SymPDE → Model

Incorporating PDE symmetries into equivariant models

MSc Thesis intermediate presentation

Elias Dubbeldam

Supervised by Alex Gabel

UNIVERSITY OF AMSTERDAM

slides at: github.com/elidub/SymPDE

Outline

- Motivation
- Literature overview
- My work
- Planning

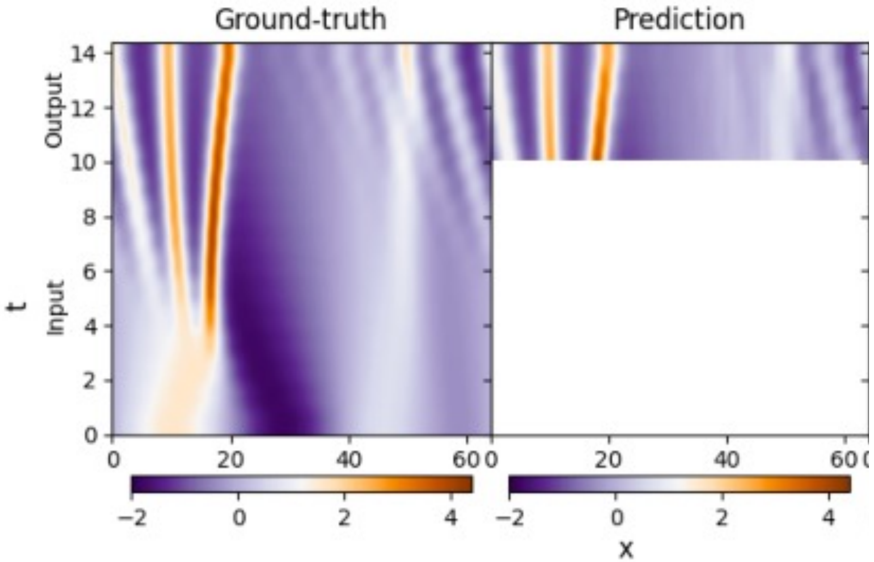
Motivation

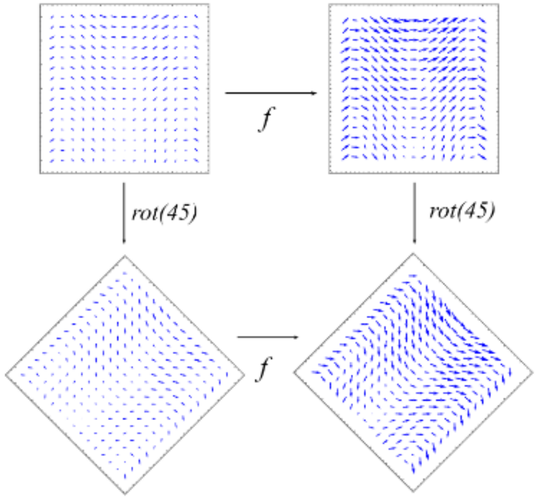
- Setting: PDE future prediction
- Neural PDE solvers can speed up classical solvers
- However:
  Neural PDE solvers are data hungry
- Therefore:
  Incorporate PDE symmetries into equivariant models: “SymPDE → Model”
- Symmetries of PDEs are described by the group *Lie point symmetries*:

$$\Delta(x,t,u) = u_t + uu_x + u_{xx} + u_{xxxx} = 0.$$

$$\Delta(\boldsymbol{x},\boldsymbol{u}) = 0 \Longrightarrow \Delta[g\cdot(\boldsymbol{x},\boldsymbol{u})] = 0, \quad \forall g \in G$$

$$g_1 = \partial_t,\ g_2 = \partial_x,\ g_3 = t\partial_x - \partial_u,\ g_4 = 3t\partial_t + x\partial_x - 2u\partial_u$$



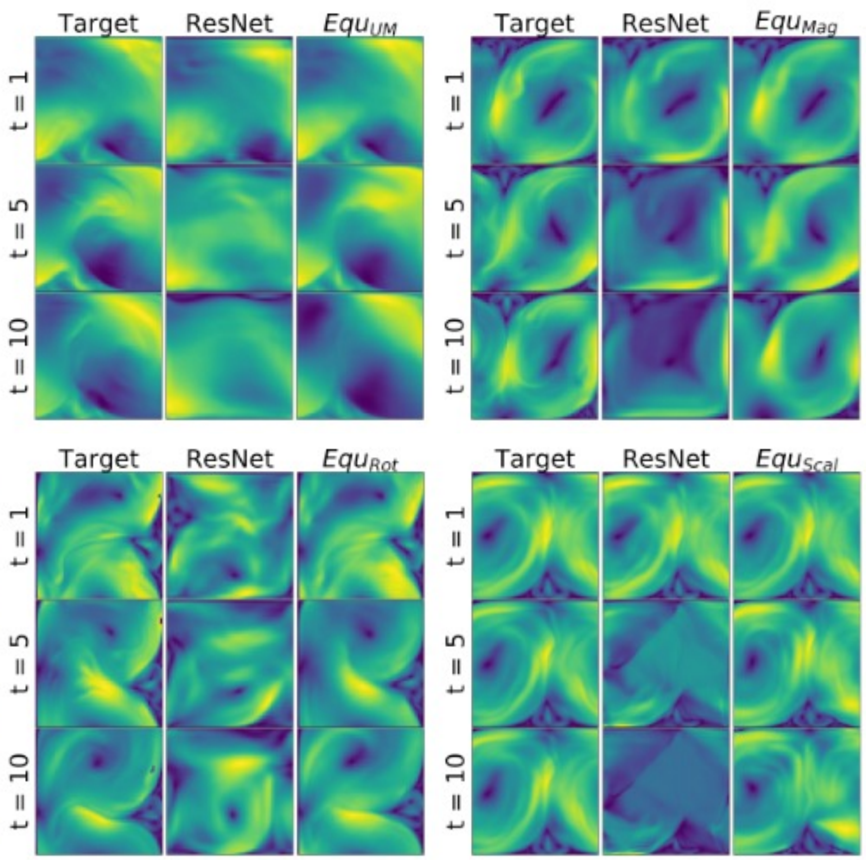


Single Symmetry Nets

Incorporating Symmetry into Deep Dynamics Models for Improved Generalization

ICLR21 by Wang, Yu & Walters (Univ of California)

- First work that incorporates Lie point symmetries (LPS).
- 'Only' able to incorporate a single symmetry (besides space and time translation) with `e2cnn`
- Implemented separate equivariant ResNet and U-Net.

