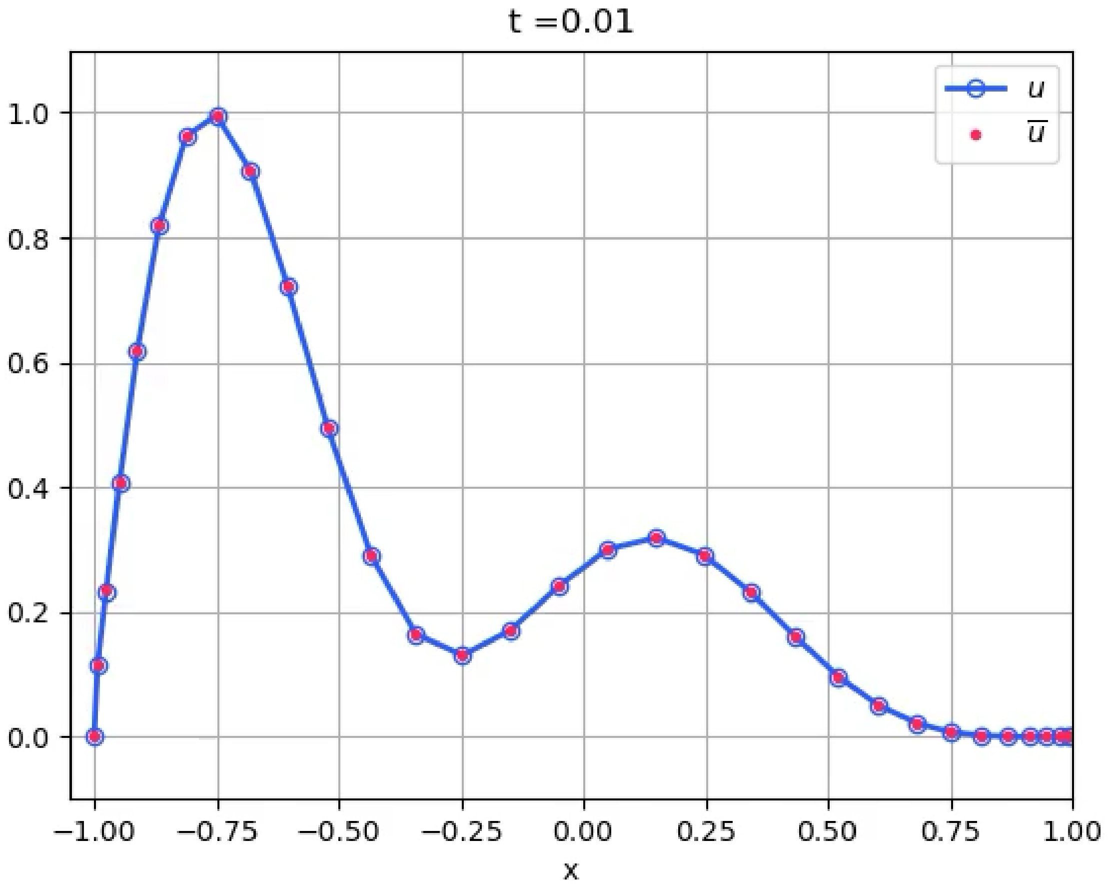
$$\begin{cases} u_{xx} + k_u u = f(x), \\ u'(-1) = u'(1) = 0, \end{cases}$$

$$\begin{cases} -\varepsilon u_{xx} + u u_x = f, \\ u(-1) = u(1) = 0. \end{cases}$$





Time dependent problem



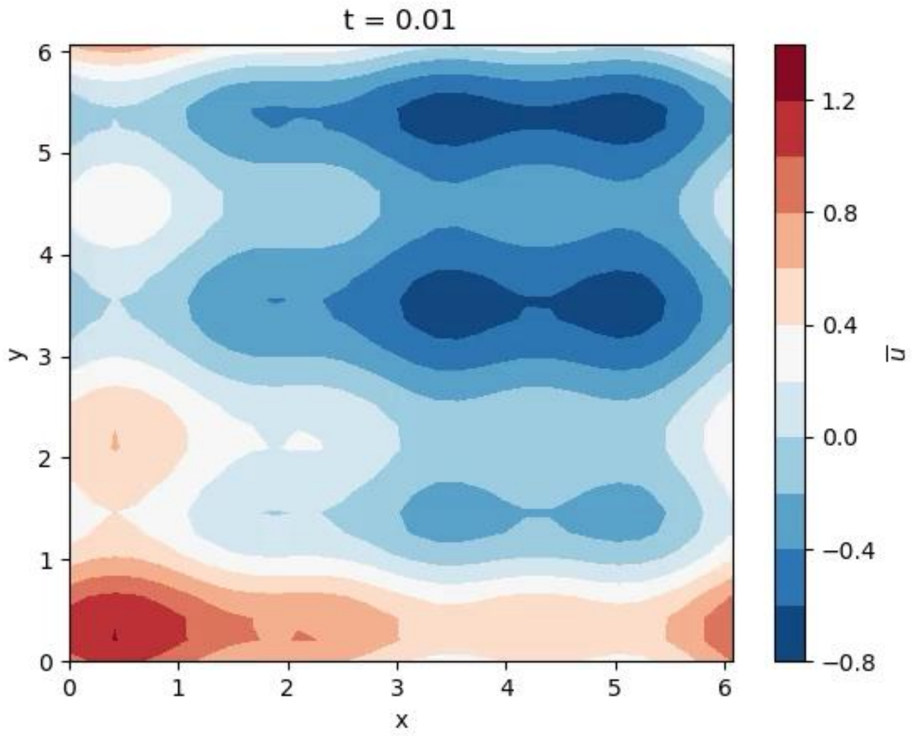
$$
\begin{aligned}
&u_t - \nu u_{xx} - u_x = 0, \quad \text{for} \quad t > 0, x \in \Omega, \\
&u = u_0(x), \quad \text{for} \quad t = 0, \quad x \in \Omega, \\
&u(-1) = 0 = u(1), \quad \text{for} \quad t \leq 0.
\end{aligned}
$$

Linear convection diffusion with small viscousity. We expect the boundary layer near x=-1.

