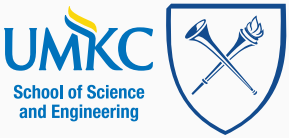



Accurate Fine-Tuning of Spatiotemporal Fourier Neural Operator for Turbulent Flows

Shuhao Cao[1] Francesco Brarda[2] Ruipeng Li[3] Yuanzhe Xi[2]

[1]UMKC [2]Emory [3]LLNL

Efficient and Reliable Deep Learning Methods and their Scientific Applications
Birs Workshop, June 2025

Question: which is more important in operator learning tasks?
Data or Model (training algorithm, neural architectures)?

Spatiotemporal Operator Learning

Setting the Stage

Toy model: approximating Navier-Stokes equations in a 2D periodic box

- (Velocity) Find $\boldsymbol{u} \in \boldsymbol{H}^1(\mathbb{T}^2) \cap \{\boldsymbol{v} \in \boldsymbol{H}(\mathrm{div}) : \nabla \cdot \boldsymbol{v} = 0\}$

$$\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} - \nu \Delta \boldsymbol{u} + \alpha \boldsymbol{u} = \boldsymbol{f},$$

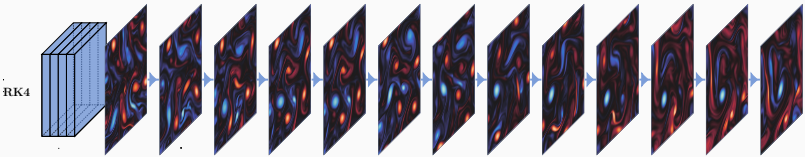
- (Vorticity-Streamfunction) Find $\omega, \psi \in H^1(\mathbb{T}^2)$

$$\partial_t \omega + (\nabla^\perp \psi) \cdot \nabla \omega - \nu \Delta \omega + \alpha \omega = 0, \quad \omega + \Delta \psi = 0.$$

A long and rich history of numerical analysis for the NSE

- Projection schemes: Chorin (1968), Shen (1992).
- (Pseudo) Spectral: Orszag (1971), Orszag (1972), Tadmor (1987), Ku, Taylor, and Hirsh (1987), Shen (1994).
- Finite element: Girault and Raviart (1986), Brezzi and Fortin (1991), Temam (1995).
- Stability, convergence, time-stepping stencils: Heywood and Rannacher (1986), C. Wang and J.-G. Liu (2002), He and W. Sun (2007), Gottlieb et al. (2012), Cheng and C. Wang (2016).

Time-Marching Solvers for NSE



- Crank-Nicolson for the diffusion term, explicit for the convection term, step size τ has to be comparable to $\mathcal{O}(\Delta x / \|\omega(t,\cdot)\|_\infty)$ and $o(\nu)$.[1]

$$\left(I - \frac{\tau}{2}\nu L\right)\omega^{(t_{\ell+1})} = \left(I + \frac{\tau}{2}\nu L\right)\omega^{(t_\ell)} - \tau\left(\nabla^\perp(-\Delta)^{-1}\omega^{(t_\ell)}\right)\cdot\nabla\omega^{(t_\ell)}.$$

- Aliasing error caused by the nonlinear term in pseudo-spectral methods: the modes extend "outside" of the approximation space. Possible remedies: filtering ("2/3-rule"[2]), higher-order[3], or disspating the aliased modes.

[1] X. Wang (2012). "An efficient second order in time scheme for approximating long time statistical properties of the two dimensional Navier–Stokes equations". In: *Numerische Mathematik*.

[2] J. Goodman, T. Hou, and E. Tadmor (1994). "On the stability of the unsmoothed Fourier method for hyperbolic equations". In: *Numerische Mathematik*.

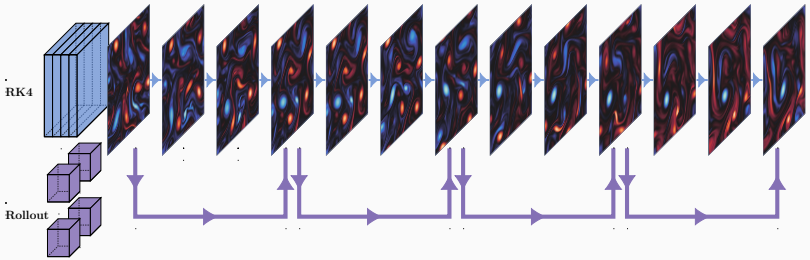
[3] S. Nagarajan, S. K. Lele, and J. H. Ferziger (2003). "A robust high-order compact method for large eddy simulation". In: *Journal of Computational Physics*.

Learning Operators between "Function Spaces"

Operator-valued NN and operator learning: $\mathcal{L}_\theta : \mathbb{R}^{n\times m\times d_{\text{in}}} \to \mathbb{R}^{n\times m\times d_{\text{out}}}$.