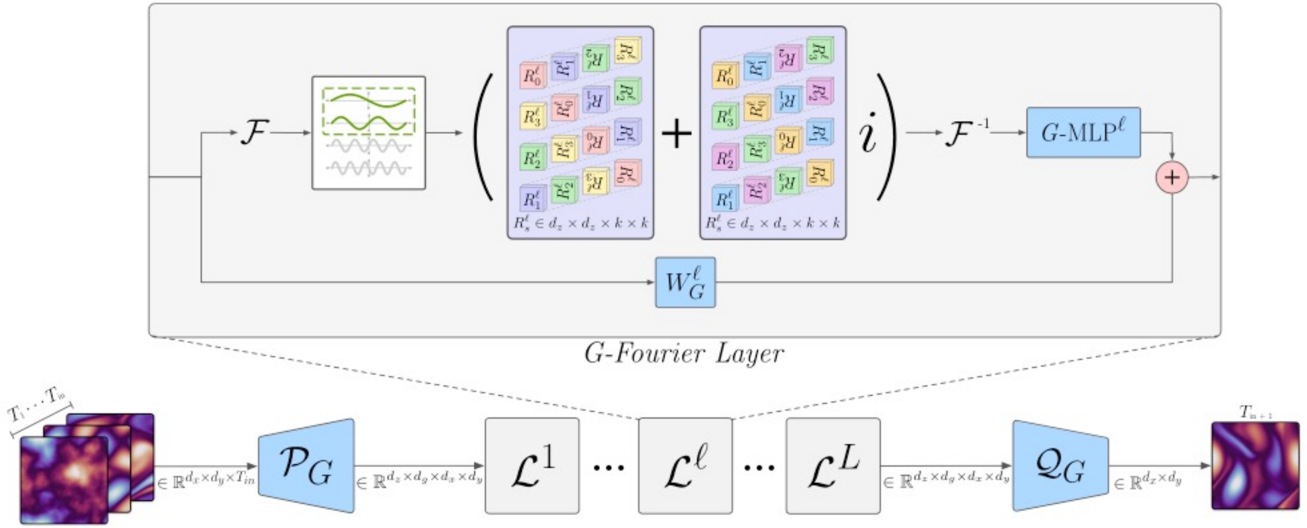


Literature Overview

Equivariant architectures

- **Single Symmetry Nets** *Incorporating Symmetry into Deep Dynamics Models for Improved Generalization*, ICLR21 by Wang, Yu & Walters (Univ of California)

- **G-FNO** *Group Equivariant Fourier Neural Operators for Partial Differential Equations*, PMLR23 by Helwig et al (Texas A&M University)
  Group CNNs for FNO. 'Only' for *p4m:* translations, 90° rotations and reflections



LPSDA

Lie Point Symmetry Data Augmentation for Neural PDE Solvers

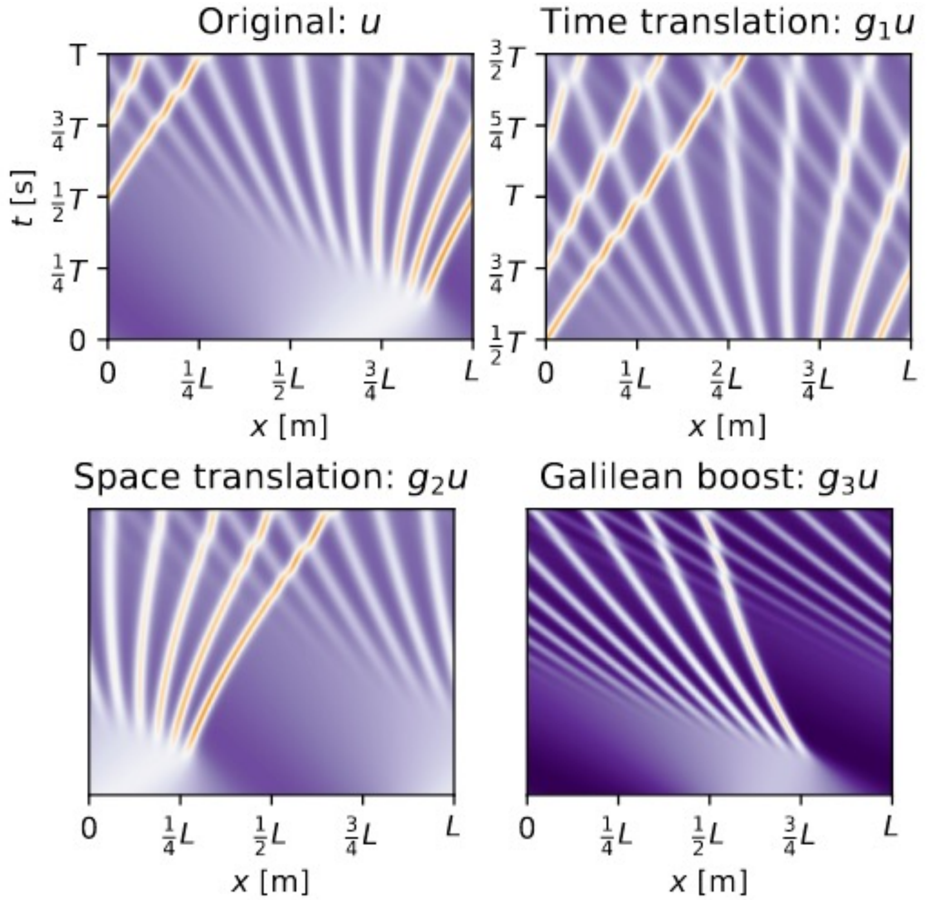
PMLR22 by Brandstetter, Welling & Worall (UvA)