Hong et al., Comput. Methods Appl. Mech. Eng. (2024)

2D KS and NSE equations

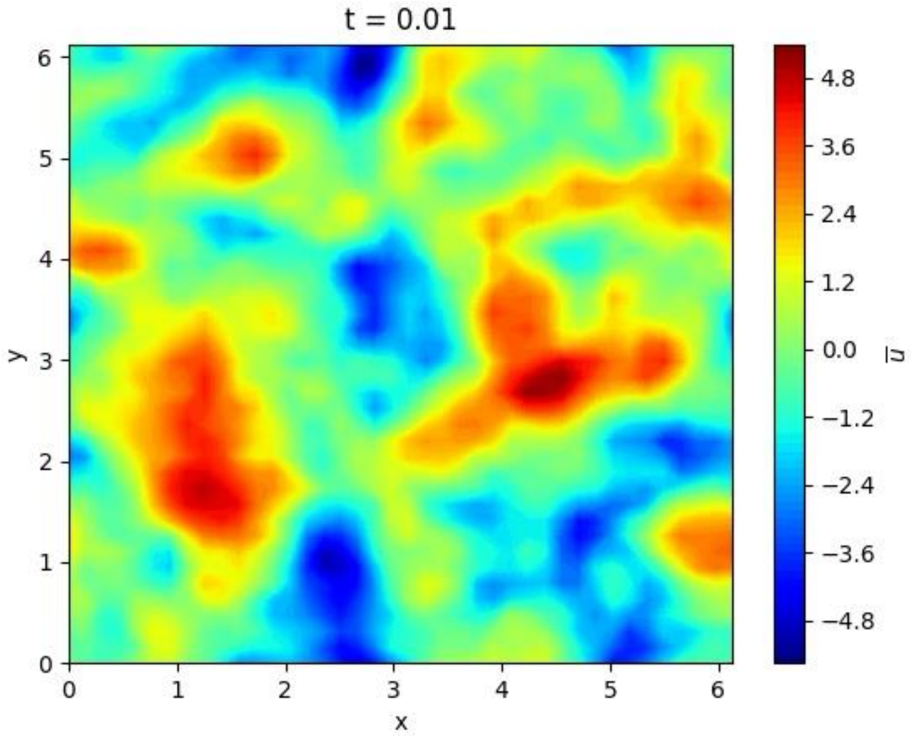
Predicted solution



$$u_t + \Delta u_{xx} + \Delta^2 u + +|\nabla u|^2 = 0, \quad \text{for} \quad t > 0, x \in \Omega,$$
$$u = u_0(x), \quad \text{for} \quad t = 0, \quad x \in \Omega,$$

2D Kuramoto-Sivashinsky equations

Predicted solution



$$\begin{aligned} \partial_t w(x,t) + u(x,t)\cdot\nabla w(x,t) &= \nu\Delta w(x,t) + f(x), && x \in (0,1)^2, t \in (0,T] \\ \nabla \cdot u(x,t) &= 0, && x \in (0,1)^2, t \in [0,T] \\ w(x,0) &= w_0(x), && x \in (0,1)^2 \end{aligned}$$

2D Navier-Stokes equations

Hong et al., Comput. Methods Appl. Mech. Eng. (2024)

2D NSE

Incompressible Navier–Stokes equation (NSE) in its vorticity form for a viscous on the unit torus (2D),

$$\boxed{\begin{aligned} & w_t + \mathbf{u}\cdot\nabla w - \frac{1}{Re}\Delta w = f, \quad \text{for} \quad t>0, x\in\Omega, \\ & \mathbf{u} = \nabla\times\psi\hat{\mathbf{z}} \\ & \Delta\psi = w \\ & w = w_0(x), \quad \text{for} \quad t=0, \quad x\in\Omega, \end{aligned}}$$

$$\Rightarrow \quad w_N(x_n, y_m) = \frac{1}{(2\pi)^2}\sum_{\xi_y=-N/2+1}^{N/2}\sum_{\xi_x=-N/2+1}^{N/2} e^{\mathrm{i}(\xi_x x_n + \xi_y y_m)}\alpha_{\xi_x,\xi_y}, \quad n,m = 1\cdots, N.$$

$$\frac{\mathcal{F}_{\xi_x,\xi_y}(\widetilde{w}) - \mathcal{F}_{\xi_x,\xi_y}(w^r)}{\Delta t} + \frac{|\mathbf{k}|^2}{2Re}\mathcal{F}_{\xi_x,\xi_y}(\widetilde{w}) + \mathcal{F}_{\xi_x,\xi_y}(w^r) + \mathrm{i}\mathbf{k}\cdot\left(\mathcal{F}_{\xi_x,\xi_y}(u^r w^r), \mathcal{F}_{\xi_x,\xi_y}(v^r w^r)\right) - \mathcal{F}_{\xi_x,\xi_y}(f) = 0.$$

