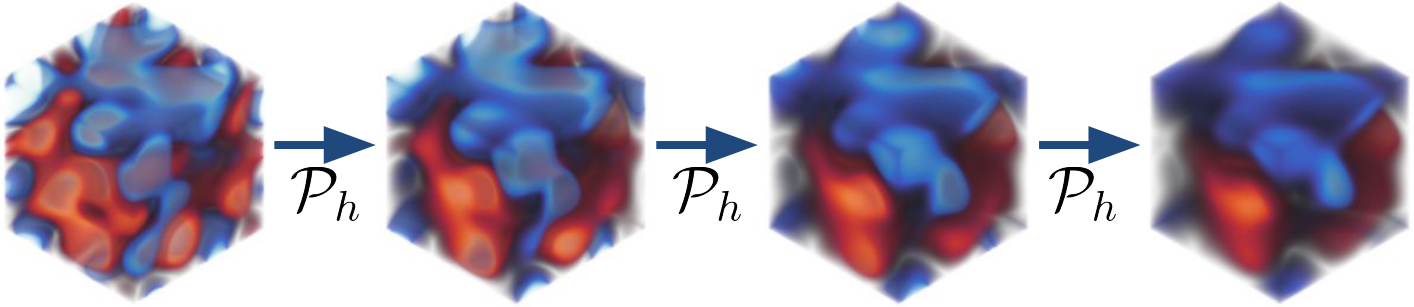


APEBench

A Benchmark for Autoregressive Neural Emulators of PDEs

Felix Koehler, Simon Niedermayr, Rüdiger Westermann, Nils Thuerey

NEURAL INFORMATION PROCESSING SYSTEMS

Munich Center for Machine Learning

Technical University of Munich

Kuramoto-Sivashinsky 2D

Kolmogorov Flow 2D

Gray-Scott 2D

Kuramoto-Sivashinsky 3D

Swift-Hohenberg 3D

Gray-Scott 3D

Simulating Spatio-Temporal Phenomena

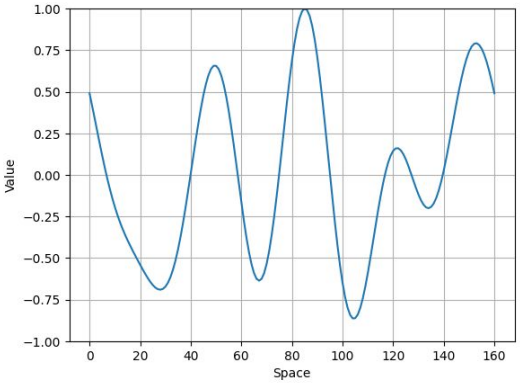
$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}$$