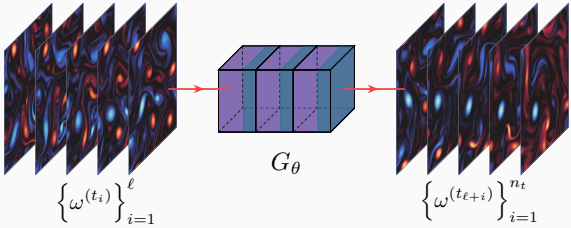
- How the "basis" (frames) are constructed and how are they aggregated?
  - Latent frames are (nonlinear) universal approximators, and then are linearly aggregated through coefficients independent of the current latent space; Fourier Neural Operator: Z. Li, Kovachki, et al. (2021), and many others.
  - Frames/"basis" are linear projections of the current latent representations, then aggregated nonlinearly (input-dependent kernel integral). Nonlocal kernel: Buades, Coll, and Morel (2005), Gilboa and Osher (2007). Transformer/Attention: C. (2021), Z. Li, Meidani, and Farimani (2023), Hao et al. (2023), Bartolucci et al. (2024), Wu et al. (2024), Yu et al. (2024), and many others.
  - Both aggregation and frames are nonlinear. DeepONet: Lu et al. (2021), S. Wang, H. Wang, and Perdikaris (2021).
- Neural super-resolution operator. Spatial: Kochkov et al. (2021); temporal: Z. Sun, Yang, and Yoo (2023).

Learning Time-Marching with Data



- Train an "autoregressive" operator learner G_θ that maps $\{\omega(t,\cdot)\}_{t\in(t_{\ell+1},t_{\ell+10})}$ to $\omega(t_{\ell+11},\cdot)$ for different ℓs. This procedure repeats a "roll-out" prediction until certain steps.
- These "time-slices" are treated as the input "channels" in neural operators, such as FNO and its variants, Transformer-based NOs (Galerkin, GNOT, Oformer, DPOT, Transolver, many others).
- Time steps can be pretty big (e.g., $= 100\tau$), yet the relative difference with the ground truth is horrible in evaluation ($\approx 1\%$), no stability at all.

Learning Maps between Bochner Spaces



Problem of interest

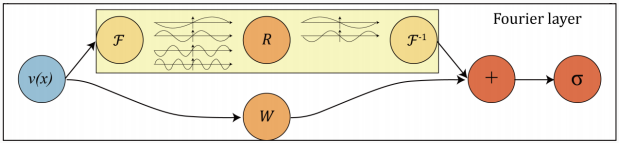
Construct an operator-valued neural network G_θ to "approximate" G:

$$G : L^2(t_1, t_\ell; \mathcal{V}) \to L^2(t_{\ell+1}, t_{\ell+n_t}; \mathcal{V}), \quad \{\omega(t,\cdot)\}_{t\in(t_1,t_\ell)} \mapsto \{\omega(t,\cdot)\}_{t\in(t_{\ell+1},t_{\ell+n_t})}.$$

$$G_\theta : \mathcal{S}_\ell \to \mathcal{S}_{n_t}, \quad \mathbb{R}^{n\times n\times \ell} \ni \mathbf{w}_{\text{in}} \mapsto \mathbf{w}_{\text{out}} \in \mathbb{R}^{n\times n\times n_t}.$$

- $\mathcal{V} := H^1(\mathbb{T}^2)$ or $\{\boldsymbol{v} \in \boldsymbol{H}^1(\mathbb{T}^2) : \nabla \cdot \boldsymbol{v} = 0\}$.
- $\mathcal{S}_n \simeq \prod_{j=1}^{n} \mathcal{S}$, where $\mathcal{S}$ is a finite-dimensional approximation space of $\mathcal{V}$.
- n, ℓ, n_t can all vary for training and evaluation.

Fourier Neural Operator



Source: Figure 1 from the FNO paper by A. Stuart with his collaborators and students.[4]

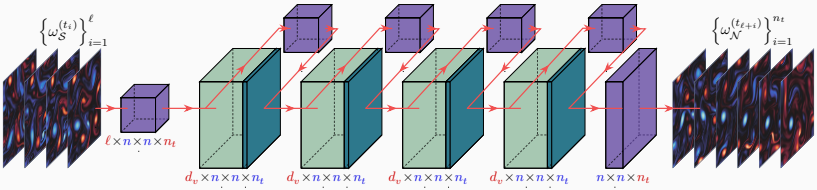
- The matrix-valued kernel matrix $R_\theta(x)$ in the Fourier multiplication operator SpConv is learned from data

$$v^{\text{out}}(x) = \int_\Omega \kappa_\theta(x-y)v(y)dy = \mathcal{F}^{-1}\mathcal{F}\left(\int_\Omega \kappa_\theta(x-y)v(y)dy\right)$$
$$= \mathcal{F}^{-1}\left(\mathcal{F}(\kappa_\theta)\cdot\mathcal{F}(v)\right)(x) \approx: \mathcal{F}^{-1}(R_\theta\cdot\mathcal{F}(v))(x)$$

- Frequency truncation in parametrization: for 2D problems, $v \in \mathbb{R}^{H_i\times n\times n}$, $R_\theta \in \mathbb{C}^{H_o\times H_i\times d\times d}$ with $d \ll n$. Outputs can be viewed as learnable reduced "basis", e.g., $n = 128$ and $d = 12$ (# channel = # basis).