Find velocity with

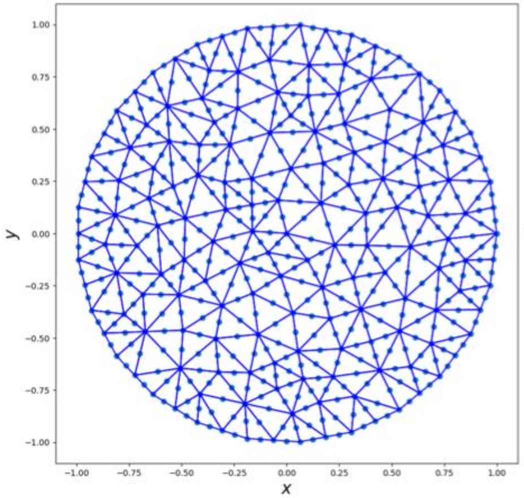
$$\boxed{\begin{aligned} & \Delta\widetilde{\psi} = \widetilde{w} \\ & \widetilde{\mathbf{u}} = \nabla\times\widetilde{\psi}\hat{\mathbf{z}}. \end{aligned}}$$

$$\Rightarrow \quad \frac{\mathcal{F}_{\xi_x,\xi_y}(w^{r+1}) - \mathcal{F}_{\xi_x,\xi_y}(w^r)}{\Delta t} = -\frac{|\mathbf{k}|^2}{2Re}\mathcal{F}_{\xi_x,\xi_y}(w^{r+1}) + \mathcal{F}_{\xi_x,\xi_y}(w^r) \quad - \frac{\mathrm{i}\mathbf{k}}{2}\cdot\left((\mathcal{F}(u^r w^r), \mathcal{F}_{\xi_x,\xi_y}(v^r w^r)) + (\mathcal{F}_{\xi_x,\xi_y}(\widetilde{u}\widetilde{w}), \mathcal{F}_{\xi_x,\xi_y}(\widetilde{v}\widetilde{w}))\right) + \mathcal{F}_{\xi_x,\xi_y}(f),$$

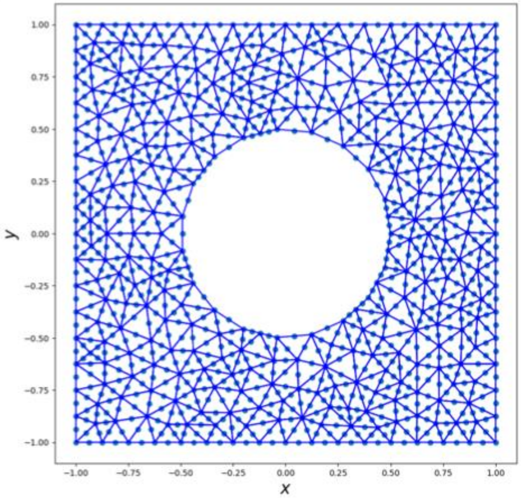
$$\Rightarrow \quad w_N^{r+1}(x_n, y_m) = \frac{1}{(2\pi)^2}\sum_{\xi_y=-N/2+1}^{N/2}\sum_{\xi_x=-N/2+1}^{N/2} e^{\mathrm{i}(\xi_x x_n + \xi_y y_m)}\mathcal{F}_{\xi_x,\xi_y}(w^{r+1}), \quad n,m = 0\cdots, N-1.$$

Finite Element Operator Network

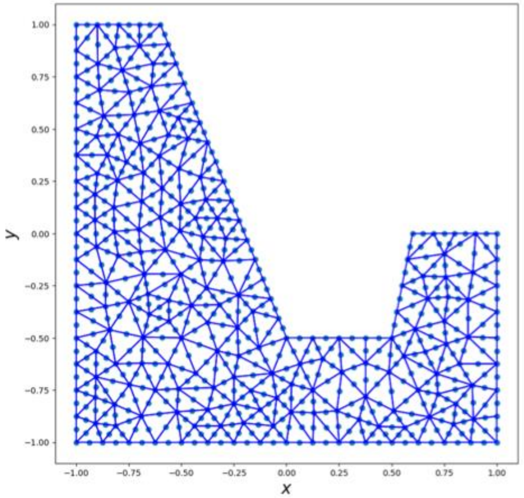
Can we handle general smooth domain?



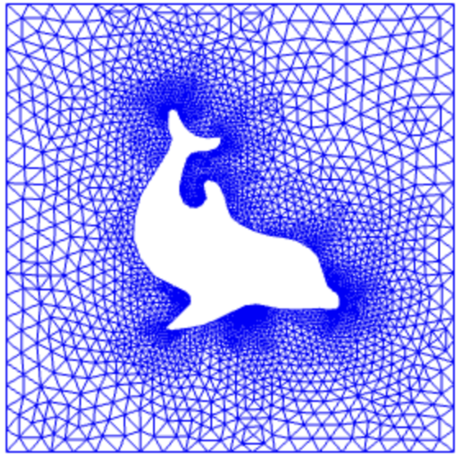
(a) Circle



(b) Square with a hole



(c) Polygon



(d) Dolphin

Hong et al., SIAM Journal on Scientific Computing (2024)