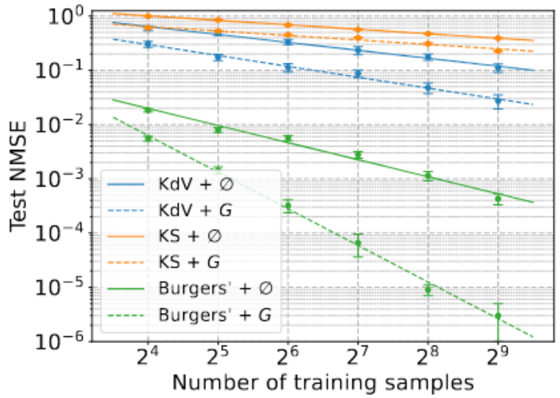
- Augment training data with continuous LPS:

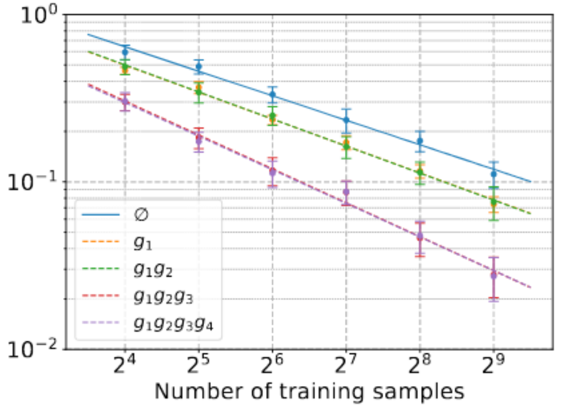
$$\mathbf{u}' = g_d(\epsilon_d) \cdots g_1(\epsilon_1)\mathbf{u}$$

- With g time and space translation, Galilean boosting and scaling;
  Typically, $\epsilon \sim \mathcal{U}(-0.5, 0.5)$

- Using FNO and ResNets







Literature Overview

Equivariant architectures

- **Single Symmetry Nets** *Incorporating Symmetry into Deep Dynamics Models for Improved Generalization*, ICLR21 by Wang, Yu & Walters (Univ of California)

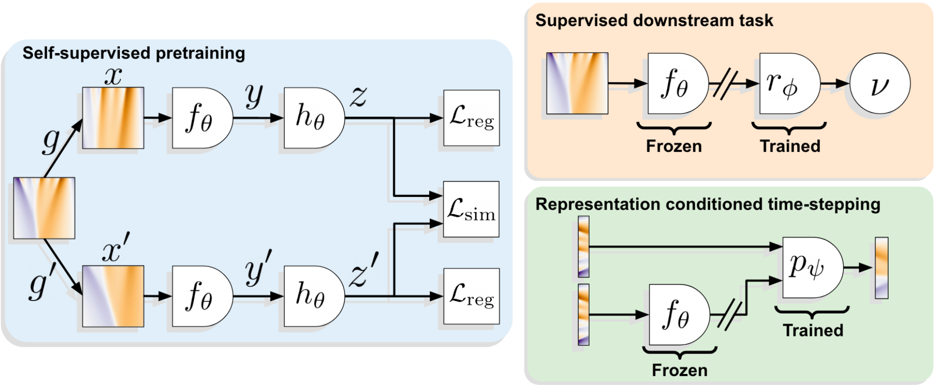
- **G-FNO** *Group Equivariant Fourier Neural Operators for Partial Differential Equations*, PMLR23 by Helwig et al (Texas A&M University)
  Group CNNs for FNO. 'Only' for *p4:* translations, 90° rotations and reflections

"External" equivariant models

- **LPSDA** *Lie Point Symmetry Data Augmentation for Neural PDE Solvers*, PMLR22 by Brandstetter, Welling & Worall (UvA)

- **LPS in PINN** *Lie Point Symmetry and Physics Informed Neural Networks,* ICML23 and NeurIPS23, by Akhound-Sadegh, Brandstetter, et all (McGill, Quebec, MSR)
  Trains a PINN with an additional symmetry loss term: $\mathcal{L}(\theta) = \alpha\mathcal{L}_{\text{PDE}} + \beta\mathcal{L}_{\text{data-fit}} + \gamma\mathcal{L}_{\text{sym}}$

- **SSL** *Self-Supervised Learning with Lie Symmetries for Partial Differential Equations*, ICLR23 by Mialon *et al* (incl. LeCun) (Meta AI - FAIR, LIGM, MIT)
  Similar approach a LPSDA, but trained on similarity between augmentations from same PDE using SSL: $\mathcal{L}(\boldsymbol{Z}, \boldsymbol{Z}') \approx \alpha\mathcal{L}_{\text{sim}}(\boldsymbol{Z}, \boldsymbol{Z}') + \beta(\mathcal{L}_{\text{reg}}(\boldsymbol{Z}) + \mathcal{L}_{\text{reg}}(\boldsymbol{Z}'))$



Literature Overview

Equivariant architectures

- **Single Symmetry Nets** *Incorporating Symmetry into Deep Dynamics Models for Improved Generalization*, ICLR21 by Wang, Yu & Walters (Univ of California)
- **G-FNO** *Group Equivariant Fourier Neural Operators for Partial Differential Equations*, PMLR23 by Helwig et al (Texas A&M University)
  Group CNNs for FNO. 'Only' for *p4:* translations, 90° rotations and reflections

