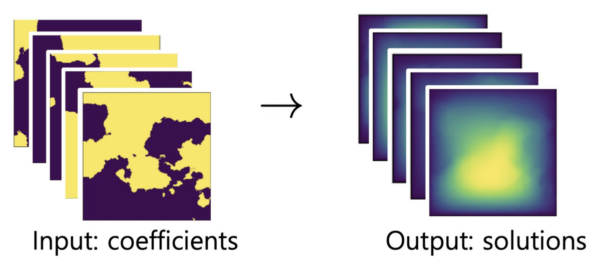
NODE-FNO And C-FNO: Improve Fourier Neural Operators through Adjoint Training and Conservation Enforcing

Kai Chang, Department of Mathematics, Emory University

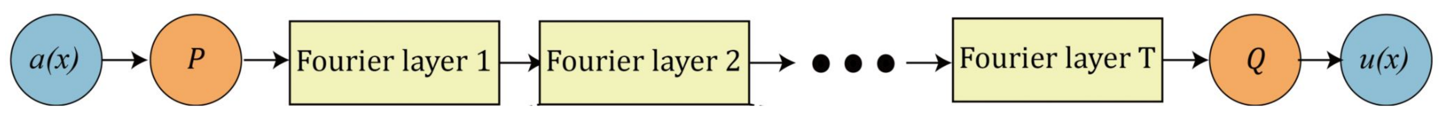
The University of Texas at Austin
Oden Institute for Computational Engineering and Sciences

Fourier Neural Operators

- FNOs approximate mappings between infinite-dimensional spaces.
- Used for solving parametric PDEs.

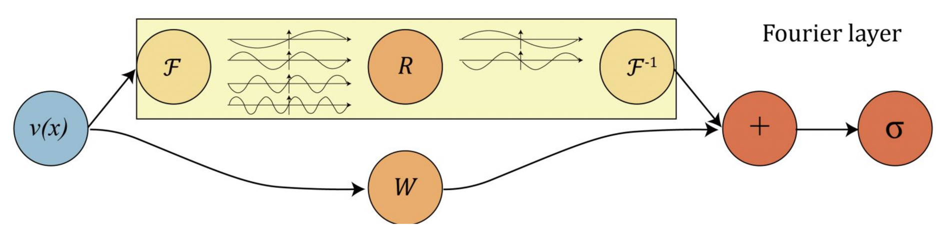


- Main Components of an FNO: encoder, Fourier layers, decoder



- The updating formula for Fourier layers:

$$v_{l+1}(x) = \sigma\left(W(v_l(x)) + \mathcal{F}^{-1}(\boldsymbol{R}_l \cdot \mathcal{F}(v_l(x)))\right)$$



Issues with Vanilla FNOs

- Naive Training. High memory cost.
- Barely take any physical information into consideration.

Solution: NODE-FNO and C-FNO

NODE-FNOs

- Modify the Fourier layer formula as

$$v_{l+1}(x) = v_l(x) + \sigma\left(W(v_l(x)) + \mathcal{F}^{-1}(\boldsymbol{R}_l \cdot \mathcal{F}(v_l(x)))\right)$$

- This can be viewed as Euler discretization of the ODE

$$\frac{dv(t,x)}{dt} = \sigma\left(W(v(t,x)) + \mathcal{F}^{-1}(\boldsymbol{R} \cdot \mathcal{F}(v(t,x)))\right)$$

- Train it as a **Neural ODE**

C-FNOs

- Common FNO loss: $\mathcal{L}_{FNO} := \frac{\|\mathcal{NN}_\theta(a) - u\|_{L^p}}{\|u\|_{L^p}}$.
- Conservation law of 1-D Burgers' equation:

$$\int_0^1 u(x,t)dx = \text{constant } \forall t$$

- Define $\mathcal{L}_C := \sum_{k=1}^{N_s} \left(\int_0^1 [\mathcal{NN}_\theta(a)|_{t=1} - \mathcal{NN}_\theta(a)|_{t=t_k})] \, dx\right)^2$.
- **C-FNO Loss:**

$$\mathcal{L}_{CFNO} = \mathcal{L}_{FNO} + \mathcal{L}_C$$

1. #Training: 1000. #Testing: 200. Initial learning rate: 0.01. Learning rate decay: 0.5/50 epochs.