Finite Element Operator Network

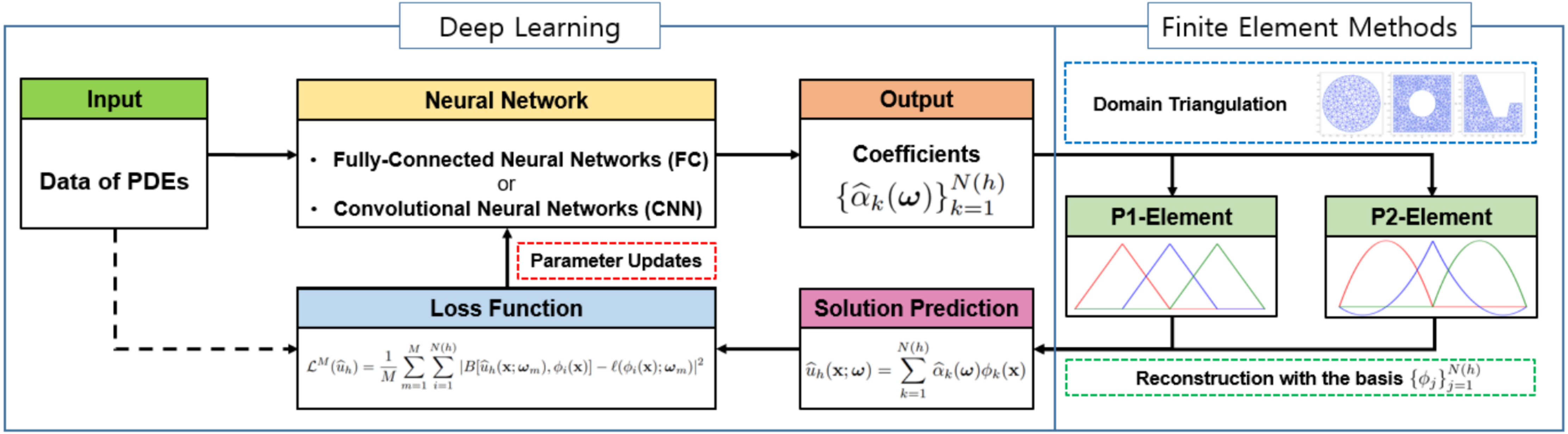
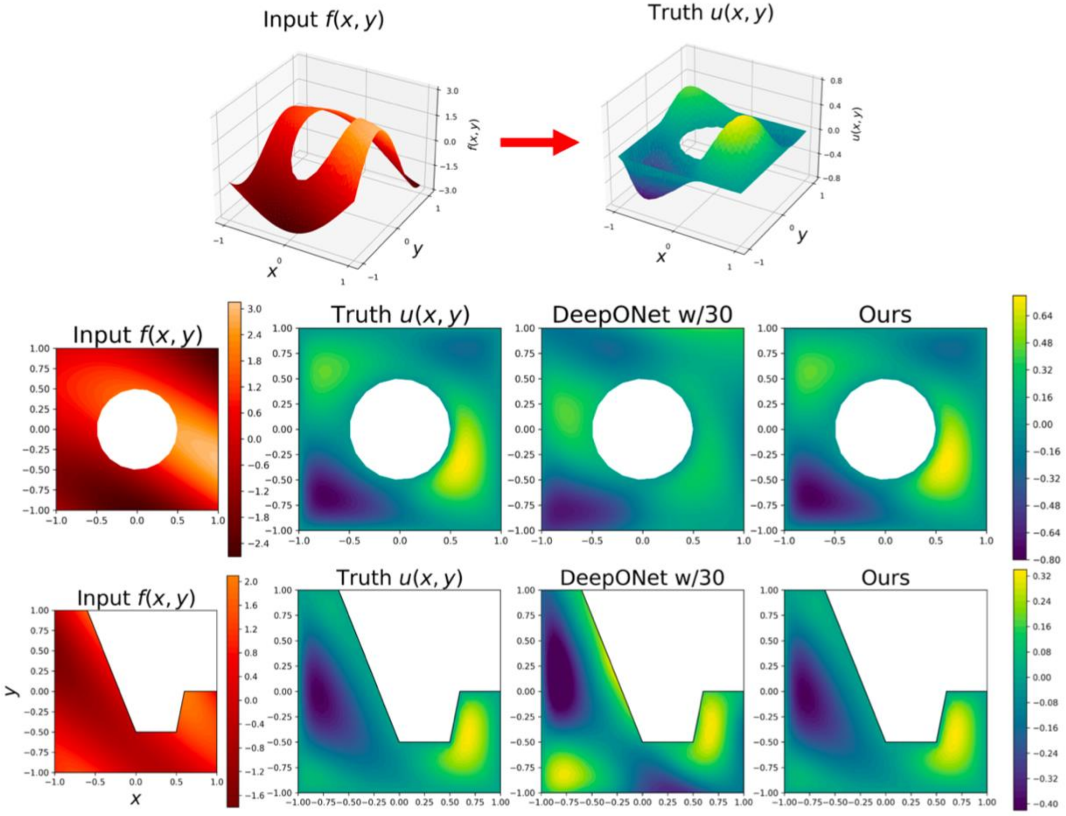
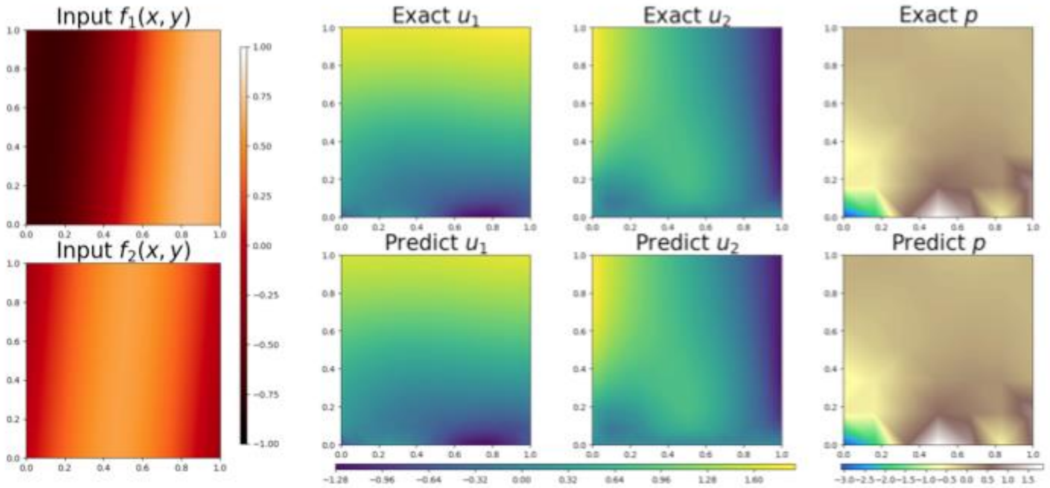