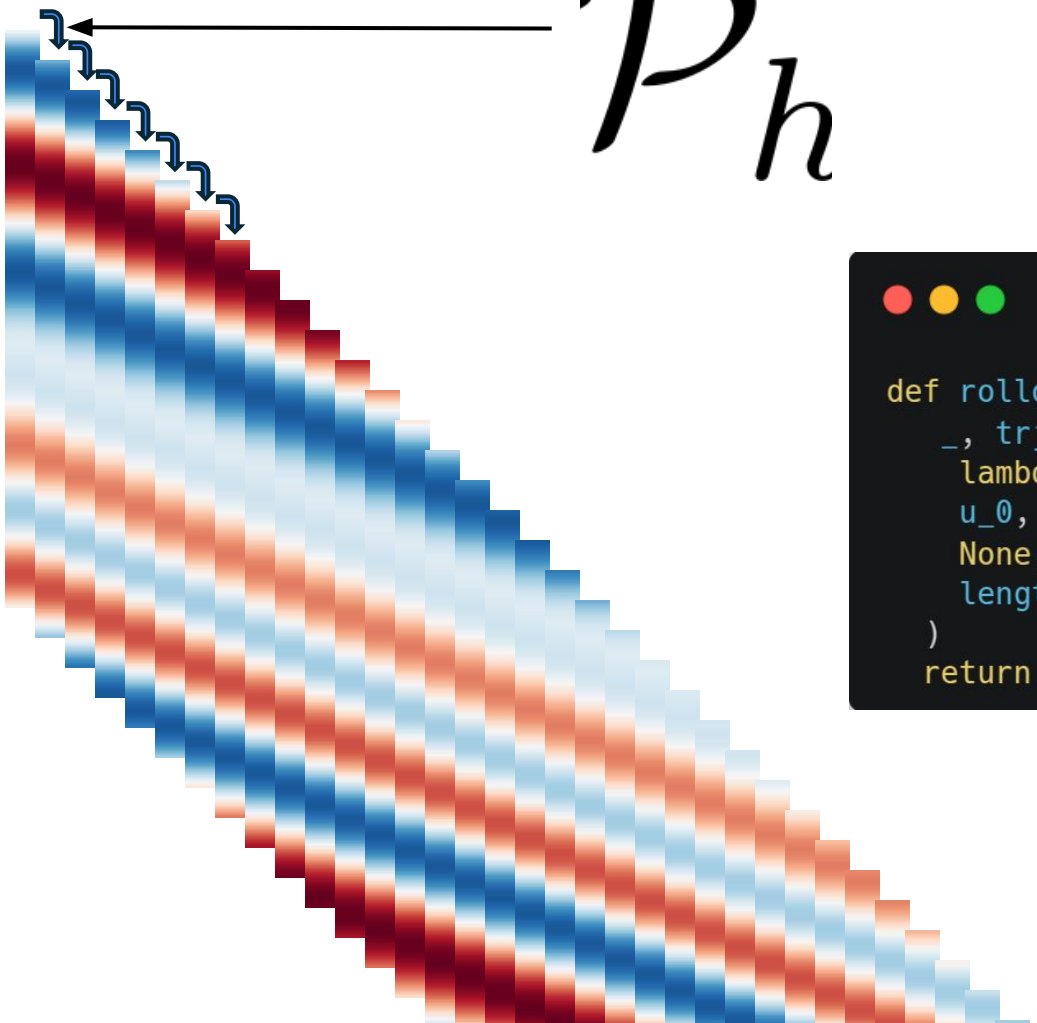
Symbolic Manipulation
+ Implementation

$$\mathcal{P}_h$$

$$u^{[0]}(x) = \sum_i a_i \sin\left(i\frac{2\pi}{L}x - \phi_i\right)$$

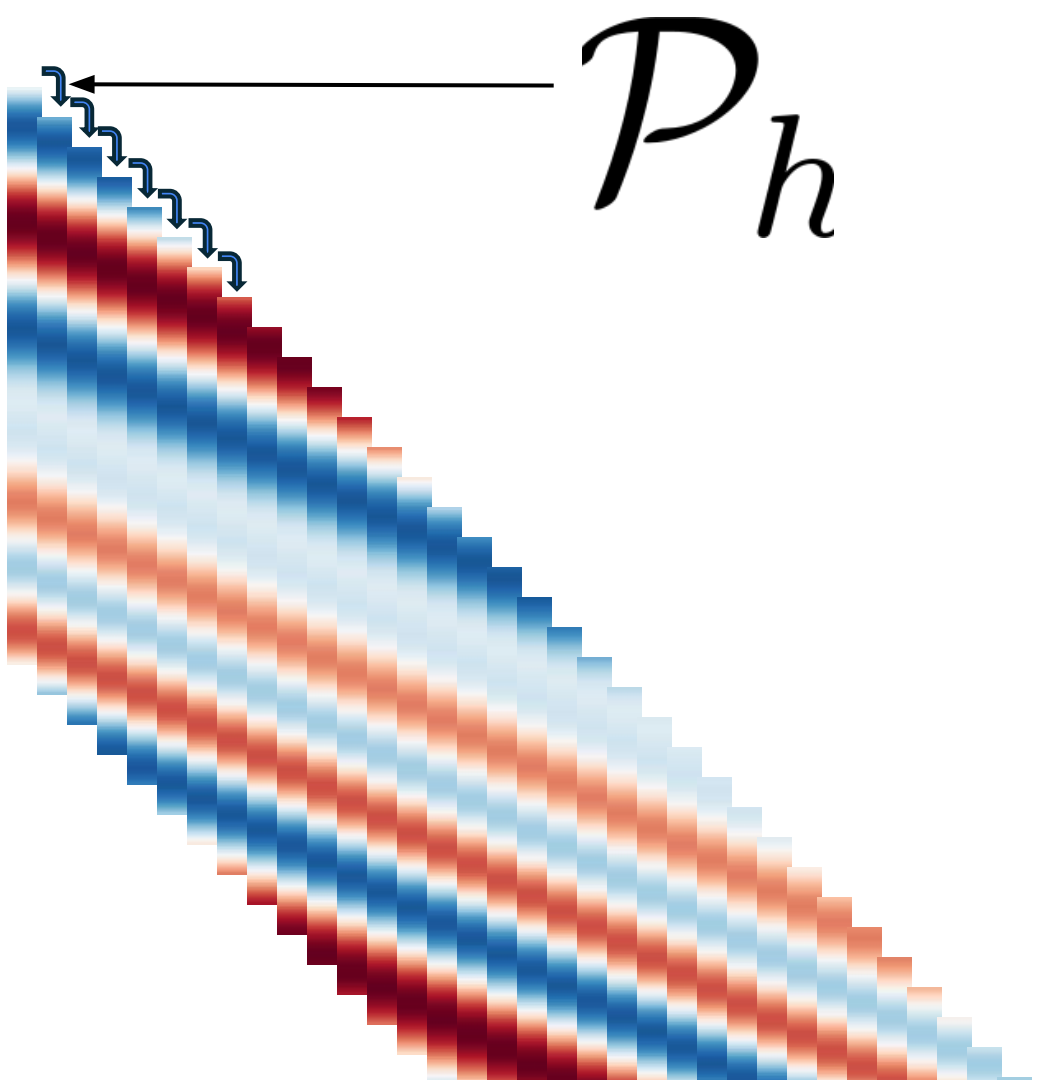
Discretization



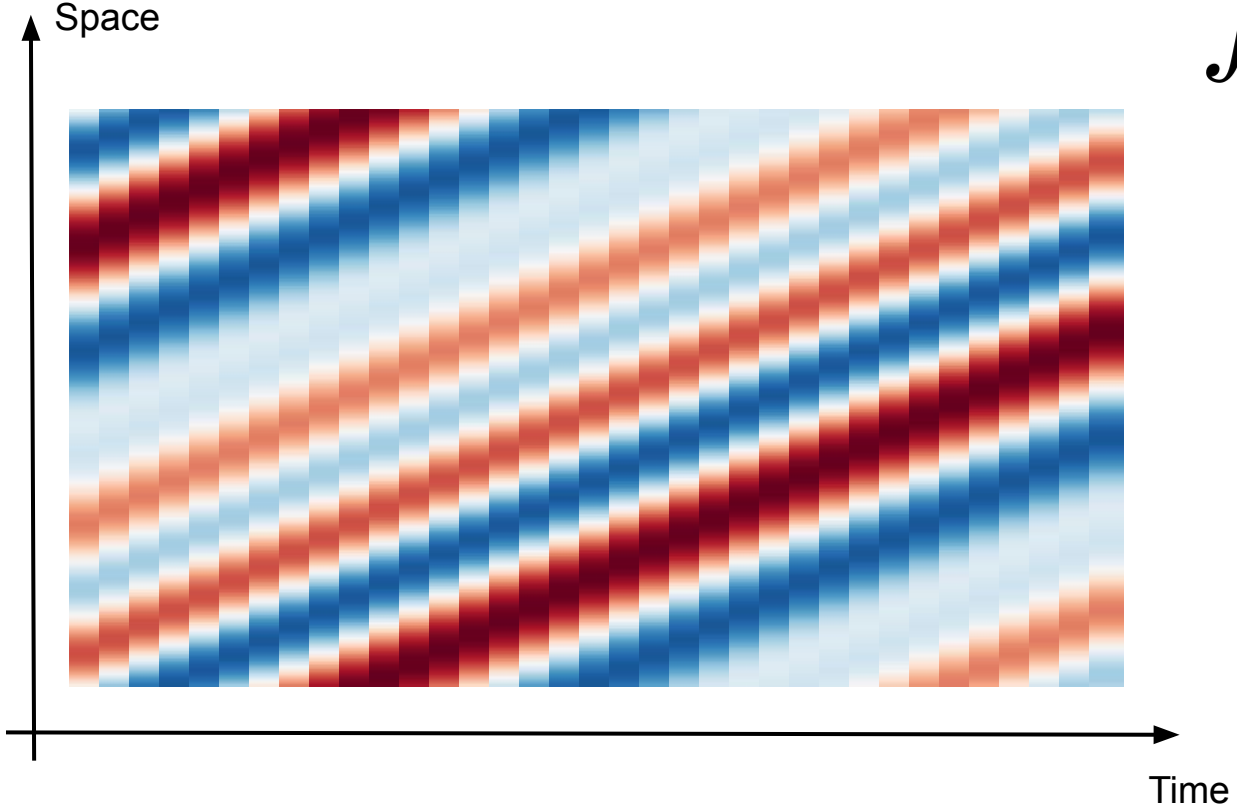


$\mathcal{P}_h$

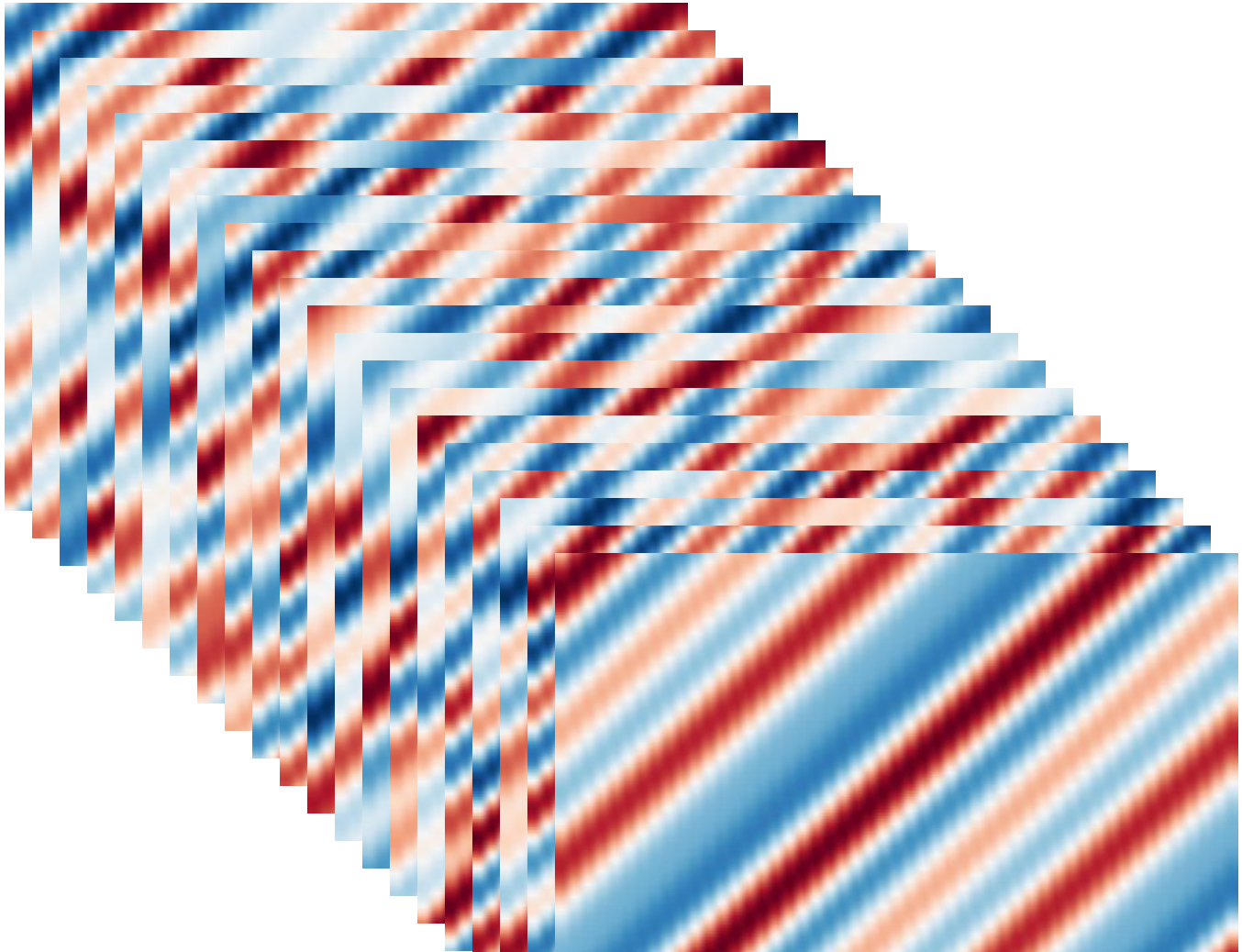
```
def rollout(u_0):
   _, trj = jax.lax.scan(
    lambda u, _: (stepper(u), stepper(u)),
    u_0,
    None,
    length=100,
  )
  return trj
```