[4] Z. Li, N. B. Kovachki, et al. (2021). "Fourier Neural Operator for Parametric Partial Differential Equations". In: *International Conference on Learning Representations*.

Fourier Neural Operator



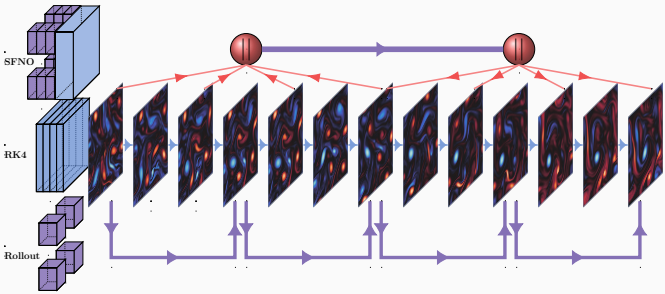
The architectural schematics of FNO for NSE: red represents fixed dimensions; blue represents dimensions that accept arbitrary-sized discretizations. : spectral convolution layer ; : pointwise `nn.Conv3d` that works as channel expansion/reduction; : pointwise nonlinearity.

- The novelty of FNO for NSE: for the input tensor of dimension $n \times n \times n_t$

  $$\text{channel dimension } d_v \leftarrow \text{time steps in the temporal dimension } n_t.$$

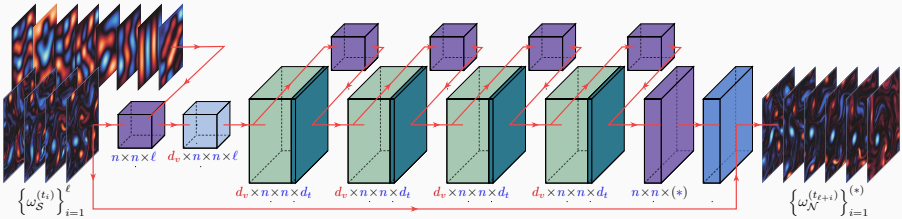
  However, this renders FNO unable to represent data pair in Bochner spaces. The number of parameters depends on the size of time discretization.
- The data for training and evaluation are prepared by applying a pointwise Gaussian normalizer that depends on the output steps n_t.

Toward Spatiotemporal Neural Operators



- **Goal**: construct an operator-valued neural network with arbitrary spatial- or temporal-grid sizes input/output.
- **Example**: if the end-to-end pipeline picks the time slices in a time span of 0.5 as input, the slices in the subsequent 0.5 as the output, e.g., training data pair with the input/output tensor of dimension `(64, 64, 10)`, then the evaluation should be able to deal with input size such as `(128, 128, 40)` or `(512, 512, 100)`.

Spatiotemporal Fourier Neural Operator 3D

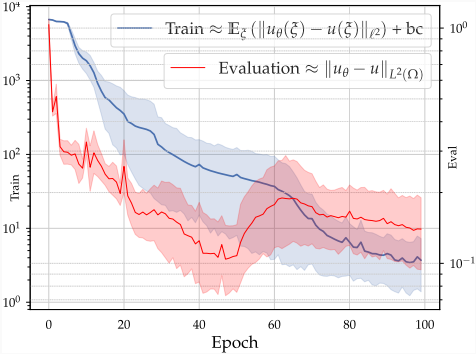


The architectural schematics of ST-FNO: ■: layer normalization after concatenation with positional encodings.

- Concatenation of random projection of the positional encodings with the input tensor act as a "channel expansion" (positional encodings $\oplus$ the input data, then go through a linear layer), which is a depth-wise linear combination.
- Layer normalization works as a dimension-agnostic normalizer.
- The new output operator has an extra single-channel spectral convolution layer with less truncated modes. It maps the latent time step dimension (d_t) to a given output time steps using FFT+iFFT's natural super-resolution by zero-padding the temporal steps.

How to Train Spatiotemporal FNO?

Motivations: Function Representation

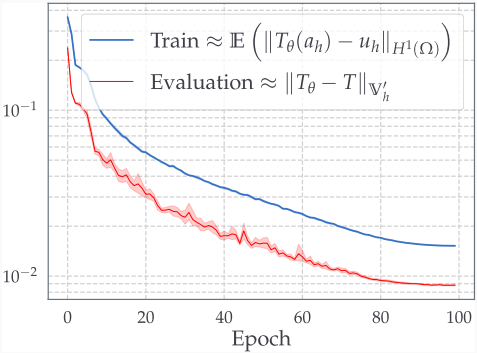


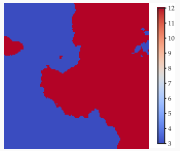
An extremely difficult problem due to the frequency principle[5] and the lottery ticket hypothesis[6]: train a function representer in low-dimensional spaces with a small number of parameters with no "labels". Trying to learn $-\Delta u = 2\pi^2 \sum_{k\in\{1,4,16\}} \sin(k\pi x)\sin(k\pi y)$. 1 epoch = 128 ADAM iterations. Error bars are plotted using different seeds (initialization).