Figure: Schematic diagram of FEONet

Numerical Experiments





$$
\begin{aligned}
-\nabla \cdot (\nabla \boldsymbol{u} - pI) &= \boldsymbol{f}, && (x, y) \in D = [0,1]^2, \\
\nabla \cdot \boldsymbol{u} &= 0, && (x, y) \in D, \\
\boldsymbol{u} &= \boldsymbol{g}, && (x, y) \in \Gamma_D = \{(x, y) \in D | y = 0\}, \\
(\nabla \boldsymbol{u} - pI) \cdot \boldsymbol{n} &= (0, 0), && (x, y) \in \Gamma_N = \{(x, y) \in D | x = 0, 1 \text{ or } y = 1\}
\end{aligned}
$$

Numerical Experiments

Model (#Train data)	Domain I	Domain II	Domain III	BC I	BC II	Eq I	Eq II
DON (supervised, w/30)	27.15$\pm$1.16	51.21$\pm$3.58	53.92$\pm$4.59	21.75$\pm$1.19	22.75$\pm$1.05	24.38$\pm$1.37	10.26$\pm$0.14
DON (supervised, w/300)	2.10$\pm$0.75	5.62$\pm$0.37	6.22$\pm$0.96	0.68$\pm$0.11	0.96$\pm$0.06	0.76$\pm$0.10	**0.20**$\pm$0.09
DON (supervised, w/3000)	**0.69**$\pm$0.17	4.75$\pm$0.75	6.20$\pm$1.00	0.53$\pm$0.36	0.33$\pm$0.09	0.33$\pm$0.27	0.24$\pm$0.13
PIDeepONet (w/o labeled data)	9.80$\pm$9.41	101.03$\pm$167.46	32.89$\pm$6.34	1.51$\pm$0.46	1.43$\pm$0.45	19.41$\pm$11.30	2.66$\pm$0.71
Ours (w/o labeled data)	1.24$\pm$0.00	**1.76**$\pm$0.03	**0.51**$\pm$0.00	**0.13**$\pm$0.01	**0.32**$\pm$0.03	**0.13**$\pm$0.01	0.54$\pm$0.07

- **Domain I** (circle), **Domain II** (square with a hole), **Domain III** (polygon) with the equation

$$\begin{aligned} -\varepsilon\Delta u + \mathbf{v}\cdot\nabla u &= f(x,y), \quad && (x,y)\in D, \\ u(x,y) &= 0, && (x,y)\in\partial D. \end{aligned}$$

- **BC I** (Dirichlet), **BC II** (Neumann) with the equation

$$\begin{aligned} -\varepsilon u_{xx} + b u_x &= f(x), \quad && x\in D, \\ u(x) = 0 \quad \text{or} \quad u_x(x) &= 0, && x\in\partial D, \end{aligned}$$

- **Eq I** : general second-order linear equation, **Eq II** : Burgers' equation (nonlinear).

Limitation and Advantages for Sci. ML.

- Limitation:
  - Not reliable yet (error analysis not complete)
  - Low accuracy
  - Training time is long

- Advantages:
  - Overcome curse of dimensionality
  - Fast inference
  - Inverse problems
  - Potential to tackle complex problems

Limitation and Advantages for Sci. ML.

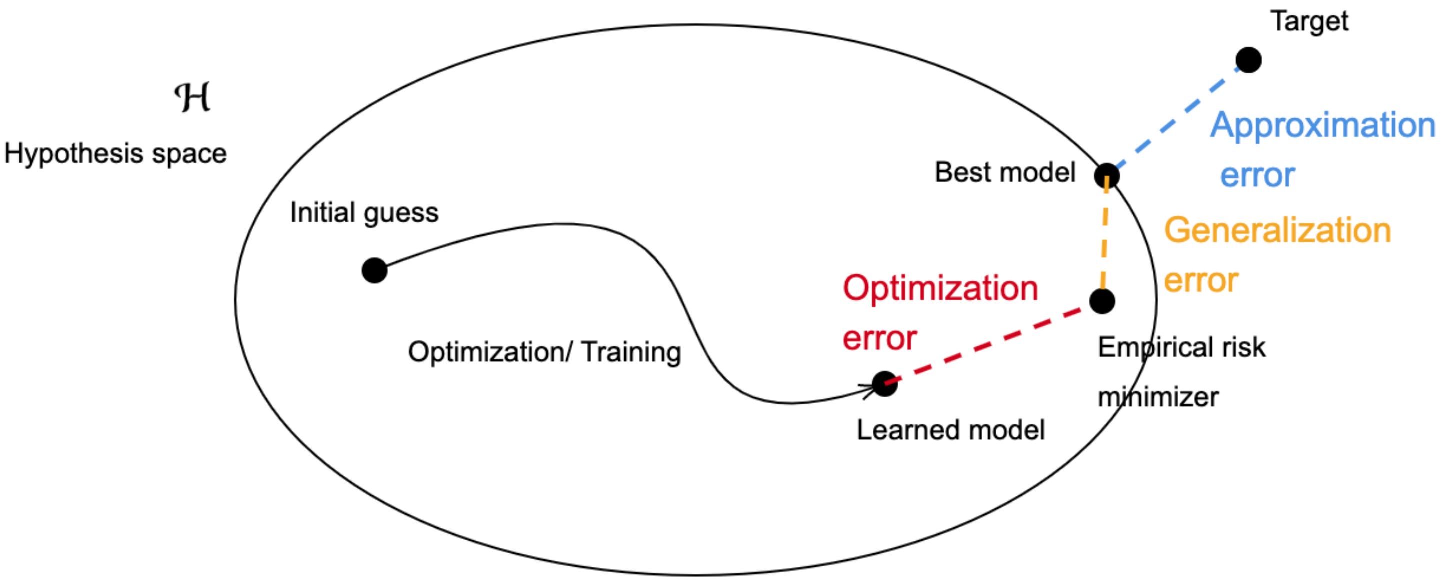
- Limitation:
  - Not reliable yet (error analysis not complete)
  - Low accuracy
  - Training time is long