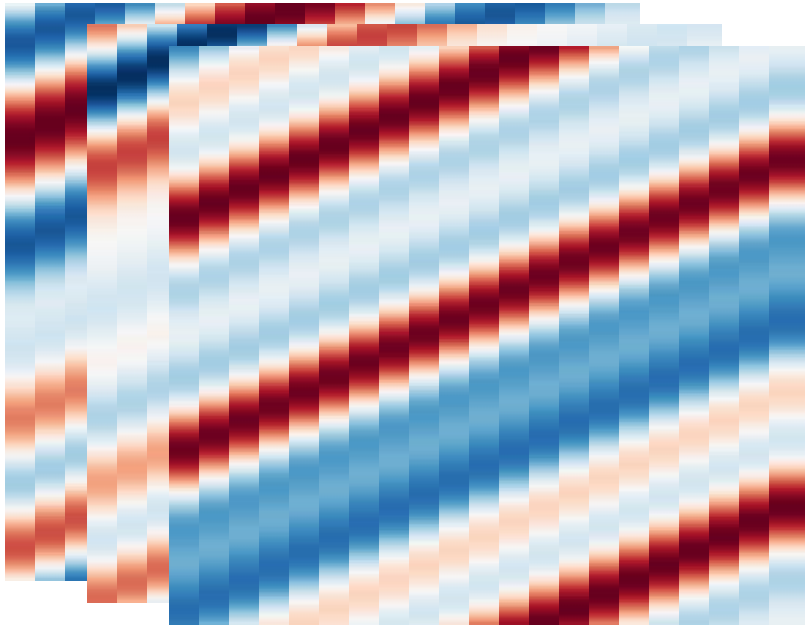

$\mathcal{P}_h$

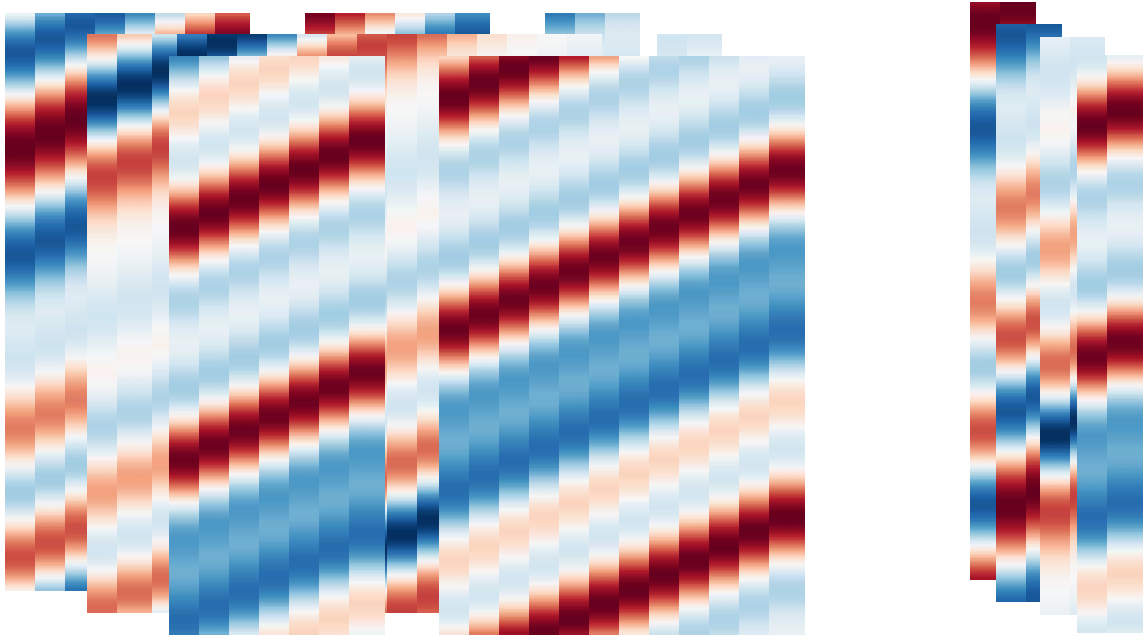
$$f_\theta \approx \mathcal{P}_h$$

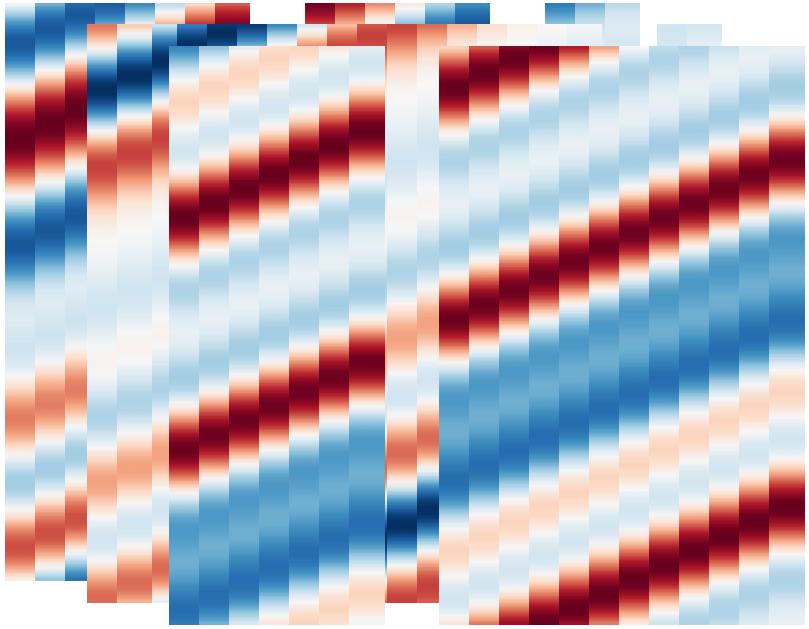


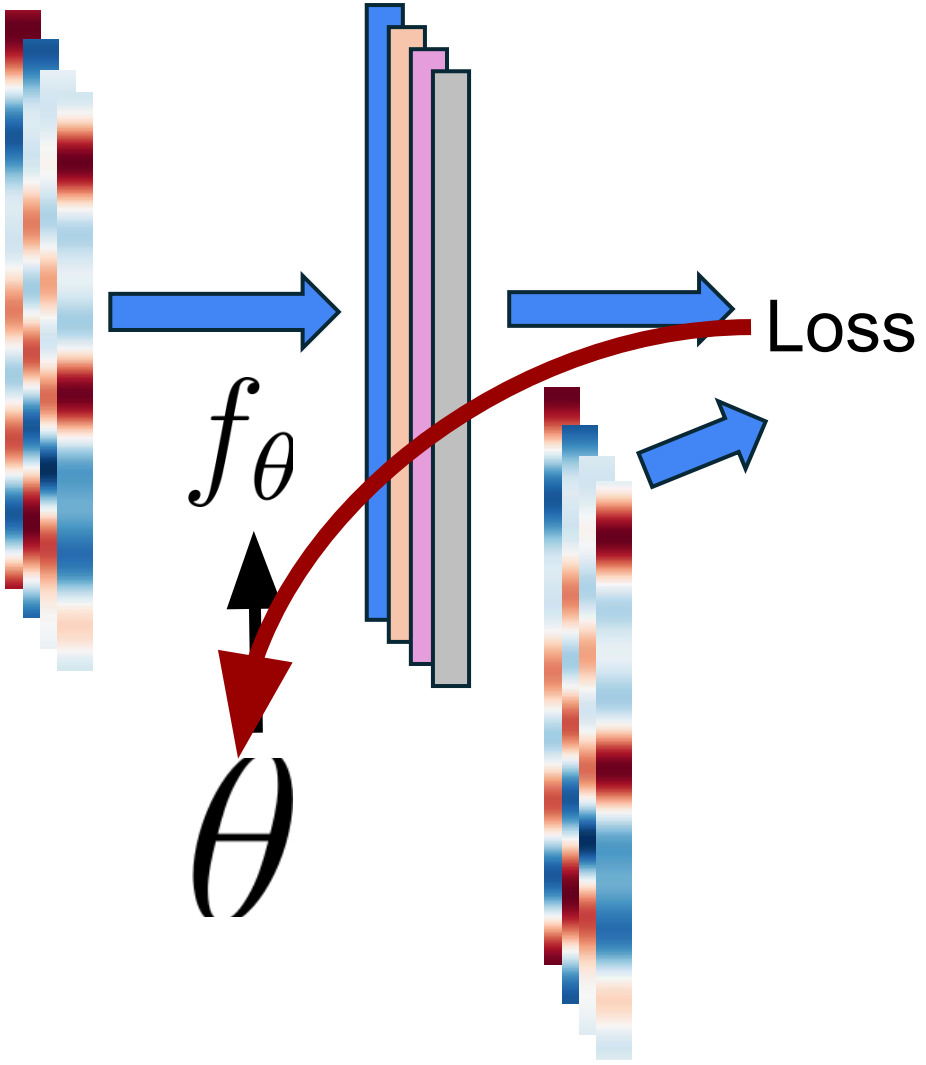
```
trj_set = jax.vmap(
    rollout
)(u_0_set)
```