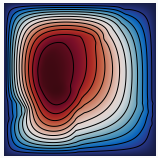
[5] B. Ronen, D. Jacobs, Y. Kasten, and S. Kritchman (2019). "The Convergence Rate of Neural Networks for Learned Functions of Different Frequencies". In: *Advances in Neural Information Processing Systems*.

[6] J. Frankle and M. Carbin (2019). "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*.

Motivations: Operator Learning







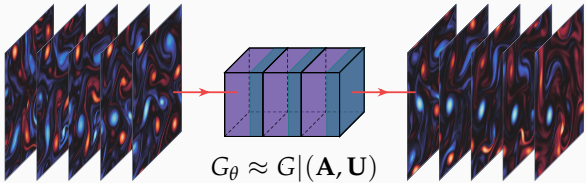
A not-so-difficult problem: "typical" convergence results for end-to-end operator-valued neural network training and evaluations[7] for a 2D benchmark problem of porous media[8][9].

[7] C. (2021). "Choose a Transformer: Fourier or Galerkin". In: *Advances in Neural Information Processing Systems (NeurIPS)*.

[8] A. P. Roberts and M. Teubner (1995). "Transport properties of heterogeneous materials derived from Gaussian random fields: bounds and simulation". In: *Physical Review E*.

[9] N. H. Nelsen and A. M. Stuart (2021). "The random feature model for input-output maps between Banach spaces". In: *SIAM Journal on Scientific Computing*.

Are Neural Operators Really Learning Operators?



> *Given the training data pairs $\{(a_m, u_m)\}_{m=1}^{M}$, the operator learning problem is a Bayesian inverse problem with a linear or nonlinear operator as the unknown object to be inferred from data.*

- DE HOOP, KOVACHKI, NELSEN, and STUART (2023)[10].

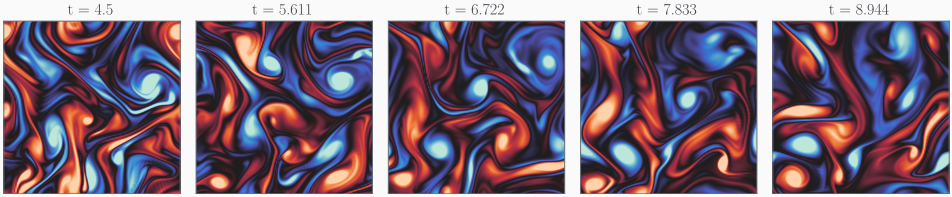
- With the "right" data, the Bayesian inverse problem is well-conditioned.

[10] M. V. DE HOOP, N. B. KOVACHKI, N. H. NELSEN, and A. M. STUART (2023). "Convergence rates for learning linear operators from noisy data". In: *SIAM/ASA Journal on Uncertainty Quantification*.

Statistical Property of NSE: Energy Cascade

- Inverse cascade: a flux of energy from smaller scales (high frequency) to larger scales (low frequency)[11][12].
- Direct cascade: if the energy dissipation is caused by larger scale vortices inducing vortex stretching via viscosity, then a stationary regime can be established[13].

$$\mathcal{E}(t,k) = \sum_{k-\delta k \le |\boldsymbol{k}| \le k+\delta k} |\widehat{\nabla \times \boldsymbol{u}}(t,\boldsymbol{k})|^2 \sim \mathcal{O}(k^{-\beta}) \quad \text{after } t > t_0$$

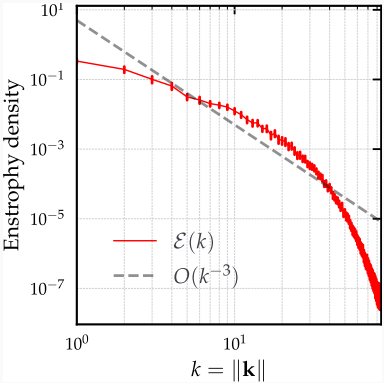


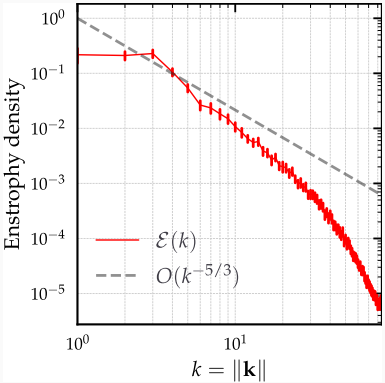
[11] A. N. Kolmogorov (1941). "The local structure of turbulence in incompressible viscous fluid for very large Reynolds". In: *Numbers. In Dokl. Akad. Nauk SSSR*.

[12] D. Kochkov et al. (2021). "Machine learning–accelerated computational fluid dynamics". In: *Proceedings of the National Academy of Sciences*.

[13] J. C. McWilliams (1984). "The emergence of isolated coherent vortices in turbulent flow". In: *Journal of Fluid Mechanics*.