Benchmarks

1-D Burgers' Equation

$$\partial_t u(x,t) + \partial_x \left(u^2(x,t)/2\right) = \nu \partial_{xx} u(x,t)$$
$$u(x,0) = u_0(x)$$
$$t \in [0,1] \; x \in [0,1]$$

Goal: learn a map

$$\mathcal{NN}_\theta : u_0 \mapsto u|_{t=1}$$

2-D Darcy Flow

$$-\nabla \cdot (a(x)\nabla u(x)) = f(x)$$
$$BC: \; u_b(x) = 0$$
$$x \in (0,1) \times (0,1)$$

Goal: learn a map

$$\mathcal{NN}_\theta : a \mapsto u$$

2-D Navier-Stokes

$$\partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) = \nu \Delta w(x,t) + f(x)$$
$$\nabla \cdot u(x,t) = 0$$
$$w(x,0) = w_0(x)$$

Goal: learn a map

$$\mathcal{NN}_\theta : w|_{t_1:t_{10}} \mapsto w|_{t_{11}:t_{50}}$$

1-D Burgers' Equation

$$\partial_t u(x,t) + \partial_x \left(u^2(x,t)/2\right) = \nu \partial_{xx} u(x,t)$$
$$u(x,0) = u_0(x)$$
$$t \in [0,1] \; x \in [0,1]$$

Goal: learn a map

$$\mathcal{NN}_\theta : u_0 \mapsto u$$

Results[1]

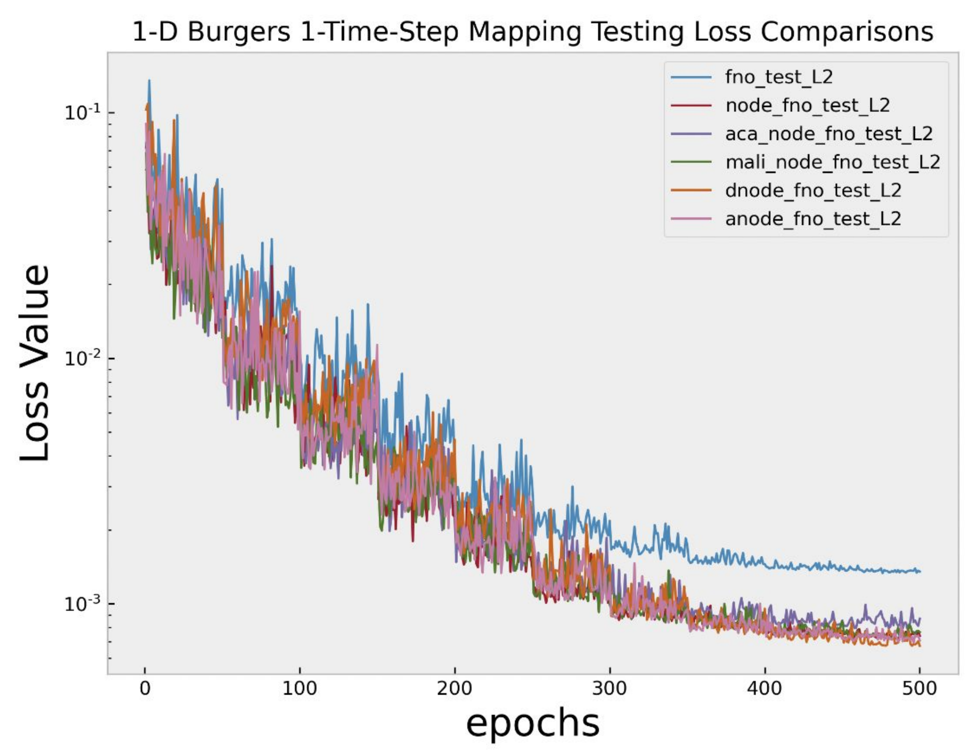


Figure 1: 1-D Burgers' 1-Step Mapping. Training discretization: 512. Testing discretization: 2048. Performance: errors are 40.8–49.4% lower than FNO.