"External" equivariant models

- **LPSDA** *Lie Point Symmetry Data Augmentation for Neural PDE Solvers*, PMLR22 by Brandstetter, Welling & Worall (UvA)
- **LPS in PINN** *Lie Point Symmetry and Physics Informed Neural Networks,* ICML23 and NeurIPS23, by Akhound-Sadegh, Brandstetter, *et al* (McGill, Quebec, MSR)
  Trains a PINN with an additional symmetry loss term: $\mathcal{L}(\theta) = \alpha\mathcal{L}_{\text{PDE}} + \beta\mathcal{L}_{\text{data-fit}} + \gamma\mathcal{L}_{\text{sym}}$
- **SSL** *Self-Supervised Learning with Lie Symmetries for Partial Differential Equations*, ICLR23 by Mialon *et al* (incl. LeCun) (Meta AI - FAIR, LIGM, MIT)
  Similar approach a LPSDA, but trained on similarity between augmentations from same PDE using SSL: $\mathcal{L}(\boldsymbol{Z}, \boldsymbol{Z}') \approx \alpha\mathcal{L}_{\text{sim}}(\boldsymbol{Z}, \boldsymbol{Z}') + \beta(\mathcal{L}_{\text{reg}}(\boldsymbol{Z}) + \mathcal{L}_{\text{reg}}(\boldsymbol{Z}'))$

Lie symmetry convolutions

- **LieConv** *Generalizing Convolutional Neural Networks for Equivariance to Lie Groups on Arbitrary Continuous Data*, PMLR20 by Finzi *et al* (NYU)
- **L-conv** *Automatic Symmetry Discovery with Lie Algebra Convolutional Network*, NeurIPS21 by Dehmamy, Walters, Yu, *et al* (Northwestern Univ)

Limitations of current SymPDE → Model

Data: Existing work analyses only a few PDEs, landscape is scattered.

- Models are hard to compare
- Contribution per symmetry is poorly understood

	Previous work	Heat	Burger’s	KdV	KS	NS	SWE	Other
Equivariant architectures	**Single Symmetry Nets**	2D						Ocean currents, Rayleigh–Bénard convection
	G-FNO					2D, 3D	2D, 3D	
“External” equivariant models	**LPSDA**		1D	1D	1D			
	LPS in PINNs	1D	1D					
	SSL		1D	1D	1D	2D		

Model

- Equivariant architectures don’t use all symmetries and only work discretized → Not all inductive biases are used
- “External’ equivariant models do not exploit symmetry inside model, therefore not guaranteed.

Data: PDE Suite

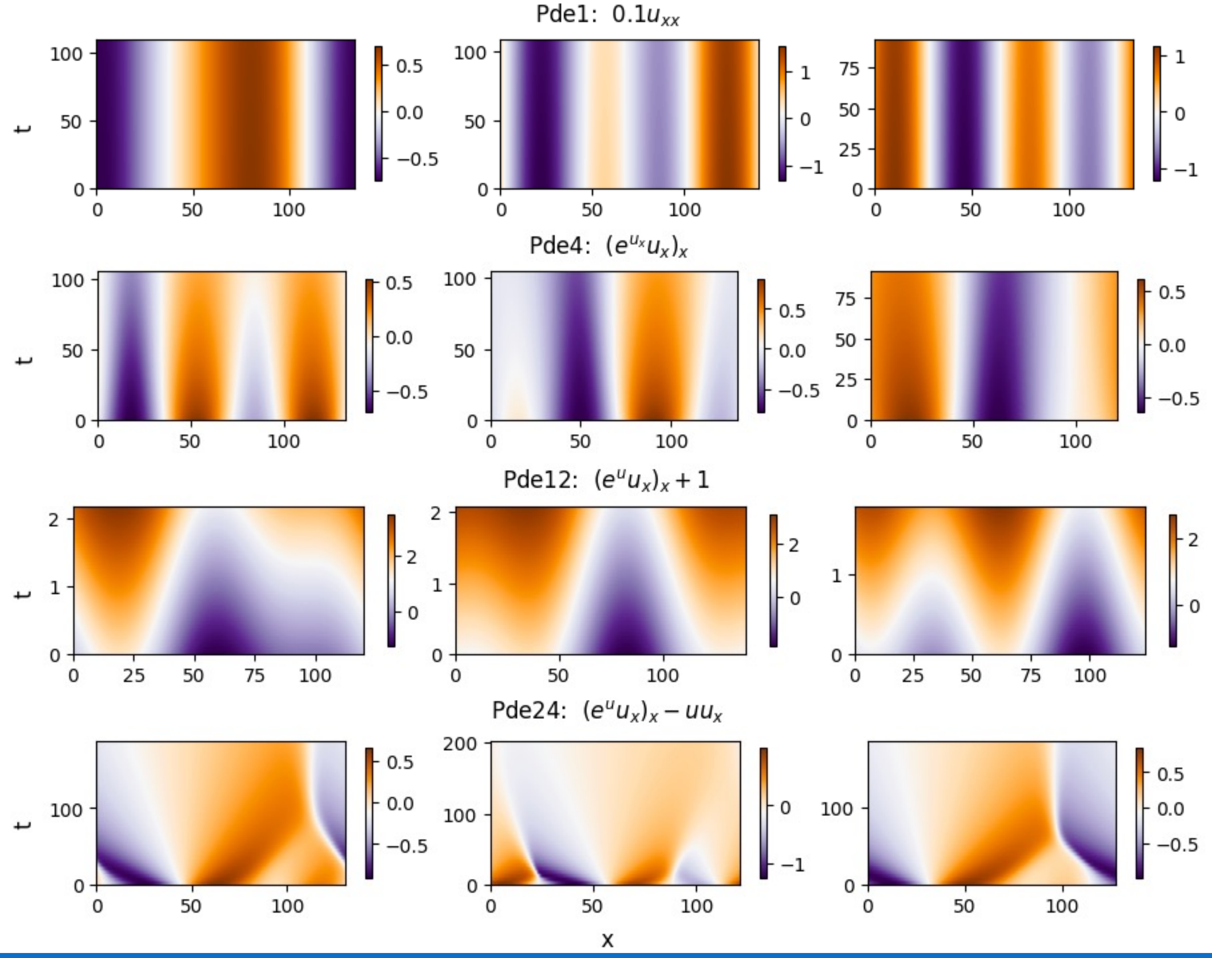
PDE Suite: a benchmark for SymPDE → Model