- Advantages:
  - Overcome curse of dimensionality
  - Fast inference
  - Inverse problems
  - Potential to tackle complex problems

Convergence of Neural Networks

The convergence of neural networks can be decomposed by

$$u - u_{n,M,N} = \underbrace{(u - u_n)}_{\textit{approx. error}} + \underbrace{\left(u_n - u_{n,M}\right)}_{\textit{gen. error}} + \underbrace{\left(u_{n,M} - u_{n,M,N}\right)}_{\textit{opt. error}}$$



Convergence Analysis of FEONet

- **Constructing hypothesis space** · **(Approximation error)**

  : the error of approximating the solution of a PDE using neural networks;

$$\inf_{\hat{f}\in V_n} \|\hat{f} - f\| \lesssim n^{-\alpha}\|f\| \quad \Rightarrow \quad V_n = \begin{cases} \text{piecewise polynomials} & \text{(Bramble–Hilbert lemma);} \\ \text{neural networks} & \text{(universal approximation theorem).} \end{cases}$$

- **Formulating the loss function** · **(Generalization error)**

  : the error of the neural network-based approximate solution on predicting unseen data;

$$\begin{cases} \mathcal{L} \quad \approx \quad \mathcal{L}^M \ \text{(Rademacher complexity)} \\ \qquad\qquad \text{stability of PDE} \end{cases} \Rightarrow \quad \arg\min \mathcal{L} \quad \approx \quad \arg\min \mathcal{L}^M$$

- **Finding the solution** · **(Optimization error)**

  : the error incurred by the optimization algorithm used in the training of neural networks for PDEs.

Convergence analysis of FEONet

Theorem (FEM Theory)

P-ℓ element method: $\|u - u_h\|_{L^2(D)} \leq Ch^l$

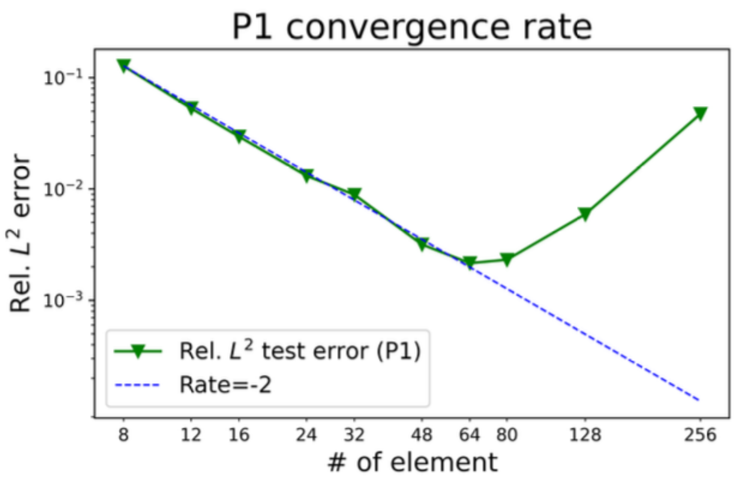
Theorem (Ko, Lee, and H.)

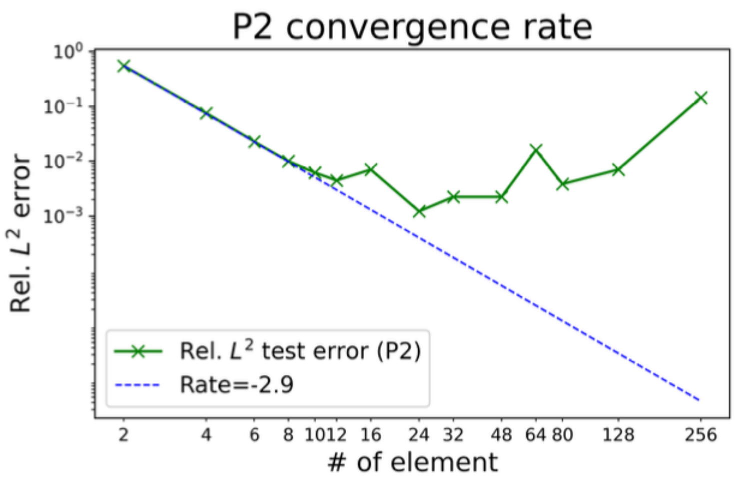
If we let u_h be the finite element approximation of the true solution u and $\widehat{u}_{h,n,M}$ be the approximate solution computed by the FEONet, then we have

$$\mathbb{E}_{\boldsymbol{\omega}}\left[\|u_h - \widehat{u}_{h,n,M}\|^2_{L^2(D)}\right] \to 0 \quad \text{as } n,\ M \to \infty.$$

Convergence analysis of FEONet

But something weird happens...





Does this come from ML things? Otherwise, should we dive into details?

- Why does the phenomenon of errors increasing again at a certain point occur?
- The objective is to investigate the underlying principle of FEONet based on the mathematical analysis.