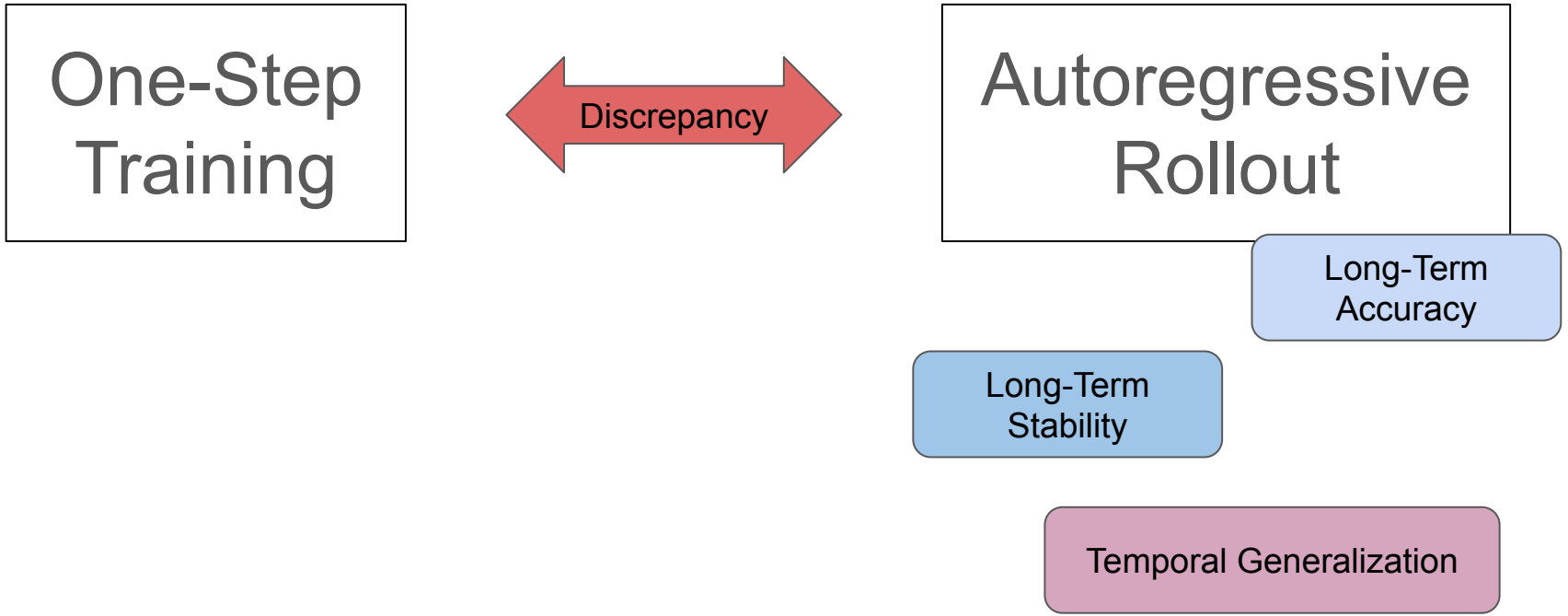


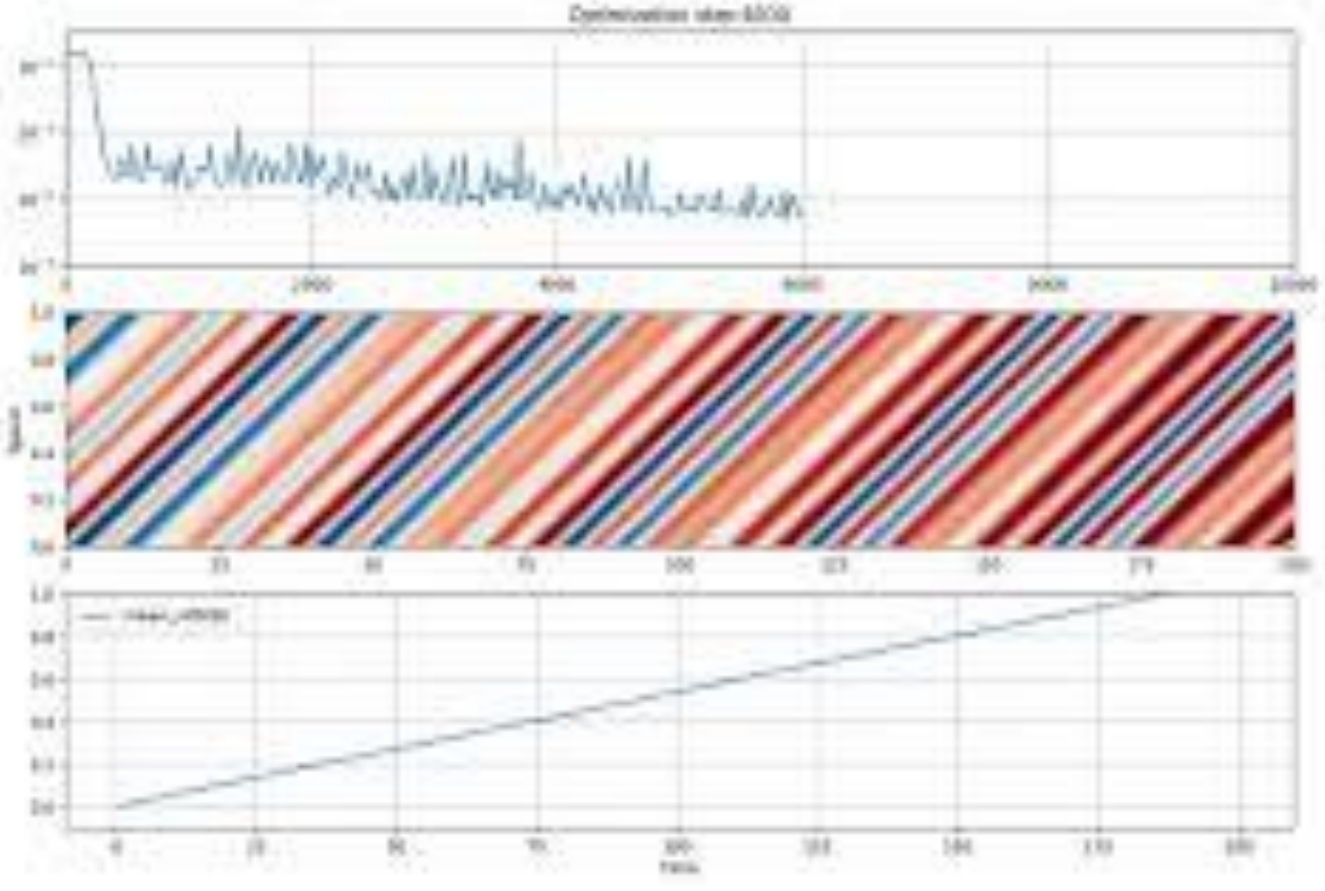


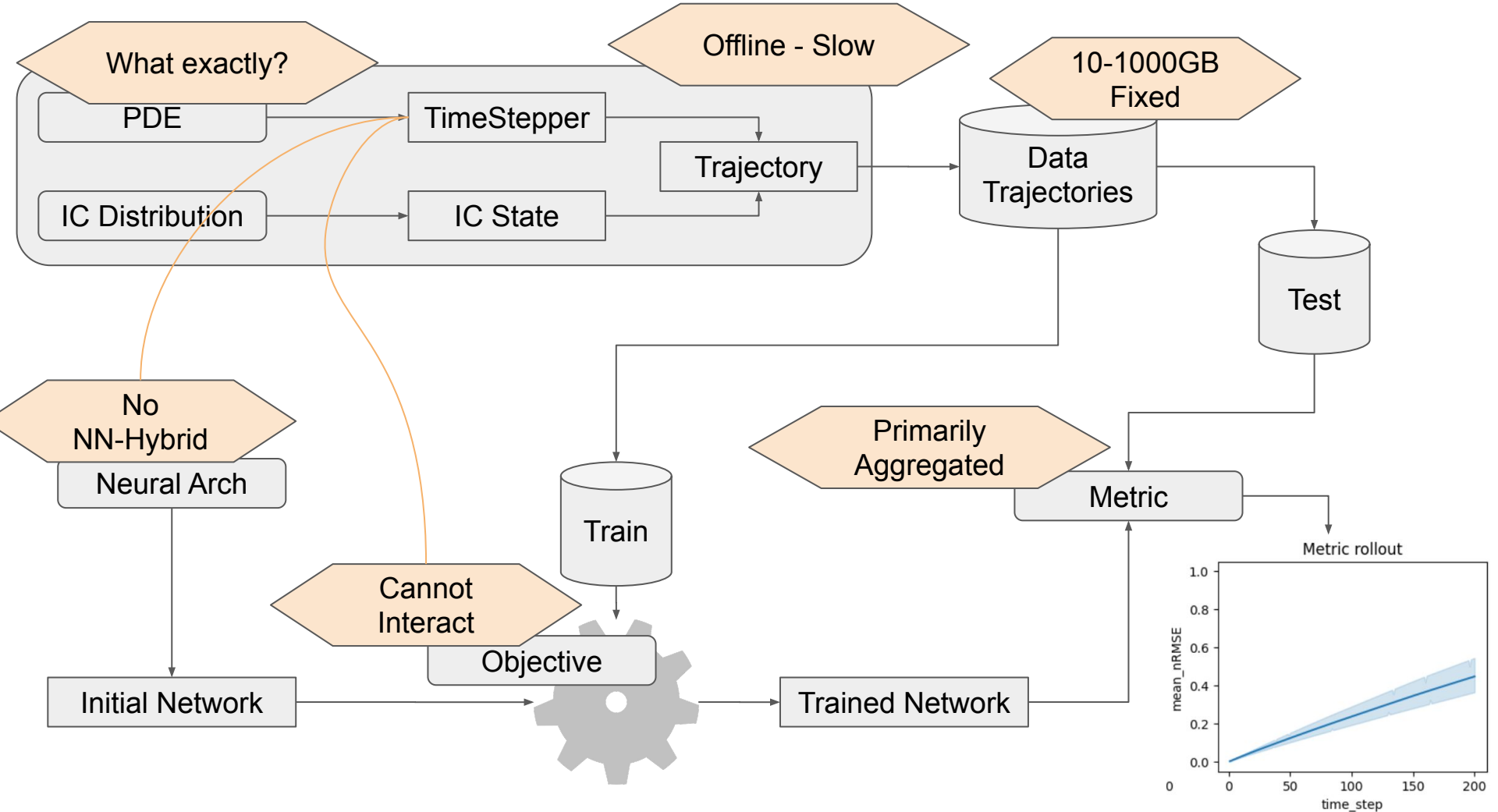





Loss
f_θ
θ


One-Step Training
Discrepancy
Autoregressive Rollout
Long-Term Accuracy
Long-Term Stability
Temporal Generalization




What exactly?
Offline - Slow
10-1000GB
Fixed
PDE
TimeStepper
Trajectory
IC Distribution
IC State
Data
Trajectories
Test
No
NN-Hybrid
Neural Arch
Primarily
Aggregated
Metric
Train
Cannot
Interact
Objective
Initial Network
Trained Network
Metric rollout
mean_nRMSE
time_step
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200

1D

PDE

IC Distribution

fast

TimeStepper

IC State

Trajectory

Procedurally generated

Data Trajectories

Test

hybrid

Neural Arch

Initial Network

Train

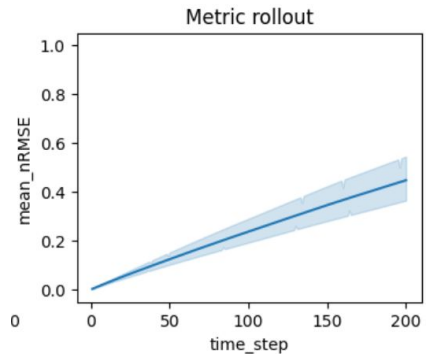
DP

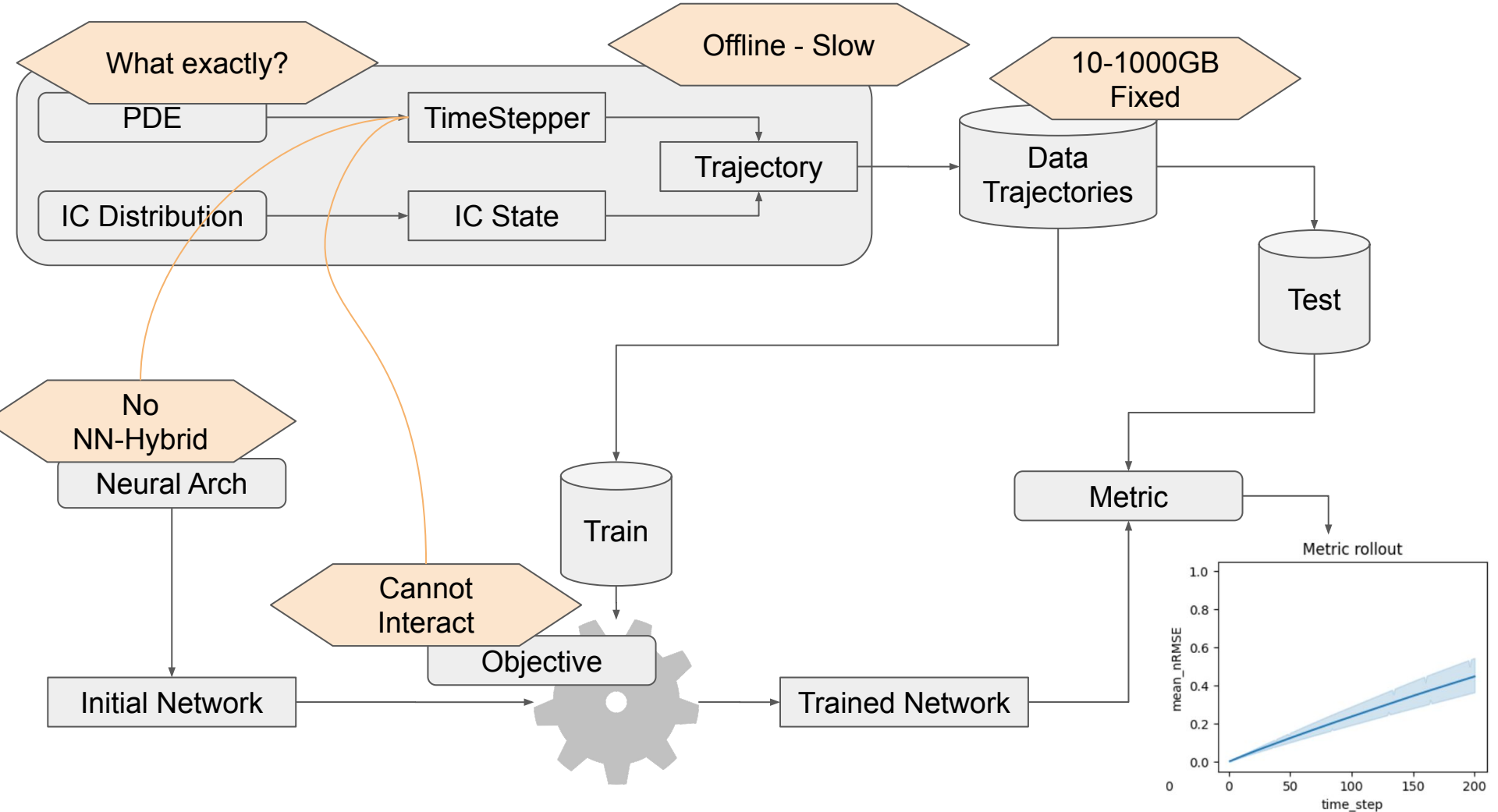
Objective

Trained Network

Rollout

Metric




What exactly?
Offline - Slow
10-1000GB
Fixed
PDE
TimeStepper
Trajectory
IC Distribution
IC State
Data
Trajectories
Test
No
NN-Hybrid
Neural Arch
Train
Metric
Cannot
Interact
Objective
Initial Network
Trained Network
Metric rollout
mean_nRMSE
time_step
0.0
0.2
0.4
0.6
0.8
1.0
0
50
100
150
200

1D

PDE

IC Distribution

fast

TimeStepper

IC State

Trajectory

Procedurally generated

Data Trajectories

Test

hybrid

Neural Arch

Initial Network

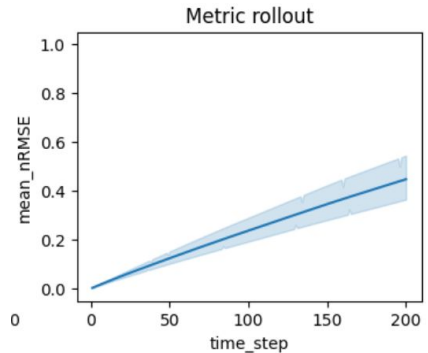
Train

DP

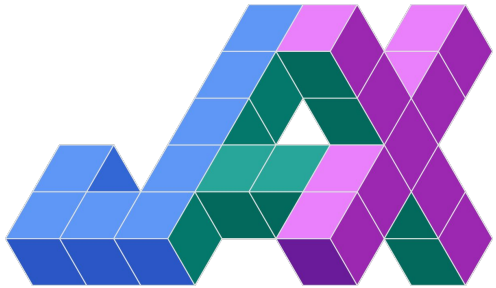
Objective

Trained Network

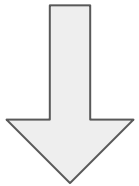
Metric



Key Contribution: Fourier-ETDRK



$$\partial_t u = \mathcal{L}u + \mathcal{N}(u)$$

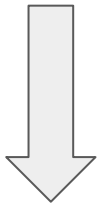


$$\hat{u}_h^{[t+1]} = \exp(\hat{\mathcal{L}}_h \Delta t) \odot \hat{u}_h^{[t]} + \frac{\exp(\hat{\mathcal{L}}_h \Delta t) - 1}{\hat{\mathcal{L}}_h} \odot \hat{\mathcal{N}}_h(\hat{u}_h^{[t]})$$

PDE Identifiers (Difficulty-based interface)

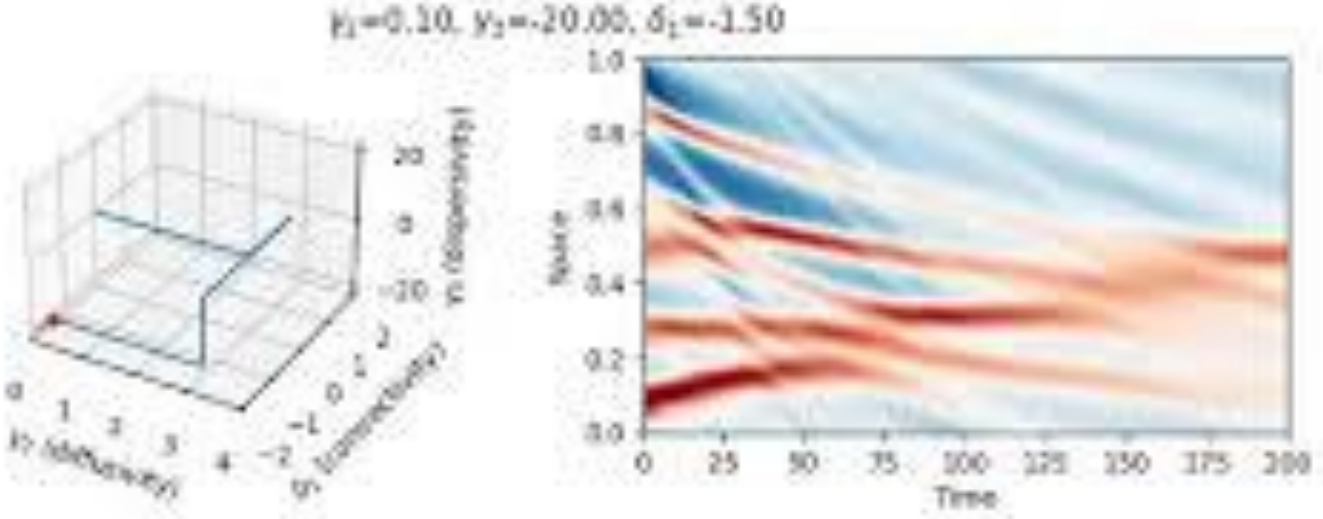
$$\frac{\partial u}{\partial t} = b_1 \frac{1}{2}\frac{\partial u^2}{\partial x} + a_0 u + a_1 \frac{\partial u}{\partial x} + a_2 \frac{\partial^2 u}{\partial x^2} + a_3 \frac{\partial^3 u}{\partial x^3} + a_4 \frac{\partial^4 u}{\partial x^4} + \dots$$

$$\{b_1, a_0, a_1, a_2, a_3, a_3, ...\} \times \{D, L, N, \Delta t\} \times \text{numerics}$$