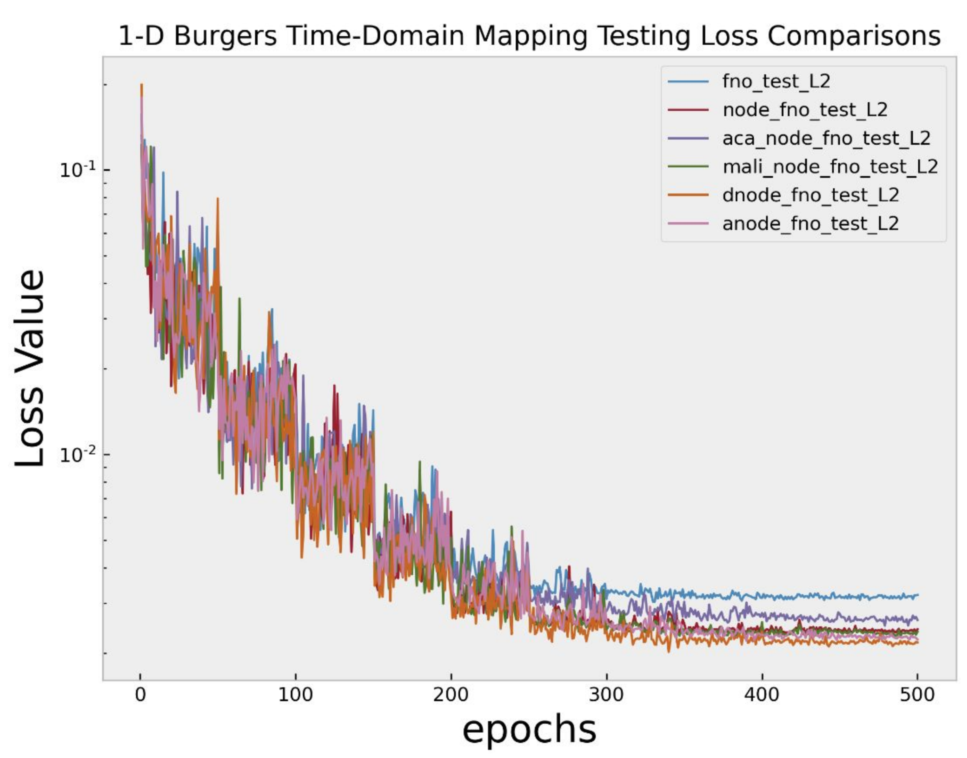


Figure 2: 1-D Burgers' whole time domain mapping. Training discretization size: 51×64. Testing discretization size: 101×128. Performance: errors are 17.5–33.3% lower than FNO.

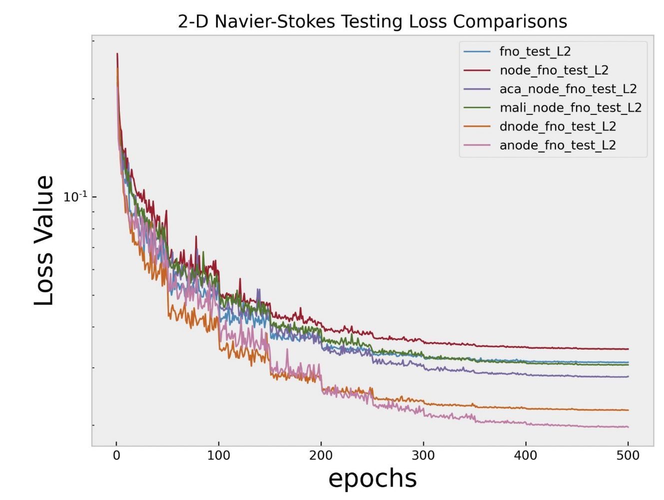


Figure 4: 2-D Navier-Stokes. Training discretization size: 64×64. Testing discretization size: 64×64. Performance: reducing the generalization error by 1.8–36.7%.

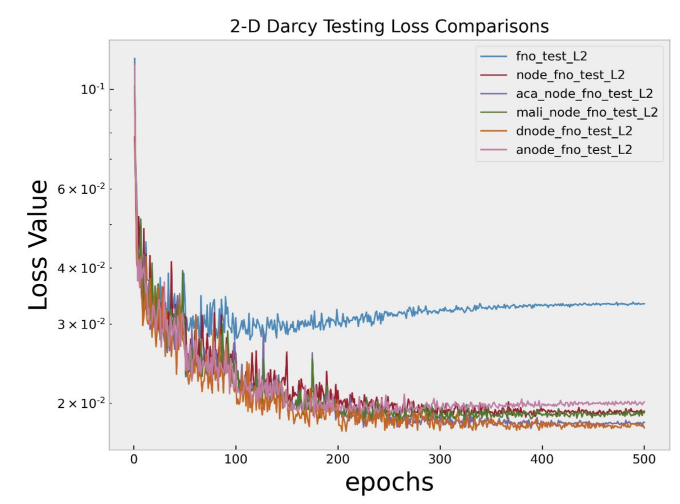


Figure 3: 2-D Darcy Flow. Training discretization size: 85×85. Testing discretization size: 421×421. Performance: 32.1–36.8% lower than FNO.