Convergence analysis of FEONet

Approximation error (Céa's lemma for the neural network approximation)

Let $\kappa(A)$ be a condition number of the given finite element matrix A. Then we have

$$\|\alpha^* - \widehat{\alpha}_n^{\mathcal{L}}\|_{L^1(\Omega)} \leq \kappa(A) \inf_{\alpha \in \mathcal{N}_n} \|\alpha - \alpha^*\|_{L^1(\Omega)}.$$

Generalization error

Let $\kappa(A)$ be a condition number of the given finite element matrix A. Then we have

$$\mathbb{E}\left[\|\widehat{\alpha}_n^{\mathcal{L}} - \widehat{\alpha}_{n,M}^{\mathcal{L}}\|_{L^1(\Omega)}\right] \lesssim \kappa(A)^{d/2}\mathcal{R}_M(\mathcal{F}_n^{\mathcal{L}}) + \kappa(A) \inf_{\alpha \in \mathcal{N}_n} \|\alpha - \alpha^*\|_{L^1(\Omega)}.$$

Error estimate for the FEONet using Pℓ-element:

$$\mathbb{E}\left[\|u - u_{h,n,M}\|_{L^1(\Omega;L^2(D))}\right] \lesssim h^{\ell+1} + \kappa(A) \inf_{\alpha \in \mathcal{N}_n} \|\alpha - \alpha^*\|_{L^1(\Omega)} + \kappa(A)^{d/2}\mathcal{R}_M(\mathcal{F}_n^{\mathcal{L}})$$

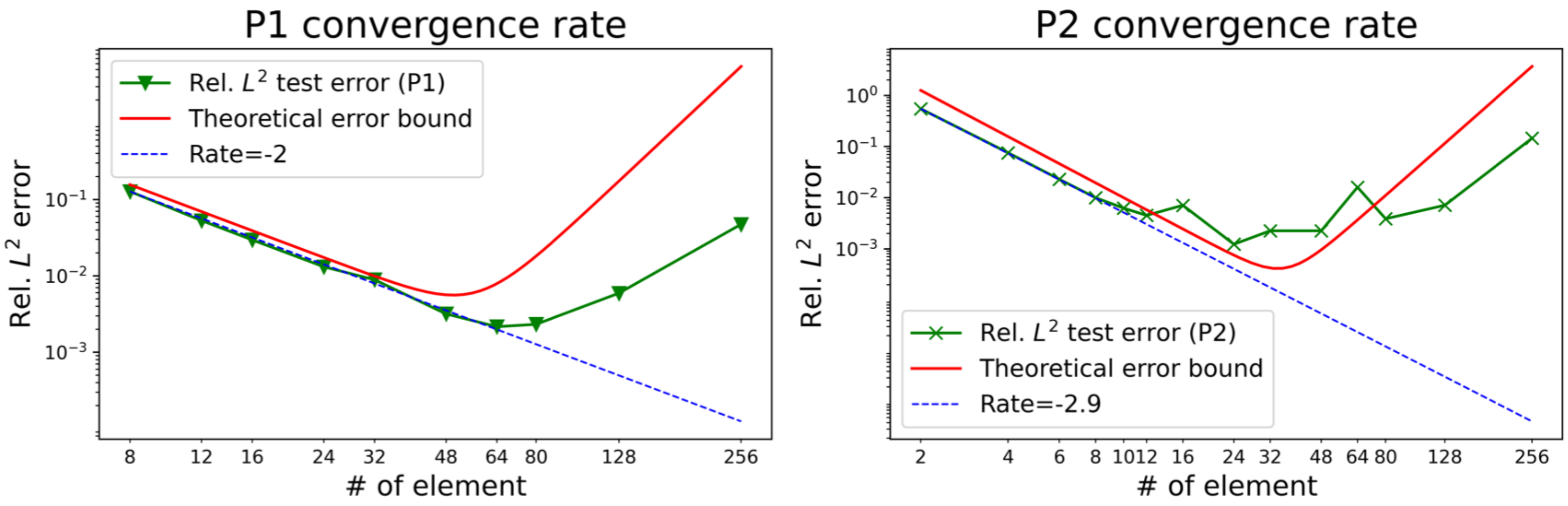
Hong et al., submitted

Convergence analysis of FEONet

Theorem (Error estimate for the FEONet)

If we use the (P^ℓ)-element, the predicted solution $u_{h,n,M}$ by the FEONet satisfies the following error estimate:

$$\mathbb{E}\left[\|u - u_{h,n,M}\|_{L^1(\Omega;L^2(D))}\right] \lesssim h^{\ell+1} + \frac{\kappa(A)^{1+d}}{\sqrt{n}} + \frac{\kappa(A)^{1+3d/2}}{\sqrt{M}}.$$



(Theory-guided strategy 1) Take $n \to \infty$ and $M \to \infty$.

(Theory-guided strategy 2) Use a preconditioned loss: replace $|A\alpha - F|$ by $|P^{-1}A\alpha - P^{-1}F|$

Hong et al., submitted

FEONet analysis

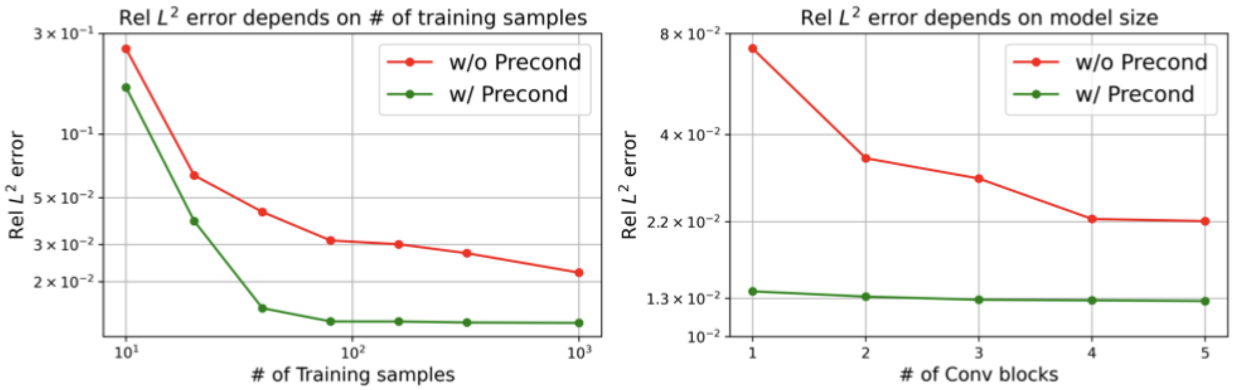


Figure: The relative L^2 errors resulting from varying the number of training samples and the model size.