\+ Stable 2nd order ETDRK

$$\{\delta_1, \gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \dots\} \times \{D, N\}$$


γ_1=0.10, γ_2=-20.00, δ_1=-1.50
20
0
-20
η (dispersivity)
0
1
2
3
4
k (diffusivity)
2
1
0
-1
-2
ζ (convection)
1.0
0.8
0.6
0.4
0.2
0.0
Value
0
25
50
75
100
125
150
175
200
Time

Unrolled Training Methodologies

$$L_{\text{one}}(\theta) = \mathbb{E}_{u_h \propto \mathcal{D}_h} \left[l(f_\theta(u_h), \mathcal{P}_h(u_h)) \right]$$

$$L_{\text{sup unr.}}(\theta) = \mathbb{E}_{u_h \propto \mathcal{D}_h} \left[\sum_{b=1}^{B=T} l(f_\theta^b(u_h), \mathcal{P}_h^b(u_h)) \right]$$

$$L(\theta) = \mathbb{E}_{u_h \propto \mathcal{D}_h} \left[\sum_{t=0}^{(T-B)} \sum_{b=1}^{B} l\left(f_\theta^{t+b}(u_h),\ \mathcal{P}_h^b(f_\theta^t(u_h)) \right) \right]$$

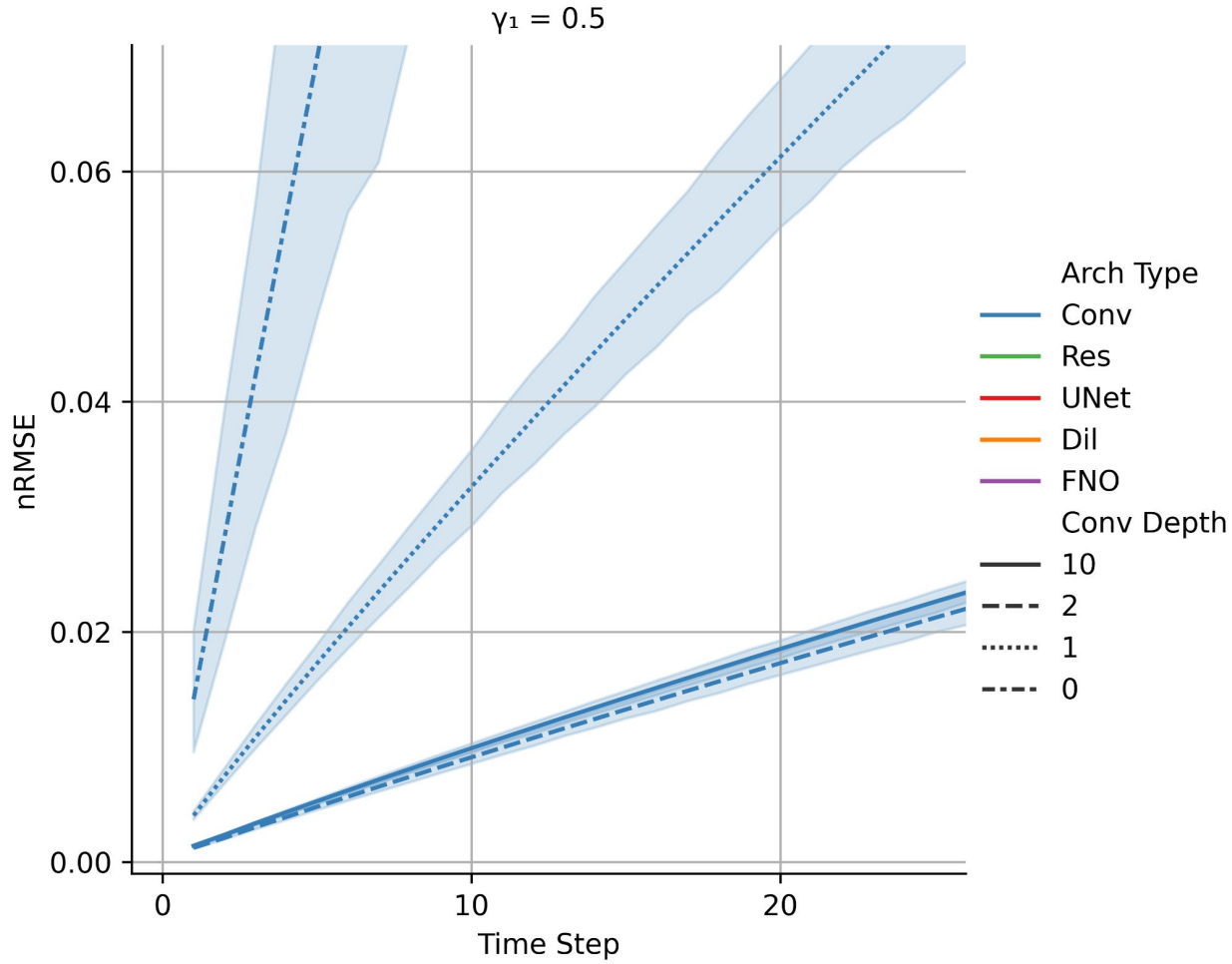
Axes of Investigation

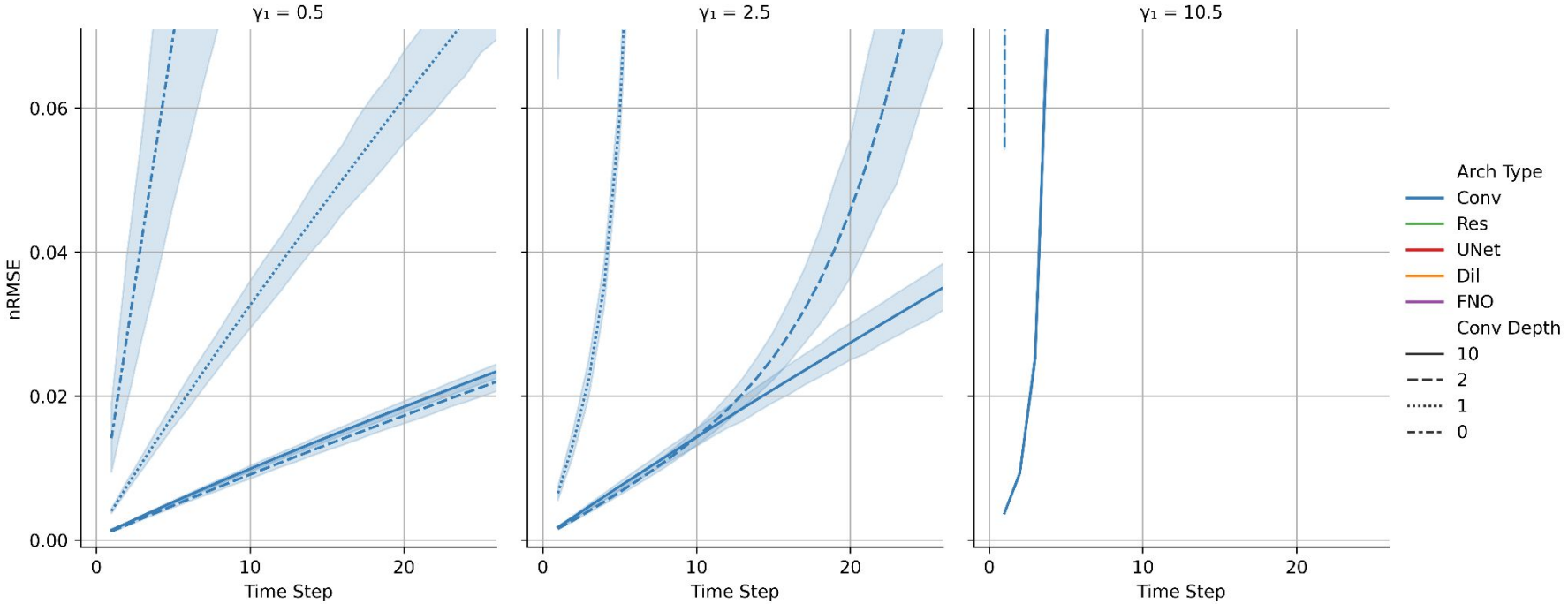
- PDE (in terms of its difficulty, resolution and spatial dims)
- Neural Architecture
- Learning Methodology
- Learning Goal (prediction or neural-hybrid correction)

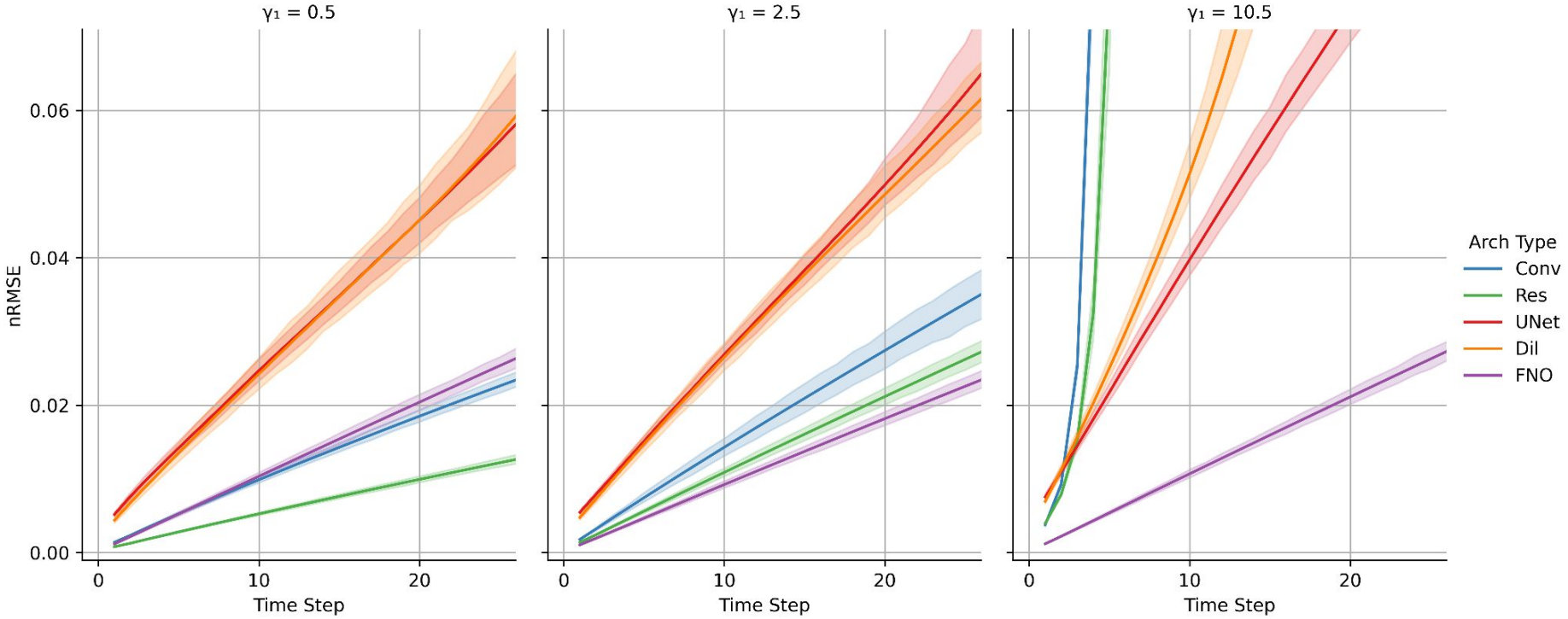
+ seed statistics for hypothesis testing

