Neural Networks with Cheap Differential Operators

Ricky T. Q. Chen, David Duvenaud

UNIVERSITY OF TORONTO

VECTOR INSTITUTE

Differential Operators

- Want to compute operators such as *divergence:*

$$\nabla \cdot f = \sum_{i=1}^{d} \frac{\partial f_i(x)}{\partial x_i}$$

where $f : \mathbb{R}^d \to \mathbb{R}^d$ is a neural net.

- Solving PDEs
- Finding fixed points
- Fitting SDEs
- Continuous normalizing flows

Automatic Differentiation (AD)

Reverse-mode AD gives cheap vector-Jacobian products:

$$v^T \left[\frac{d}{dx} f(x)\right] = \sum_{i=1}^{d} v_i \frac{\partial f_i(x)}{\partial x} = \begin{bmatrix} \text{———} & v_1 \frac{\partial f_1(x)}{\partial x} & \text{———} \\ & \vdots & \\ \text{———} & v_d \frac{\partial f_d(x)}{\partial x} & \text{———} \end{bmatrix}$$

- For full Jacobian, need d separate passes
- In general, Jacobian diagonal has **the same cost as the full Jacobian!**
- We restrict architecture to allow one-pass diagonal computations.

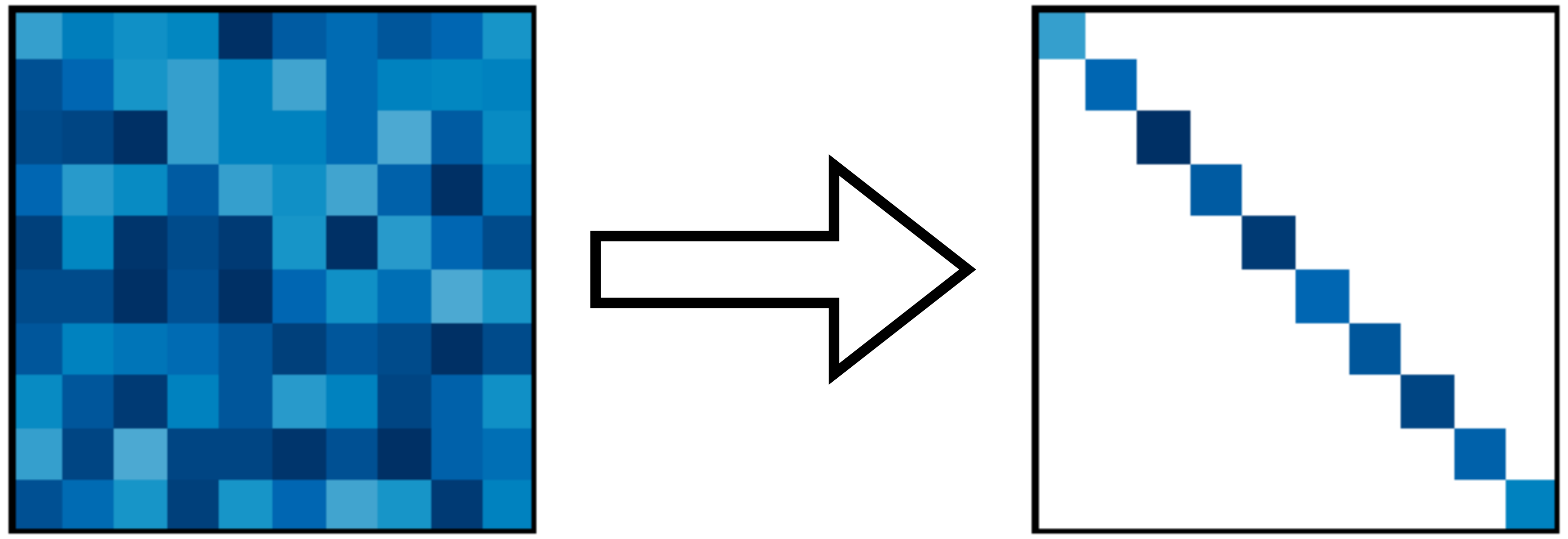
HollowNets

Allow efficient computation of *dimension-wise derivatives* of order k:

$$\mathcal{D}^k_{\mathrm{dim}} f := \left[\frac{\partial^k f_1(x)}{\partial x_1^k} \quad \frac{\partial^k f_2(x)}{\partial x_2^k} \quad \cdots \quad \frac{\partial^k f_d(x)}{\partial x_d^k} \right]^T \in \mathbb{R}^d$$

with only k backward passes, regardless of dimension.