- As a start use, the PDEs from *Data-driven Lie Point Symmetry Detection for Continuous Dynamical Systems* (Gabel, unpublished).
- Most PDEs have much more symmetries than 'standard' PDEs

This solves the problem that current SymPDE → Model landscape is scattered, enabling to:

- ❑ Have a comparative overview of SymPDE → Model techniques
- ❑ Analyse and understand the contribution per type of symmetry

Index	PDE	Generators
1	$u_t = 0.1u_{xx}$	$X_1 = \partial_t,\ X_2 = \partial_x,\ X_3 = 2t\partial_t + x\partial_x,\ \ldots,\ X_\infty = b(t,x)\frac{\partial}{\partial u}$
	$\vdots$	
4	$u_t = (e^u u_x)_x$	$X_1 = \partial_t,\ X_2 = \partial_x,\ X_3 = 2t\partial_t + x\partial_x,\ \ldots,\ X_7 = \partial_x - t\partial_t + \partial_u$
	$\vdots$	
12	$u_t = (e^u u_x)_x + 1$	$X_1 = \partial_t,\ X_2 = \partial_x,\ X_3 = e^{-t}\frac{\partial}{\partial t} + e^{-t}\frac{\partial}{\partial u},\ X_4 = x\frac{\partial}{\partial x} + 2\frac{\partial}{\partial u}$
	$\vdots$	
24	$(e^u u_x)_x - uu_x$	$X_1 = \partial_t,\ X_2 = \partial_x,\ X_3 = t\frac{\partial}{\partial t} + (x+t)\frac{\partial}{\partial x} + \frac{\partial}{\partial u}$

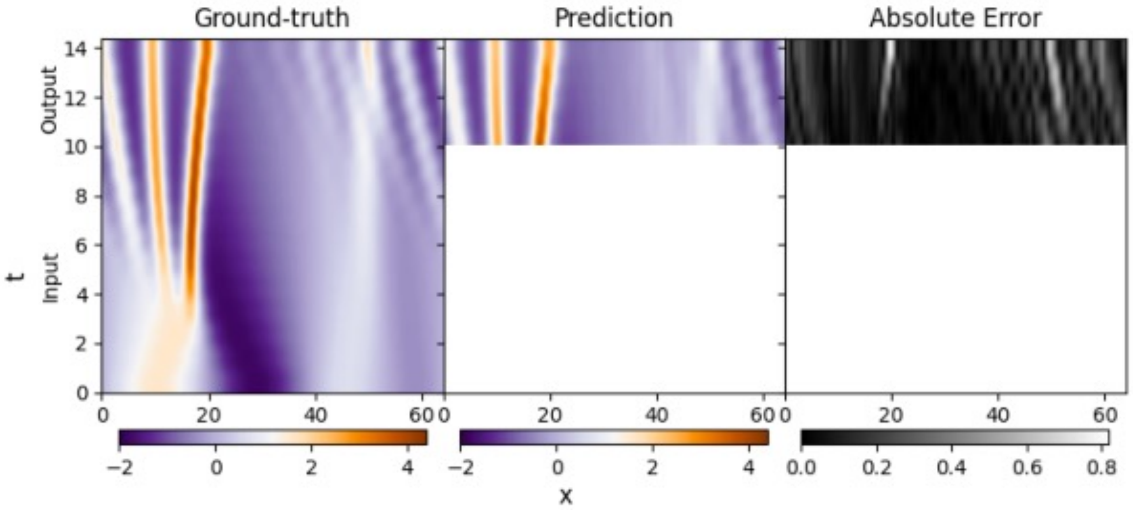


Experimenting baseline models

Task

- PDE future prediction on PDE Suite
- $(N_{train}, N_{val}, N_{test})=(500,100,100)$

Task example (KdV)

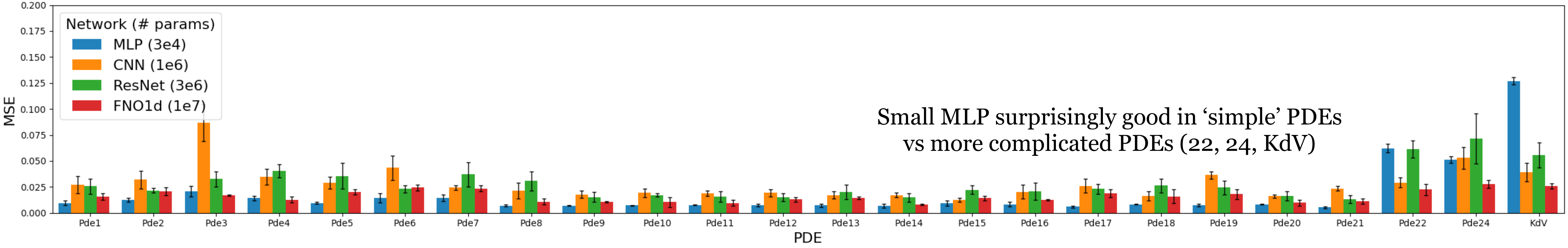


Next steps

→ Apply other SymPDE → Model techniques (such as LPSDA, LPS in PINNs, G-FNO) to PDE Suite