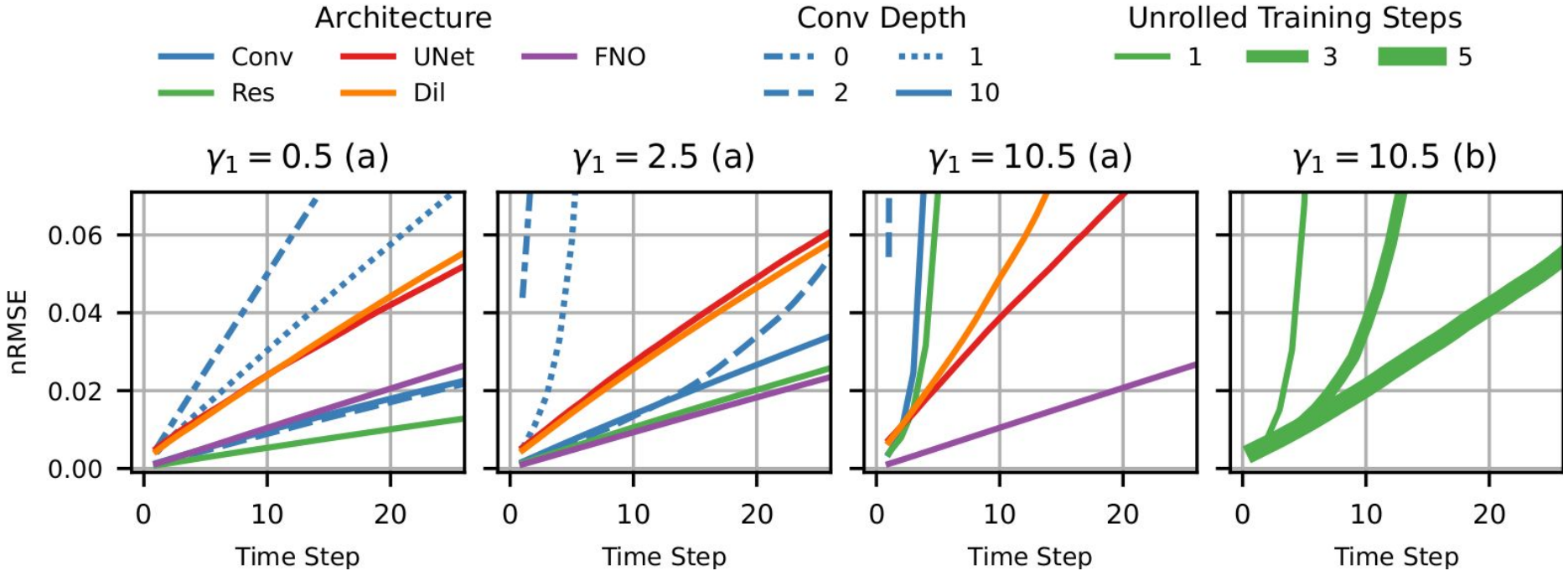
```
In [1]: import apebench

In [2]: advection_scenario = apebench.scenarios.difficulty.Advection()

In [3]: print(advection_scenario)
Advection(
  num_spatial_dims=1,
  num_points=160,
  num_channels=1,
  ic_config='fourier;5;true;true',
  num_warmup_steps=0,
  num_train_samples=50,
  train_temporal_horizon=50,
  train_seed=0,
  num_test_samples=30,
  test_temporal_horizon=200,
  test_seed=773,
  optim_config='adam;10_000;warmup_cosine;0.0;1e-3;2_000',
  batch_size=20,
  num_trjs_returned=1,
  record_loss_every=100,
  vlim=(-1.0, 1.0),
  report_metrics='mean_nRMSE',
  callbacks='',
  gammas=(0.0, -4.0, 0.0, 0.0, 0.0),
  coarse_proportion=0.5,
  advection_gamma=-4.0
)
```

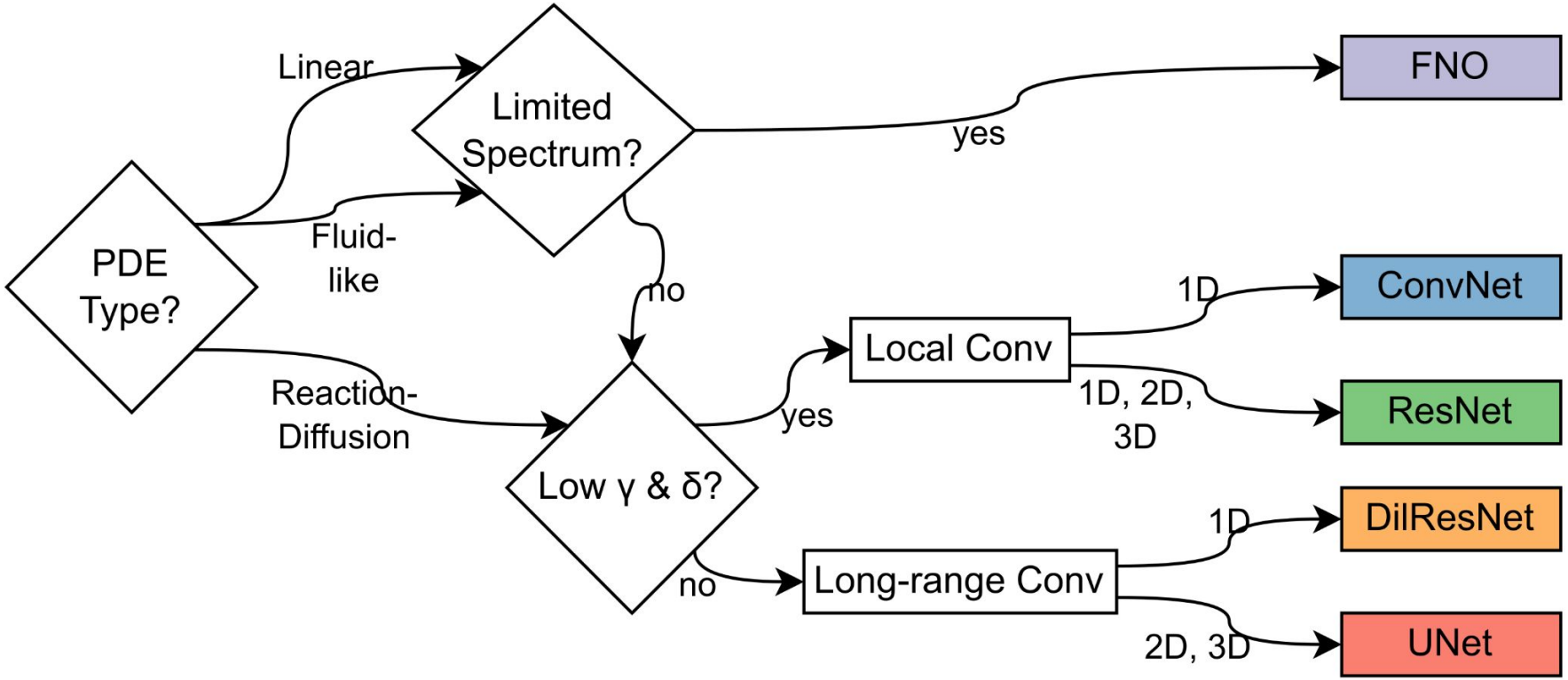
```
CONFIGS = [
    {
        "scenario": "diff_adv",
        "task": "predict",
        "net": net,
        "train": "one",
        "start_seed": s,
        "num_seeds": 10,
        "advection_gamma": advection_gamma,
    }
    for s in [0, 10, 20, 30, 40]
    for net in [
        *[f"Conv;34;{depth};relu" for depth in [0, 1, 2, 10]],
        "UNet;12;2;relu",  # 27'193 params, 29 receptive field per direction
        "Res;26;8;relu",  # 32'943 params, 16 receptive field per direction
        "FNO;12;18;4;gelu",  # 32'527 params, inf receptive field per direction
        "Dil;2;32;2;relu",  # 31'777 params, 20 receptive field per direction
    ]
    for advection_gamma in [
        0.5,
        2.5,
        10.5,
    ]
]
```


γ₁ = 0.5
nRMSE
0.00
0.02
0.04
0.06
0
10
20
Time Step
Arch Type
Conv
Res
UNet
Dil
FNO
Conv Depth
10
2
1
0


γ₁ = 0.5
γ₁ = 2.5
γ₁ = 10.5
nRMSE
0.00
0.02
0.04
0.06
0
10
20
Time Step
Arch Type
Conv
Res
UNet
Dil
FNO
Conv Depth
10
2
1
0


γ₁ = 0.5
γ₁ = 2.5
γ₁ = 10.5
nRMSE
0.00
0.02
0.04
0.06
0
10
20
Time Step
Arch Type
Conv
Res
UNet
Dil
FNO


Architecture
Conv
UNet
FNO
Res
Dil
Conv Depth
0
1
2
10
Unrolled Training Steps
1
3
5
$\gamma_1 = 0.5$ (a)
$\gamma_1 = 2.5$ (a)
$\gamma_1 = 10.5$ (a)
$\gamma_1 = 10.5$ (b)
nRMSE
0.00
0.02
0.04
0.06
0
10
20
Time Step


PDE Type?
Linear
Fluid-like
Reaction-Diffusion
Limited Spectrum?
yes
FNO
no
Low γ & δ?
yes
Local Conv
1D
ConvNet
1D, 2D, 3D
ResNet
no
Long-range Conv
1D
DilResNet
2D, 3D
UNet

What can **APE**Bench be used for

- Studies
  - Understand neural emulators better and how to train them
- Benchmarking
  - For the first time: Neural-Hybrid approaches!
- Data Generation
  - for Foundation Models

https://tum-pbs.github.io/apebench/use_cases/

APEBENCH in a nutshell

- JAX-based
- Fast reference simulator based on spectral methods:
  - Procedural data generation in seconds!
  - Differentiable Physics
- 46 PDEs in 1D, 2D, and 3D with unique identifiers
- Big selection of modern Emulator Architectures in JAX
- Built-in unrolled training with differentiable physics
- Built-in neural hybrid emulation
- An integrated volume renderer for 2D & 3D
- Understand neural emulators and draw analogies with classical numerical methods

APEBench: A Benchmark for Autoregressive Neural Emulators of PDEs

Felix Koehler
Technical University of Munich
Munich Center for Machine Learning
f.koehler@tum.de