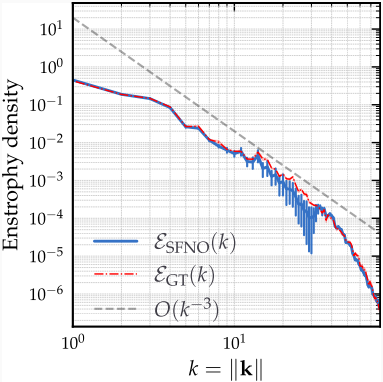
Statistical Property of NSE: Energy Cascade

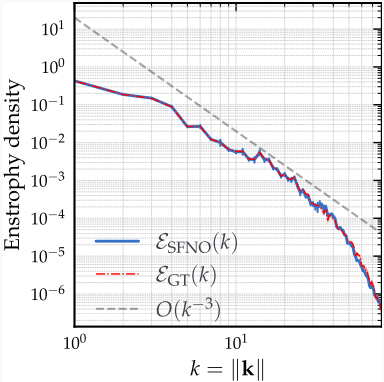




The plots of $\mathcal{E}(t,k)$ for all the training samples in the case of the direct cascade (left) and the inverse cascade (right). Both examples' initial conditions are sampled from fixed random fields, respectively. The error bars are plotted with $+/-$ 10 times the standard deviation from the mean to boost the visibility.

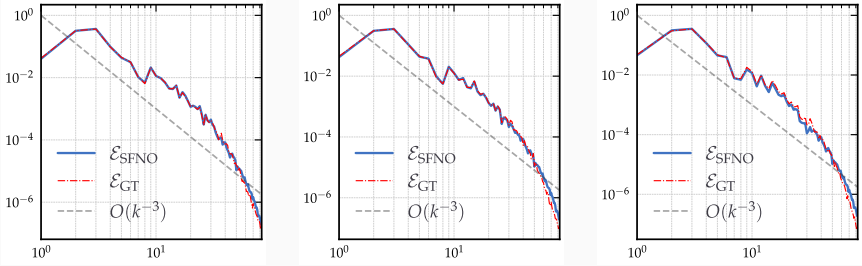
Neural Operators Learn Fast





1 epoch vs **10 epochs**: Enstrophy spectrum density comparisons to illustrate the "convergence" of ST-FNO's training. There are 10 training runs on a 64×64 grid starting from 10 different seeds. The evaluation is on a 256×256 grid for a fixed randomly chosen sample. **The error bars are plotted with $+/-$ 10 times the standard deviation from the mean to boost the visibility.**

Neural Operators Learn Low-Frequency Fast
like really fast but not that accurate



"Spectral bias/frequency principle"[14]: not only neural operators can capture the pattern of the low-frequency modes of the data well, they can also learn the low modes really fast. The plots show the enstrophy spectrum density comparison for a randomly selected trajectory for a given trained SFNO after only 10 epochs. However, the magnitude of the error/difference still dominates in low-frequency for modeling turbulence[15].

[14] J. Zhi-Qin Xu et al. (2020). "Frequency Principle: Fourier Analysis Sheds Light on Deep Neural Networks". In: *Communications in Computational Physics*.

[15] P. Lippe et al. (2023). "PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers". In: *Thirty-seventh Conference on Neural Information Processing Systems*.

Do we really need 500 epochs of training?

Table 1: Benchmarks on Navier Stokes (fixing resolution 64×64 for both training and testing)

Config	Parameters	Time per epoch	$\nu = 1e{-3}$ $T = 50$ $N = 1000$	$\nu = 1e{-4}$ $T = 30$ $N = 1000$	$\nu = 1e{-4}$ $T = 30$ $N = 10000$	$\nu = 1e{-5}$ $T = 20$ $N = 1000$
FNO-3D	$6,558,537$	$38.99s$	**0.0086**	0.1918	**0.0820**	0.1893
FNO-2D	$414,517$	$127.80s$	0.0128	**0.1559**	0.0834	**0.1556**
U-Net	$24,950,491$	$48.67s$	0.0245	0.2051	0.1190	0.1982
TF-Net	$7,451,724$	$47.21s$	0.0225	0.2253	0.1168	0.2268
ResNet	$266,641$	$78.47s$	0.0701	0.2871	0.2311	0.2753

Source: Table 1 from the original FNO paper. FNO3d is trained 500 epochs (500×128 mini-batch ADAM iterations). However, SFNO reaches 1×10^{-2} relative difference evaluated on a 256×256 grid with the ground truth in 5 to 15 epochs, and 6×10^{-3} after 500 epochs. More sophisticated Transformers with multilevel feature aggregation[16] can drive the error down further to 4×10^{-3} level but not any further.