✓ Have a comparative overview of SymPDE → Model techniques

Performance across models and PDEs



Analysing symmetries

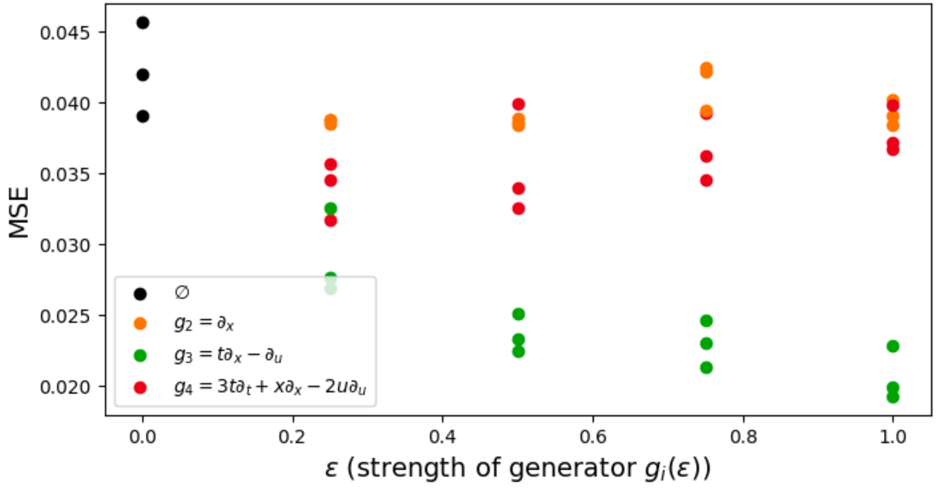
Reproducing LPSDA

Next steps

→ Apply to PDE Suite with all generators

✓ Analyse and understand the contribution per type of symmetry

Best generator strength for KdV with LPSDA



→

Contribution per symmetry

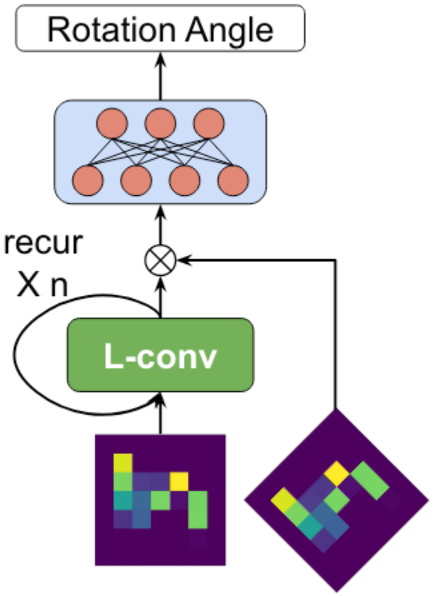
Some plot which: 'groups' the symmetries in PDE suite and analyses the contribution per symmetry

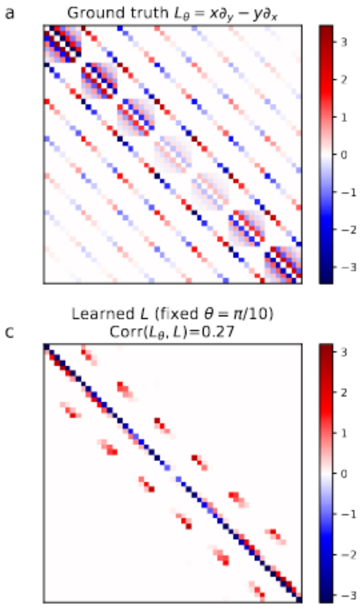
Model

Equivariant kernel method?

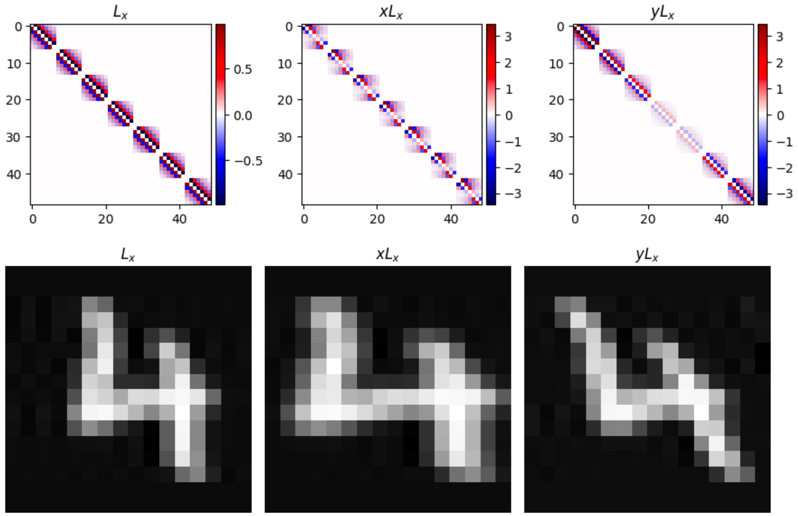
- To incorporate all symmetries continuously inside the model architecture.

L-Conv Dehmamy et al (2021)