Simon Niedermayr
Technical University of Munich
simon.niedermayr@tum.de

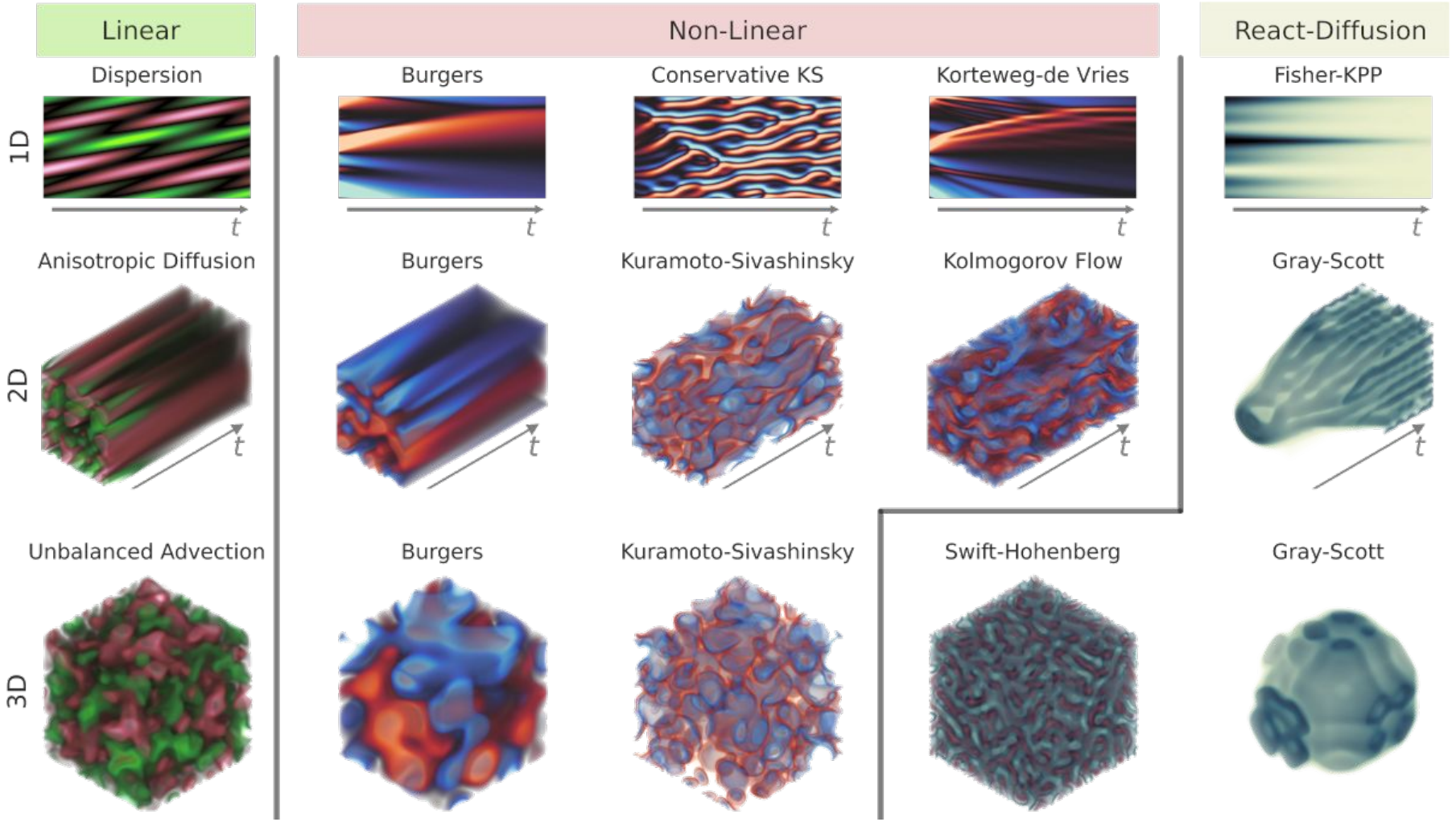
Rüdiger Westermann
Technical University of Munich
westermann@tum.de

Nils Thuerey
Technical University of Munich
nils.thuerey@tum.de

```
pip install apebench
```

tum-pbs.github.io/apebench

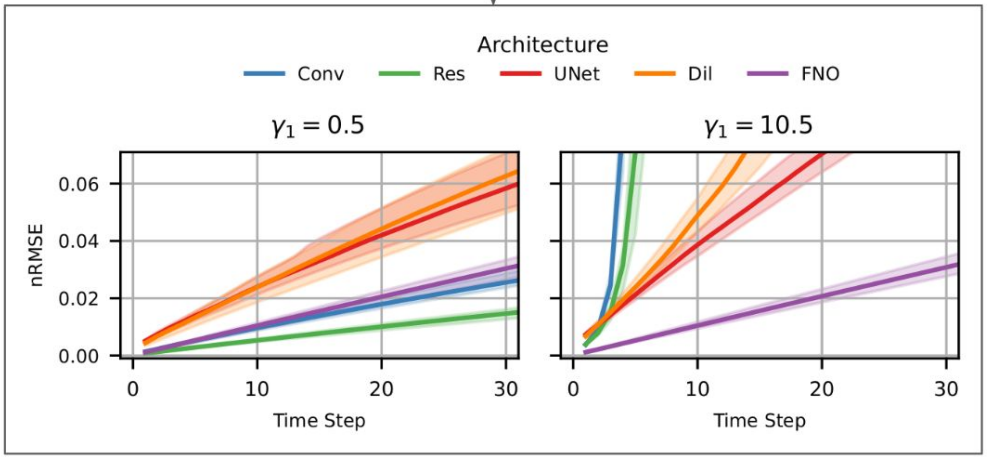
Backup Slides


Linear
Non-Linear
React-Diffusion
1D
2D
3D
Dispersion
Burgers
Conservative KS
Korteweg-de Vries
Fisher-KPP
Anisotropic Diffusion
Burgers
Kuramoto-Sivashinsky
Kolmogorov Flow
Gray-Scott
Unbalanced Advection
Burgers
Kuramoto-Sivashinsky
Swift-Hohenberg
Gray-Scott
t

```python
CONFIGS = [
    {
        "scenario": "diff_adv",
        "task": "predict",
        "net": net,
        "train": "one",
        "advection_gamma": advection_gamma,
    }
    for net in [
        "Conv;34;10;relu",  # 31'757 params, 11 receptive field per direction
        "UNet;12;2;relu",  # 27'193 params, 29 receptive field per direction
        "Res;26;8;relu",  # 32'943 params, 16 receptive field per direction
        "FNO;12;18;4;gelu",  # 32'527 params, inf receptive field per direction
        "Dil;2;32;2;relu",  # 31'777 params, 20 receptive field per direction
    ]
    for advection_gamma in [0.5, 2.5]
]
```

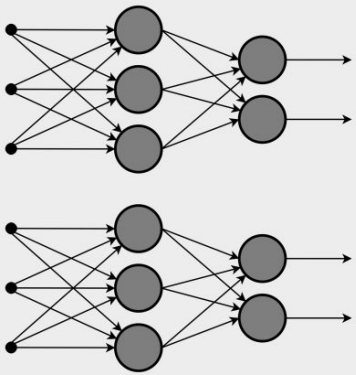
pandas

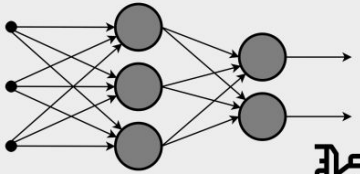
seaborn

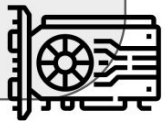


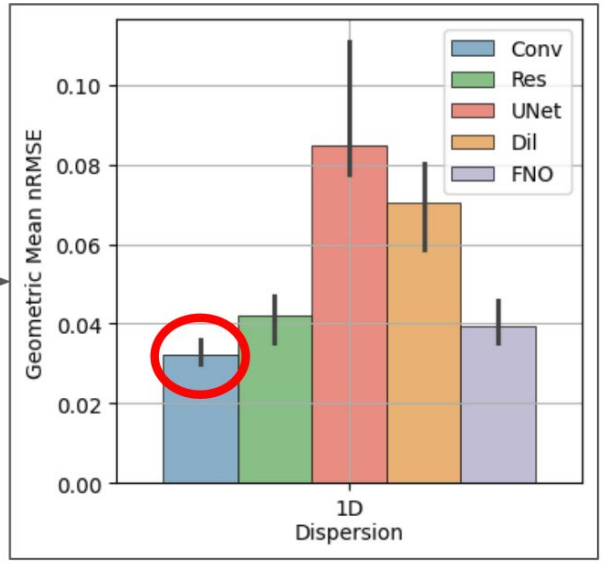
Study Definition

Multiple Random Seeds for init and minibatching



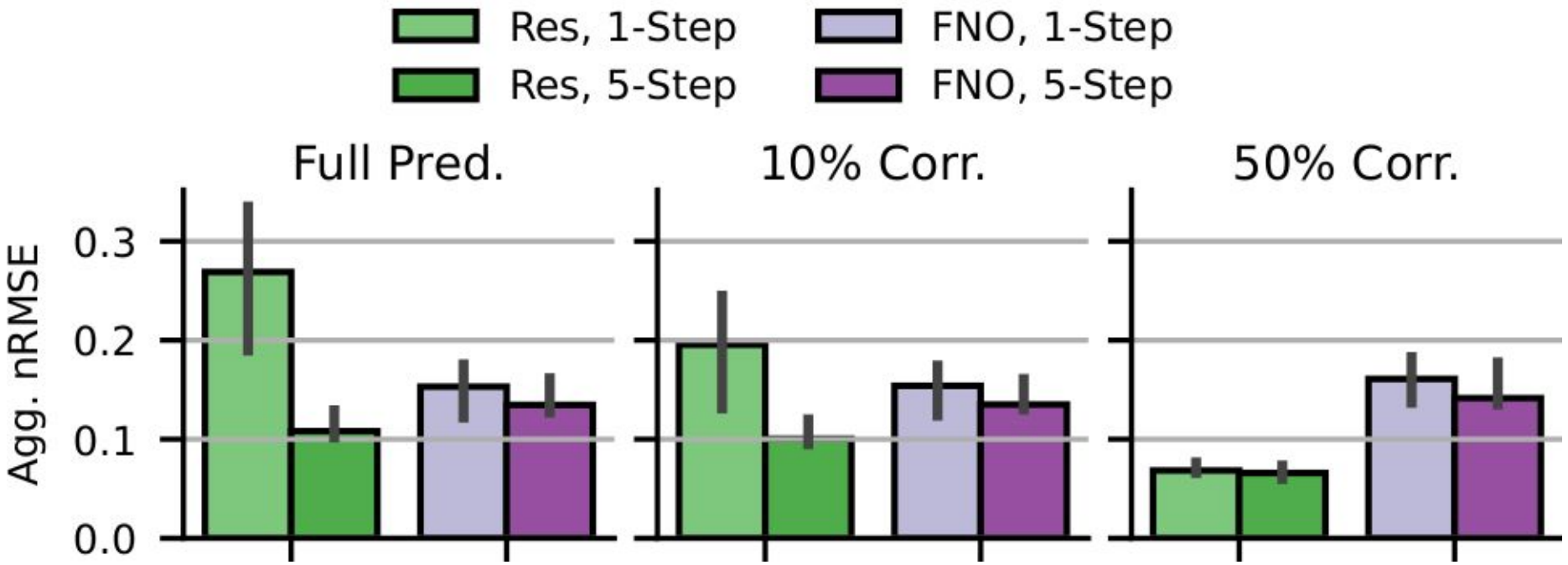




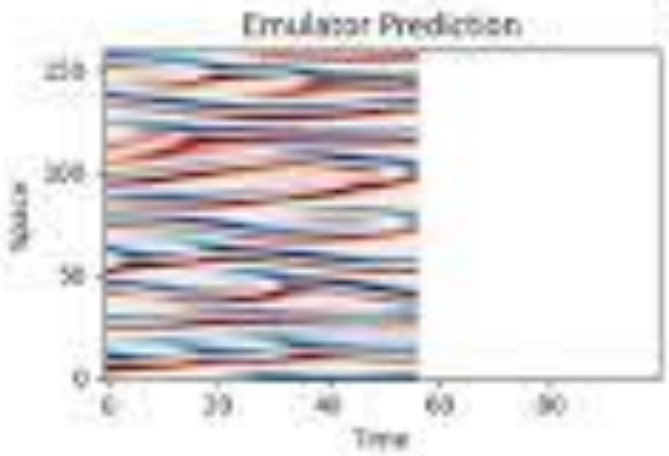


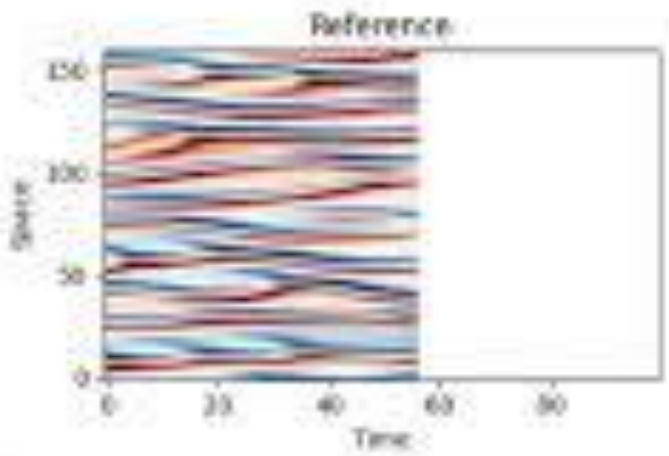
T-Test

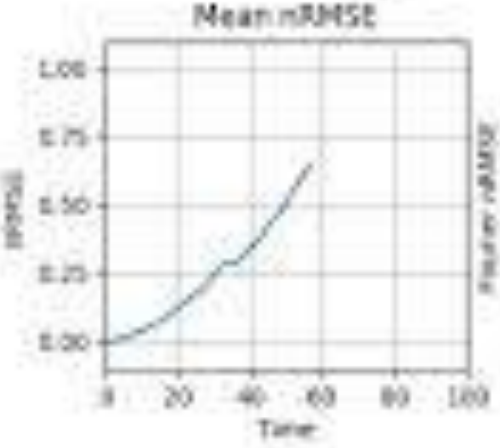
	Conv	Res	UNet	Dil	FNO
pvalue	1.0	1.1e-6	2.9e-9	3.7e-3	7.9e-8