[16] X. Liu, B. Xu, C., and L. Zhang (2024). "Mitigating spectral bias for the multiscale operator learning". In: *Journal of Computational Physics*

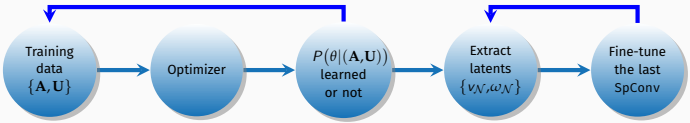
Fine-tuning/Post-processing

Fine-tuning to Improve Accuracy

- Make certain parameters in a spectral convolution layer in FNO correspond to the coefficients of a basis in the spectral domain (the last single-channel spectral conv layer).
- Use a stronger norm to train to learn low-frequency modes (L^2-based loss with aggressive frequency truncation), use a weaker norm to post-process/fine-tune (negative Sobolev norm with less frequency truncation).
  - The frequency truncation in FNO $\approx$ Tikhonov regularization (Hansen, Nagy, and O'Leary 2006).
  - Negative Sobolev norm as an implicit regularization in inverse problems: Osher, Solé, and L. Vese (2003) and Lieu and L. A. Vese (2008), Zhu, Hu, Lou, and Yang (2024).
- The negative Sobolev norm of $f \in L^2(\mathbb{T}^2)/\mathbb{R}$ can be handily computed using FFT, and this negative norm happens to be the correct functional norm to measure the residual: $R(\omega_{\mathcal{N}})(v) := \langle R(\omega_{\mathcal{N}}), v\rangle$, denote $R(\omega_{\mathcal{N}})(t,\cdot) =: r(t,\cdot)$

$$\|r\|_{\mathcal{H}'} \simeq |r|_{-1} := \sum_{\boldsymbol{k}\in 2\pi\mathbb{Z}^2_{\hat{n}}\setminus\{\mathbf{0}\}} |\boldsymbol{k}|^{-2}\left|\hat{r}(\boldsymbol{k})\right|^2.$$

New Paradigm



- Train the neural operator only for a few epochs (e.g., 10) using a strong norm as the loss until it learns the frequency signature of the data (e.g., the energy cascade of the NSE).
- Extract the latent tensor up to the **last** channel-reduction **LINEAR** SpConv layer $Q_\theta(\cdot)$, such that for $i = 1, \cdots, n_t$

$$\text{Neural representation of } \omega^{(t_{\ell+i})} = \omega_{\mathcal{S}}^{(t_\ell)} + Q_\theta(\mathbf{v}_{\text{latent}}).$$

- Remove the frequency truncation and fine-tune this layer Q_θ **ONLY** using an optimizer with a weaker norm.
- $\partial_t \omega_{\mathcal{N}}$ is approximated using an IMEX solver implemented as an `nn.Module`.
- Searching for the best possible θ^* under a functional norm (negative norm) is equivalent to solving a variational problem in the positive Sobolev norm.

New Loss: Residual in Negative Sobolev Norm

Theorem (Reliable and efficient *a posteriori* error estimation)

Under certain smoothness assumptions for $\omega \in \mathcal{H}$, if the fine-tuning problem can be solved exactly, for $m = \ell + 1, \cdots, \ell + n_t - 1$, $\mathcal{T}_m := [t_m, t_{\ell+n_t}]$, the PDE residual

$$R(\omega) := \operatorname{rot} \boldsymbol{f} - \partial_t \omega - (\nabla^\perp(-\Delta)^{-1}\omega \cdot \nabla)\omega + \nu\Delta\omega$$

is efficient in representing the error:

$$\|R(\omega_{\mathcal{N}})\|^2_{L^2(\mathcal{T}_m;\mathcal{V}')} \lesssim \|\omega - \omega_{\mathcal{N}}\|^2_{L^2(\mathcal{T}_m;\mathcal{V})} + \|\partial_t(\omega - \omega_{\mathcal{N}})\|^2_{L^2(\mathcal{T}_m;\mathcal{V}')} + \text{h.o.t.}$$

When $\omega_{\mathcal{N}}$ is sufficiently close to ω,

$$\|\omega - \omega_{\mathcal{N}}\|^2_{L^\infty(\mathcal{T}_m;L^2)} + \|\omega - \omega_{\mathcal{N}}\|^2_{L^2(\mathcal{T}_m;H^1)} \leq \|(\omega - \omega_{\mathcal{N}})(t_m,\cdot)\|^2_{H^1} + C\int_{\mathcal{T}_m} \|R(\omega_{\mathcal{N}})(t,\cdot)\|^2_{\mathcal{V}'}\,\mathrm{d}t.$$

- The optimization of this is not tied to local adaptive mesh refinement, the loss can be simply evaluated globally in the spectral domain to refine the (reduced) spectral basis, similar to ROM (Patera and Rozza 2007).

New Paradigm

(Step 1) Train $G_\Theta := Q_\theta \circ F_{\Theta\setminus\theta}$ using $\sum_{\omega\sim p}\left\{\|G_\Theta(\omega_{\mathcal{S}}) - \omega_{\mathcal{S}}\|_0^2 + \gamma\|\Theta\|_0^2\right\}$.