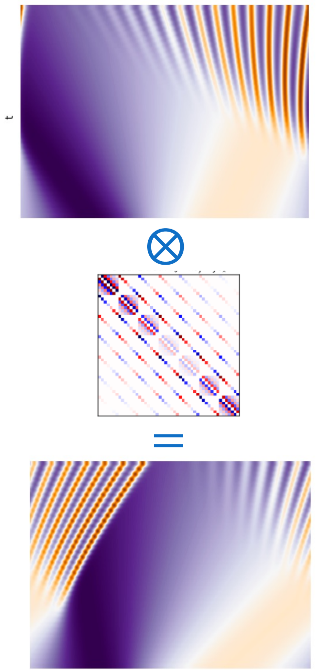




Transformations described by generators



Generalize to PDEs

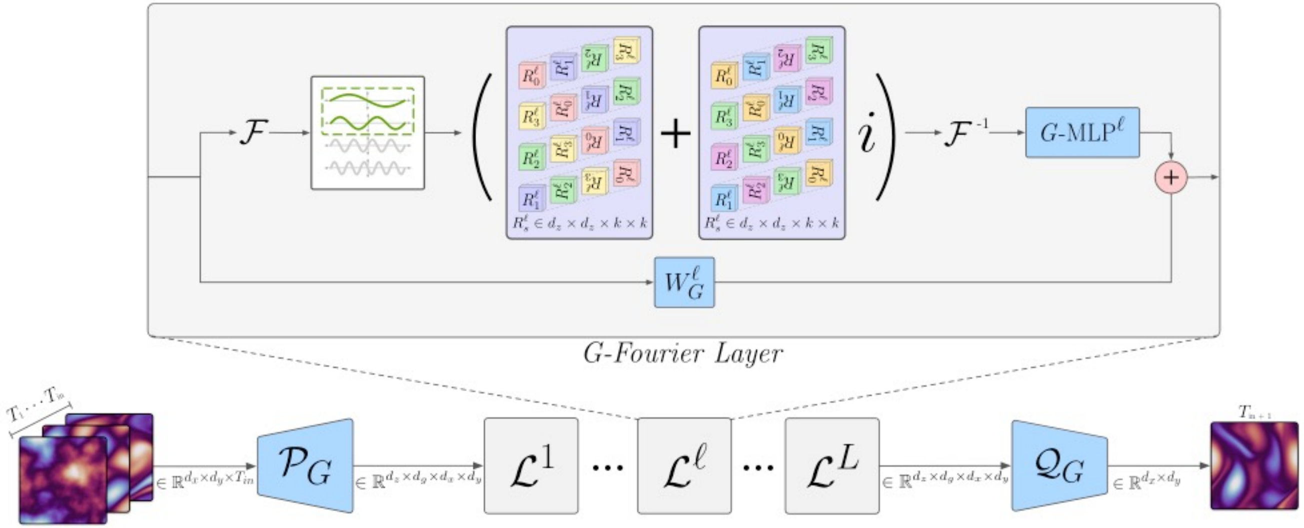


Planning

✓	Nov	Literature review, data generation
➢	Nov-Dec	Implementing literature
❑	Jan-March	Researching and experimenting for new method
❑	April	Implementing new method
❑	May-June	Finalizing and writing thesis

Backup slides

G-FNO



	NS	
	# PAR. (M)	TEST (%)
FNO	0.93	8.41(0.41)
FNO+$p4$	0.93	10.44(0.47)
FNO+$p4m$	0.93	22.09(1.46)
G-FNO-$p4$	0.85	**4.78**(0.39)
G-FNO-$p4m$	0.84	<u>6.19</u>(0.61)
U-NET-$p4$	3.65	18.40(0.44)
RADIALFNO-$p4$	1.03	9.21(0.26)
RADIALFNO-$p4m$	0.95	10.86(0.18)

LPS in PINNs

- Incorporate symmetry loss term using MLPs in PINN loss:

$$\mathcal{L}(\theta) = \alpha\mathcal{L}_{\text{PDE}} + \beta\mathcal{L}_{\text{data-fit}} + \gamma\mathcal{L}_{\text{sym}}$$

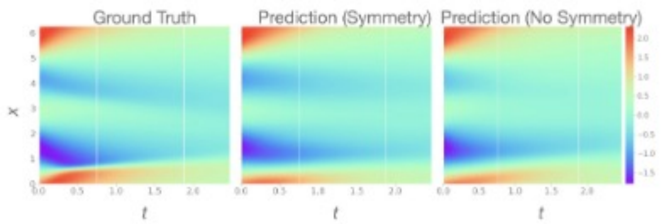
- $\mathcal{L}_{\text{PDE}}$ learns solution solving the PDE; $\mathcal{L}_{\text{data-fit}}$ learns IC and BC; $\mathcal{L}_{\text{sym}}$ learns LPS.
- $\mathcal{L}_{\text{sym}} = \sum_{k=1}^{K} J_\Delta^\top \operatorname{coef}\left(\mathbf{pr}^{(n)}\, \mathbf{v}_k\right)$
  - The orthogonality of the K prolonged vector fields and the gradient vector.
  - Equivalant to $\mathrm{pr}^{(n)}\,\mathbf{v}[\Delta] = 0$ when $\Delta = 0$. That is: asserting that applying a generator (e.g. $\mathbf{v} = 2\nu t\partial_x - xu\partial_u$) to the PDE (e.g. $\Delta = u_t - \nu u_{xx}$) still solves the PDE.

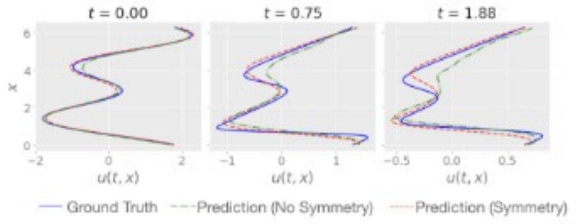
Table 1: The average test set mean-squared error for the Heat equation.

Number of Points (N_r)	No Symmetry	Symmetry
500	1.12 ± 0.58	$\mathbf{0.30 \pm 0.15}$
2000	0.36 ± 0.19	$\mathbf{0.24 \pm 0.14}$
10000	0.22 ± 0.14	$\mathbf{0.21 \pm 0.13}$