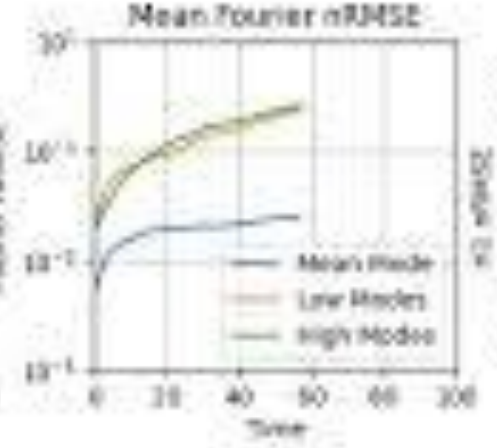

Res, 1-Step
Res, 5-Step
FNO, 1-Step
FNO, 5-Step
Full Pred.
10% Corr.
50% Corr.
Agg. nRMSE
0.3
0.2
0.1
0.0

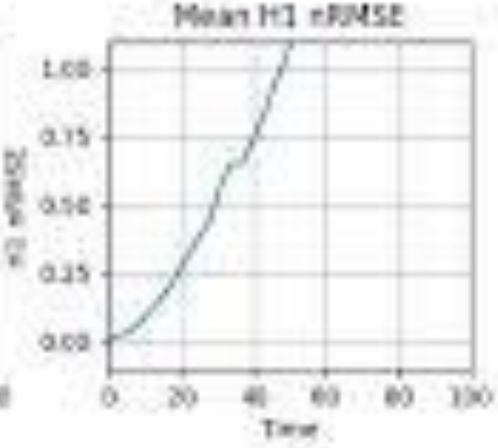
In [57]:

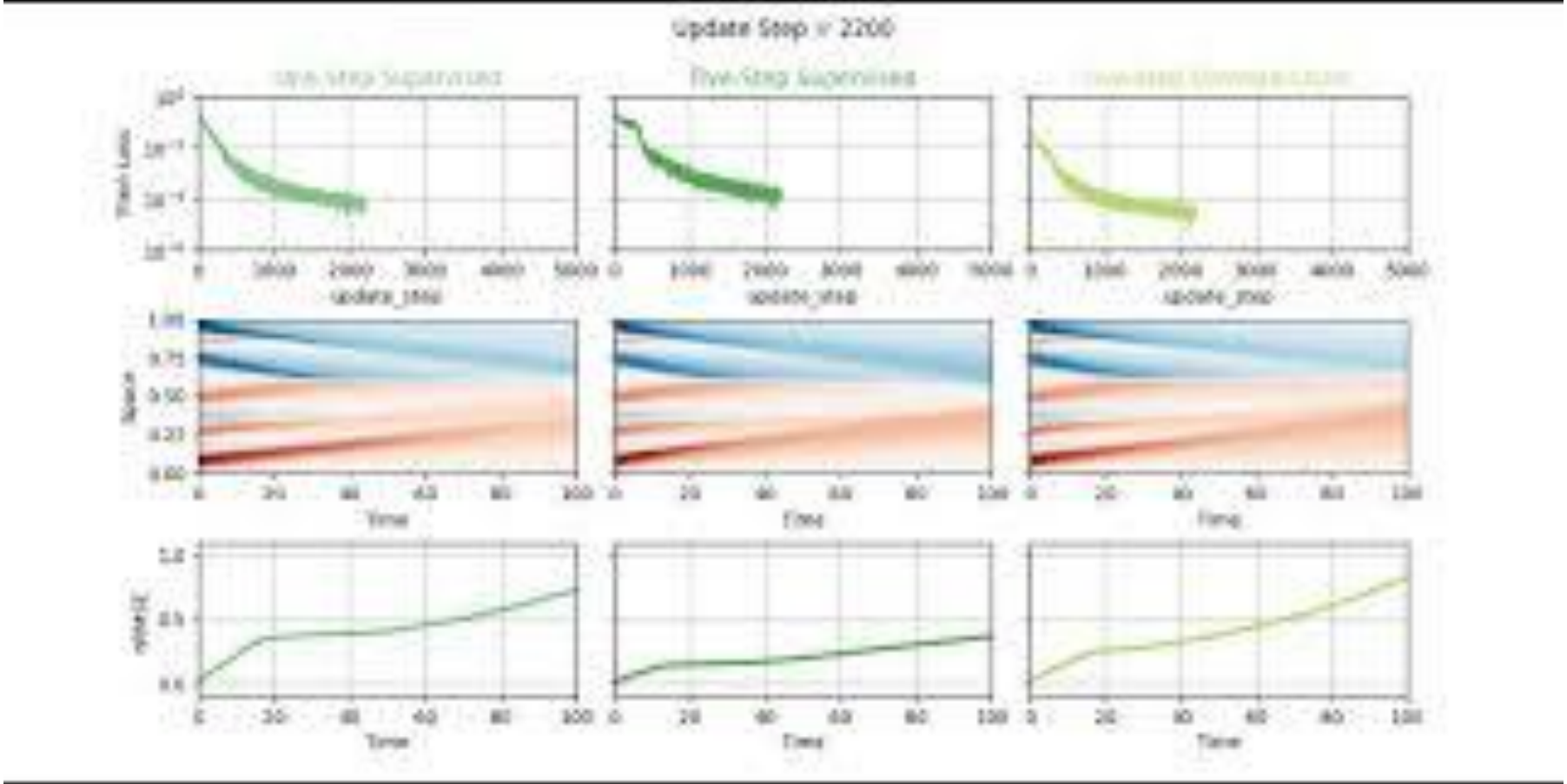

Emulator Prediction
Space
Time

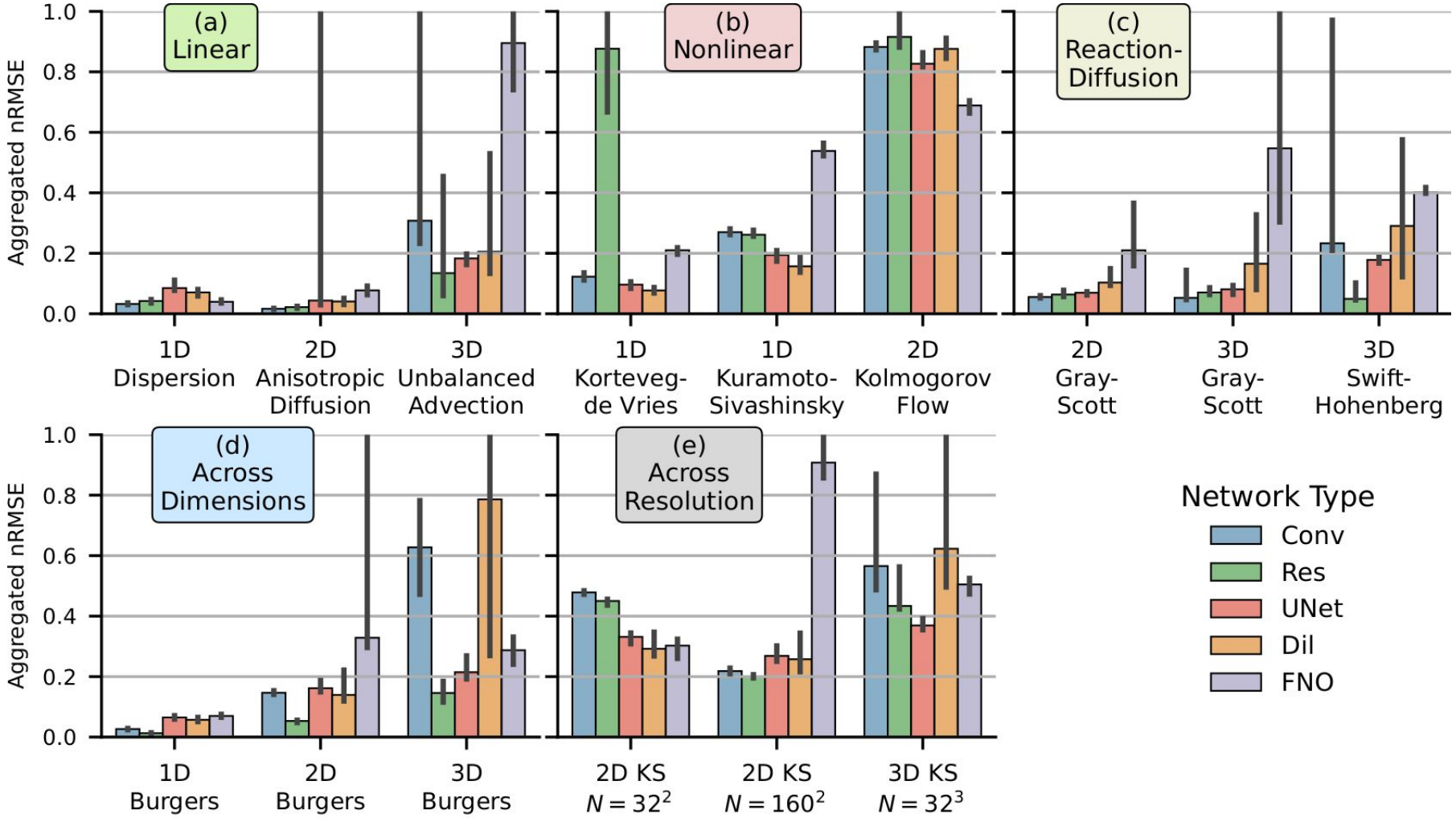

Reference
Space
Time


Mean nRMSE
nRMSE
Time


Mean Fourier nRMSE
Fourier nRMSE
Mean Mode
Low Modes
High Modes
Time


Mean H1 nRMSE
H1 nRMSE
Time


Update Step = 2200
Time
Time
Time


(a)
Linear
(b)
Nonlinear
(c)
Reaction-
Diffusion
(d)
Across
Dimensions
(e)
Across
Resolution
Aggregated nRMSE
1.0
0.8
0.6
0.4
0.2
0.0
1D
Dispersion
2D
Anisotropic
Diffusion
3D
Unbalanced
Advection
1D
Korteveg-
de Vries
1D
Kuramoto-
Sivashinsky
2D
Kolmogorov
Flow
2D
Gray-
Scott
3D
Gray-
Scott
3D
Swift-
Hohenberg
1D
Burgers
2D
Burgers
3D
Burgers
2D KS
$N = 32^2$
2D KS
$N = 160^2$
3D KS
$N = 32^3$
Network Type
Conv
Res
UNet
Dil
FNO