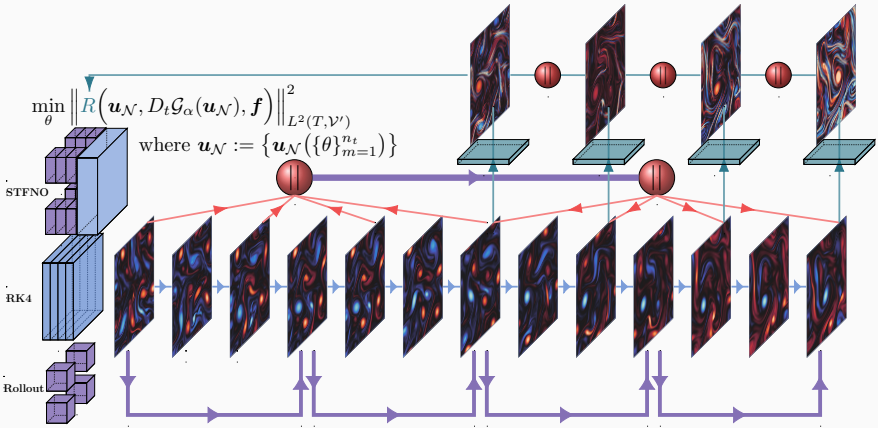
(Step 2) Stop training after the relative difference reached a threshold on the validation sets.

(Step 3) Fine-tune (post-process) the **last** layer Q_θ where $\theta_{\text{train}} \in \mathbb{C}^{1\times H\times d\times d\times d_t}$ and $\theta_{\text{ft}} \in \mathbb{C}^{1\times(H+1)\times\frac{2n}{3}\times\frac{2n}{3}\times n_t}$ for n the number of grid points in each axis of an evaluation sample, n_t evaluation time steps (e.g., 40).

- Approximating the time derivative by $\psi^{(m)} = (-\Delta_h)^{-1}\omega^{(m)}$ with $\{\omega^{(m)}\}$ being the output of G_θ, and
$$\frac{3\omega^{(m+1)} - 4\omega^{(m)} + \omega^{(m-1)}}{2\,(\delta t)} + \nabla^\perp(2\psi^{(m)} - \psi^{(m-1)})\cdot\nabla\left(2\omega^{(m)} - \omega^{(m-1)}\right) - \nu\Delta\omega^{(m+1)} = f^{(m+1)}.$$
- Apply an optimizer to find θ_{ft} by minimizing $\sum_{m\in\{\ell+1,\ldots,\ell+n_t\}}\left\|R(\omega^{(m)}(\theta_{\text{ft}}))\right\|_{-1}^2$.

(Output) A trajectory that satisfies the statistical signature and minimizes the residual while not having to march $\mathcal{O}(1/\tau)$ steps.

New Paradigm



Schematics comparison: the new method a shares striking mathematically resemblance with a parallel-in-time two-grid method using a learned reduced basis. On a very “coarse” temporal grid, the spatiotemporal surrogate model gives a very good “initial” guess to perform implicit residual smoothing.

Numerical Experiments

Taylor-Green Vortex

$$\boldsymbol{u}(t,x,y) = e^{-2\kappa^2\nu t}\begin{pmatrix}\sin(\kappa x)\cos(\kappa y)\\ -\cos(\kappa x)\sin(\kappa y)\end{pmatrix} \text{ with } \nu = 10^{-2}.$$

- Training dataset: 10 samples with $\kappa = 1,\ldots,10$; evaluation: 1 sample with $\kappa = 11$.
- Trajectories are randomly sampled before the breakdown phase.
- Ground truth: pseudo-spectral discretization, second-order Runge-Kutta for the explicit part, Crank-Nicolson for diffusion part with $\Delta t = 4\times 10^{-3}$.
- Training is done on 64×64 for 5 epochs, the evaluation is on 256×256, time step is 10 for training, 40 for evaluation. ST-FNO's parameters have capacity to represent the discrete solutions in the eval set exactly.

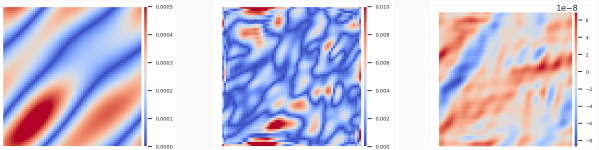
Results for Taylor-Green vortex example $\varepsilon := \omega_{\text{true}} - \omega_{\mathcal{N}}$, the errors at the final time step.

	Evaluation after training		After fine-tuning	
	$\|\varepsilon\|_{L^2}$	$\|R\|_{-1,n}$	$\|\varepsilon\|_{L^2}$	$\|R\|_{-1,n}$
FNO3d	1.84×10^{-1}	5.40×10^{-1}	N/A	N/A
ST-FNO3d	1.94×10^{-1}	2.18×10^{-1}	1.24×10^{-6}	3.21×10^{-7}
PS+RK2 (GT)	5.91×10^{-6}	1.16×10^{-5}	N/A	N/A

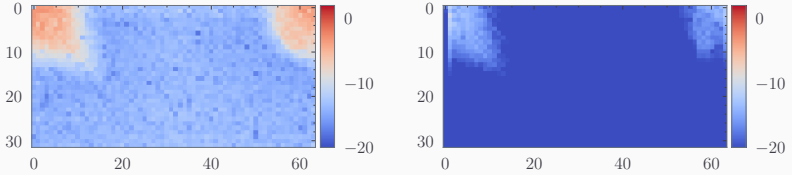
Forced Turbulence: FNO Data

Results for forced turbulence (original toy model example from the FNO paper, $\nu = 1\times10^{-3}$).