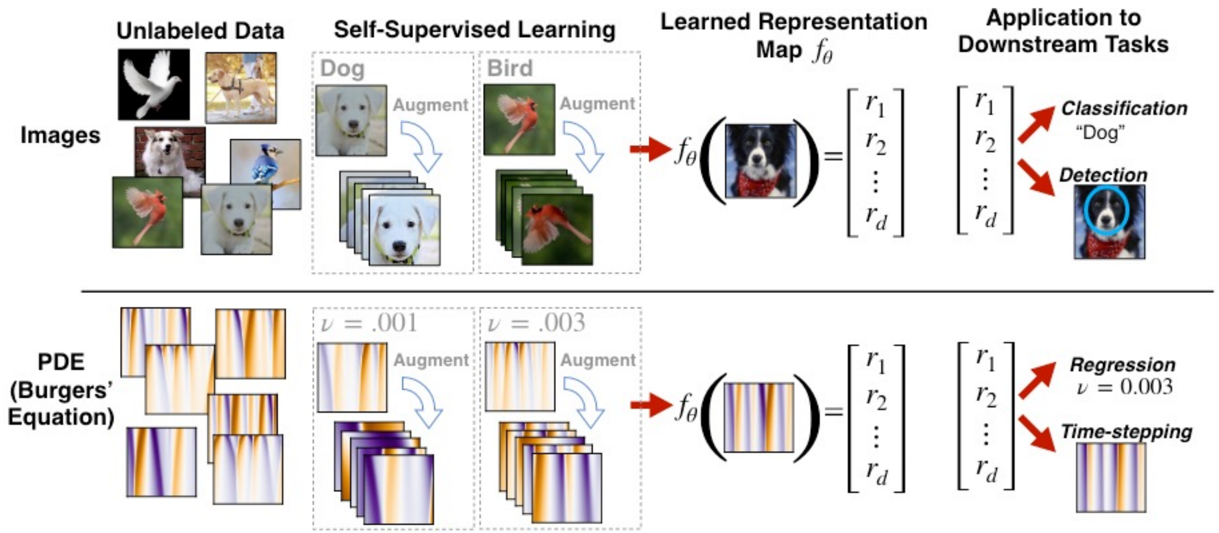
Table 2: The average test set mean-squared error for Burgers' equation.

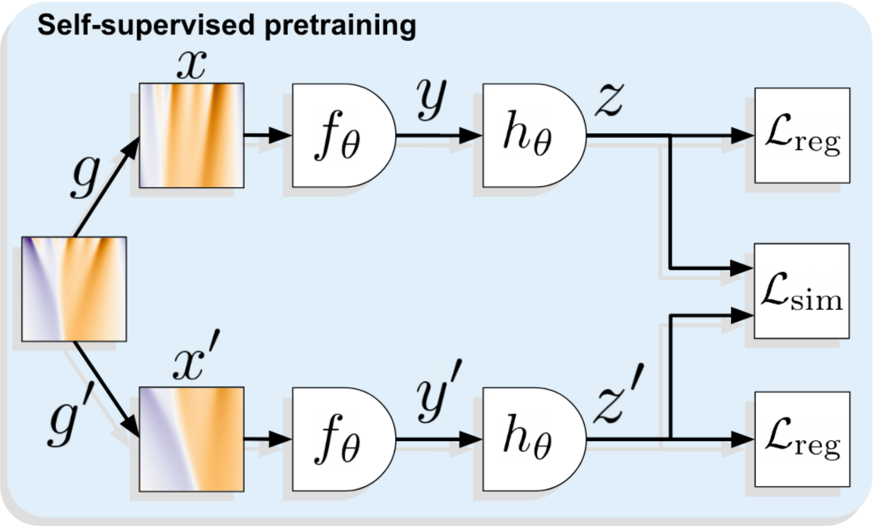
Number of Points (N_r)	No Symmetry	Symmetry
5000	0.041 ± 0.042	$\mathbf{0.034 \pm 0.039}$
25000	0.030 ± 0.038	$\mathbf{0.017 \pm 0.020}$
100000	0.018 ± 0.022	$\mathbf{0.013 \pm 0.020}$

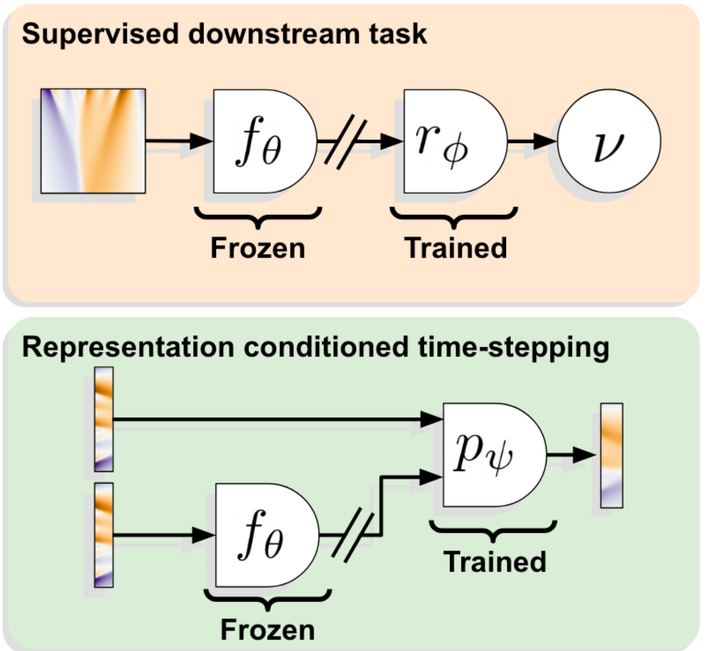




SSL


Unlabeled Data
Self-Supervised Learning
Learned Representation Map f_θ
Application to Downstream Tasks
Images
Dog
Bird
Augment
Augment
Classification
"Dog"
Detection
PDE (Burgers' Equation)
$\nu = .001$
$\nu = .003$
Augment
Augment
Regression
$\nu = 0.003$
Time-stepping


Self-supervised pretraining
x
y
z
g
f_θ
h_θ
$\mathcal{L}_{\mathrm{reg}}$
$\mathcal{L}_{\mathrm{sim}}$
x'
y'
z'
g'
f_θ
h_θ
$\mathcal{L}_{\mathrm{reg}}$


Supervised downstream task
f_θ
r_ϕ
ν
Frozen
Trained
Representation conditioned time-stepping
p_ψ
f_θ
Trained
Frozen