Original FEONet vs Preconditioned FEONet using

$$\mathcal{K}(\alpha) = \|P^{-1}A\alpha(\omega) - P^{-1}F(\omega)\|_{L^1(\Omega)}$$

$$\mathcal{K}^M(\alpha) = \frac{|\Omega|}{M}\sum_{i=1}^{M}|P^{-1}A\alpha(\omega_i) - P^{-1}F(\omega_i)|,$$

$\implies \kappa(A)$ is replaced by $\kappa(P^{-1}A)$ where $\kappa(P^{-1}A) \ll \kappa(A)$.

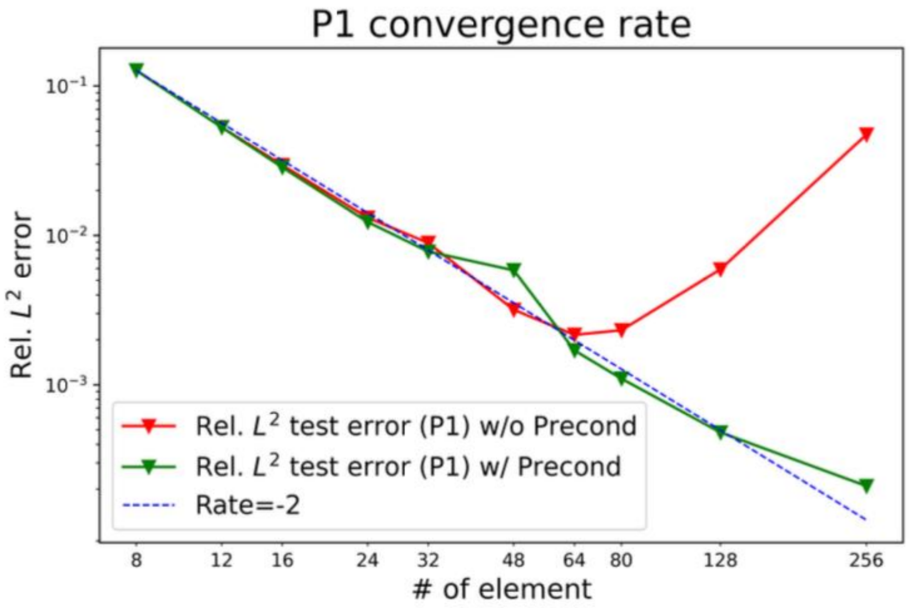
Convergence analysis of FEONet (Preconditioning)

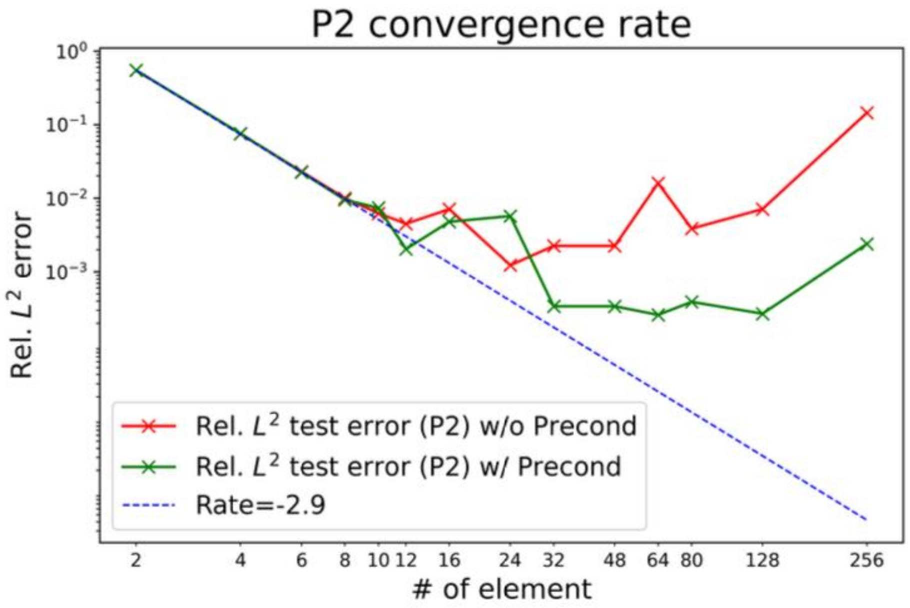
# of elements		8	12	16	24	32	48	64	80	128	256
CD Eq.	w/o precond	0.12731	0.05301	0.02958	0.01316	0.00890	**0.00318**	0.00216	0.00232	0.00593	0.04736
	w precond	**0.12728**	**0.05291**	**0.02887**	**0.01231**	**0.00780**	0.00584	**0.00170**	**0.00110**	**0.00048**	**0.00021**

TABLE 1 *Numerical errors against the number of P1 elements for convection-diffusion equation.*

# of elements		2	4	6	8	10	12	16	24	32	48	64	80	128	256
CD Eq.	w/o precond	**0.54603**	**0.07546**	0.02294	0.01000	**0.00624**	0.00451	0.00709	**0.00123**	0.00225	0.00225	0.01603	0.00387	0.00711	0.14480
	w precond	0.54614	0.07576	**0.02253**	**0.00949**	0.00738	**0.00204**	**0.00479**	0.00573	**0.00034**	**0.00034**	**0.00026**	**0.00039**	**0.00027**	**0.00239**

TABLE 2 *Numerical errors against the number of P2 elements for convection-diffusion equation.*





Hong et al., submitted

Thank you!