	Evaluation after training		After fine-tuning	
	$\|\varepsilon\|_{L^2}$	$\|R\|_{-1,n}$	$\|\varepsilon\|_{L^2}$	$\|R\|_{-1,n}$
FNO 100 ep	1.31×10^{-2}	1.30×10^{-2}	N/A	N/A
ST-FNO 10 ep + L^2 FT	1.02×10^{-2}	1.27×10^{-2}	2.82×10^{-4}	2.78×10^{-5}
ST-FNO 10 ep + H^{-1} FT	–	–	3.16×10^{-4}	4.59×10^{-7}

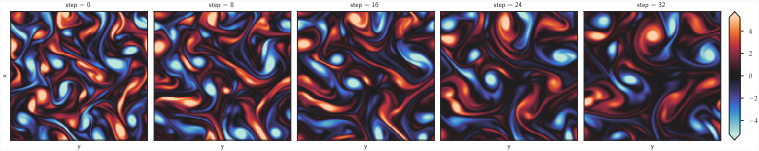


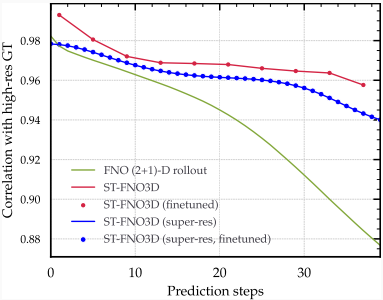
Pointwise residuals for the ground truth streamfunction, ST-FNO inference, and ST-FNO inference after fine-tuning.

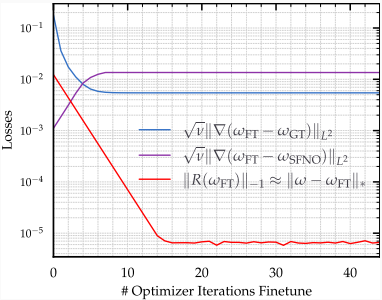


(Left) the log of $|R|$ in the frequency after training; (right) after fine-tuning.

Isotropic Turbulence: Direct Cascade (McWilliams 1984)



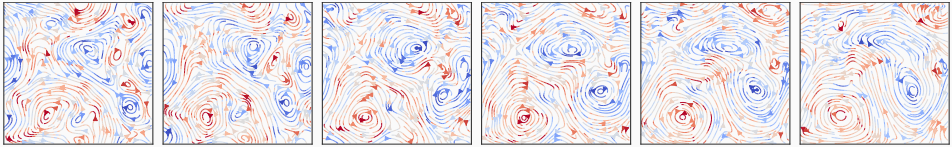




(Left) the correlation with the highest-resolution DNS ground truth, one step $= 55\Delta t$. (Right) Fine-tuning convergence.

Torch-CFD

github.com/scaomath/torch-cfd



A native PyTorch port of Google Research's Jax-CFD (Jax-CFD has not been in active development since Aug 2023) with many new and enhanced features.

- Fully implemented as "tensor-in-tensor-out" with a batch dimension `(b, c, *)`, more user-friendly for tensor-tracking debugger (VSCode).
- Gridded variables implemented natively as a subclass of `torch.Tensor` with their ops using `__torch_function__`, perfect for staggered grids.
- Modular designs: time-marching schemes, multigrid solvers, different flux schemes, also implemented as `nn.Module` with learnable components (Butcher's tableau, smoothers, weights for fluxes interpolations). Taking advantage of handy methods such as `_forward_hooks` and `_backward_hooks`.

Traditional Numerical Schemes as Neural Networks

```python
class MultigridSolver(nn.Module):
  def __init__(self) -> None:
    super().__init__()
    self.smoothers: nn.ModuleList = ...
    self.operators: nn.ModuleList = ...

  def forward(self, f, u, lvl):
      u = self.smoothers[lvl](f, u)
      r = f - self._apply_op(u, self.operators[lvl])
      if lvl == self.levels - 1:
          e = self._coarse_solve(r)
      else:
          rc = self.restrict(r)
          ec = self.forward(rc, 0, lvl=lvl + 1)
          e = self.prolong(ec)
      u += e
      return self.smoothers[lvl](f, u)
```

Acknowledgments

- Organizers: Jack Xin (UC Irvine), Gitta Kutyniok (LMU Munich), Stanley Osher (UCLA), Bao Wang (University of Utah), Andrea Bertozzi (UCLA).
- Co-authors, as well as Long Chen (UC Irvine), Ludmil Zikatanov (back at Penn State), Ari Stern (WashU).
- Source codes and data to replicate the experiments are available at
  github.com/scaomath/torch-cfd/examples 10.57967/hf/2470
- References
  - C., F. Brarda, R. Li, and Y. Xi (2025). "Spectral-Refiner: Accurate Fine-Tuning of Spatiotemporal Fourier Neural Operator for Turbulent Flows". In: *The Thirteenth International Conference on Learning Representations (ICLR)* cs.LG:2405.17211.
- This research is supported in part by NSF award DMS-2309778.