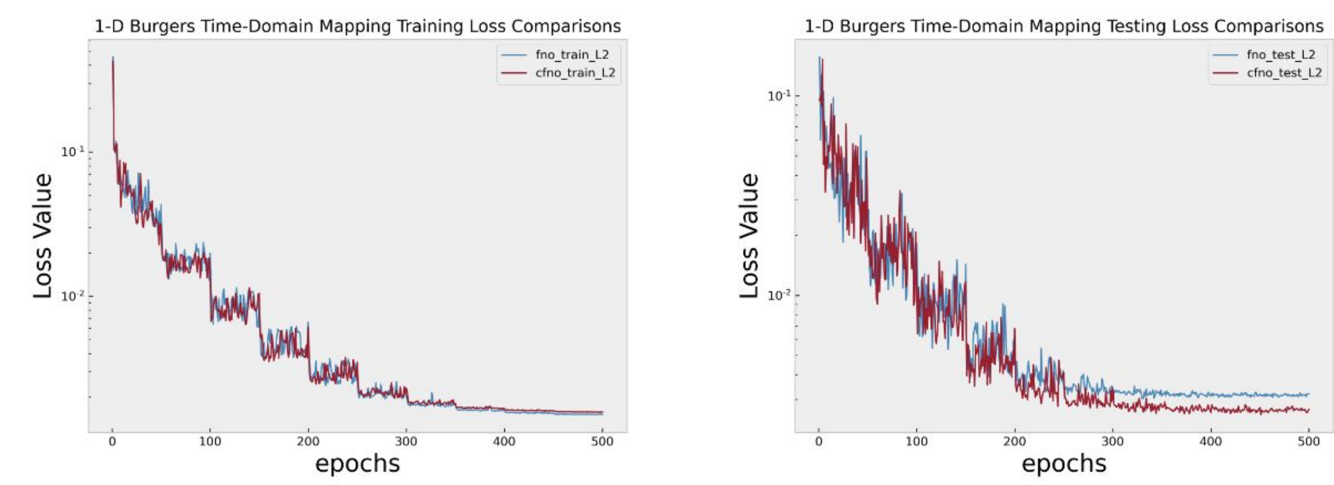


Figure 5: 1-D Burgers' whole time domain mapping. Training discretization size: 51×64. Testing discretization size: 101×128. Performance: the error is 22% lower than FNO.

Conclusion

- NODE-FNOs **beat** the vanilla FNO **on all benchmarks.**
- NODE-FNOs have **lower** memory cost.
- dNODE-FNO in general has good performance and requires less time to train. But more memory cost.
- C-FNO **generalizes better** than the vanilla FNO on 1-D Burgers' equation with full time domain mapping.

Acknowledgments

Thank Dr. Chandrajit Bajaj for his consistent support and guidance throughout the project. Thank Luke McLennan for valuable discussions and collaboration. Thank the Moncrief program for providing research funding.

References

1. Chen, Ricky TQ, et al. "Neural ordinary differential equations." *Advances in neural information processing systems* 31 (2018).
2. Li, Zongyi, et al. "Fourier neural operator for parametric partial differential equations." *arXiv preprint arXiv:2010.08895* (2020).
3. Dupont, Emilien, Arnaud Doucet, and Yee Whye Teh. "Augmented neural odes." *Advances in Neural Information Processing Systems* 32 (2019).
4. Zhuang, Juntang, et al. "MALI: A memory efficient and reverse accurate integrator for Neural ODEs." *arXiv preprint arXiv:2102.04668* (2021).
5. Zhuang, Juntang, et al. "Adaptive checkpoint adjoint method for gradient estimation in neural ode." *International Conference on Machine Learning*. PMLR, 2020.
6. Li, Zongyi, et al. "Physics-informed neural operator for learning partial differential equations." *arXiv preprint arXiv:2111.03794* (2021).
7. Li, Zongyi, et al. "Multipole graph neural operator for parametric partial differential equations." *Advances in Neural Information Processing Systems* 33 (2020): 6755-6766.
8. Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).
9. Li, Zongyi, et al. "Fourier Neural Operator with Learned Deformations for PDEs on General Geometries." *arXiv preprint arXiv:2207.05209* (2022).
10. Jagtap, Ameya D., Ehsan Kharazmi, and George Em Karniadakis. "Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems." *Computer Methods in Applied Mechanics and Engineering* 365 (2020): 113028.