Example:



Jacobian

$D^{k=1}_{dim} f(x)$ = Jacobian diagonal

HollowNet Architecture

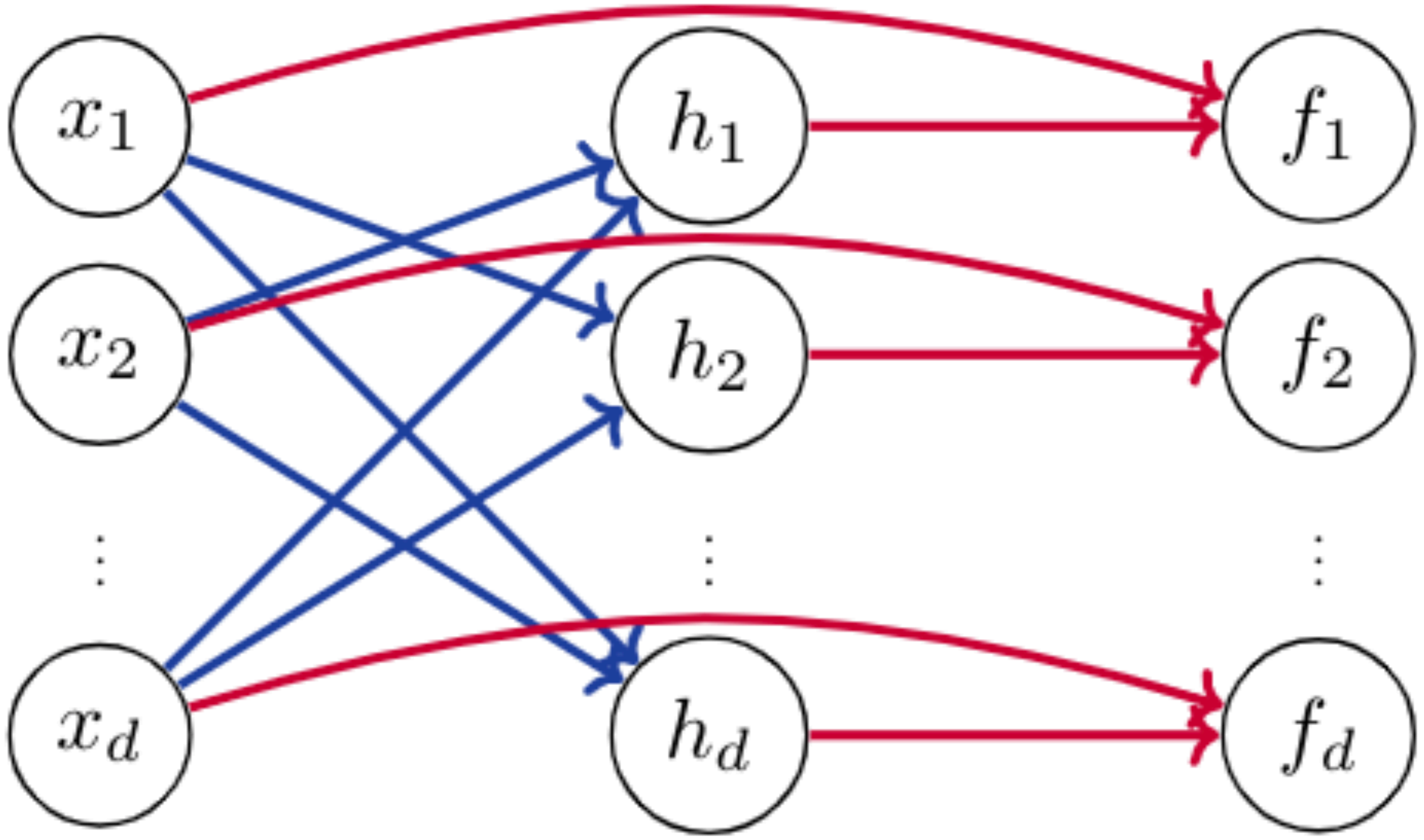
HollowNets are composed of two sub-networks:

- Hidden units which **don't depend** on their respective input:

$$h_i = c_i(x_{-i})$$

- Output units **depend only** on their respective hidden and input:

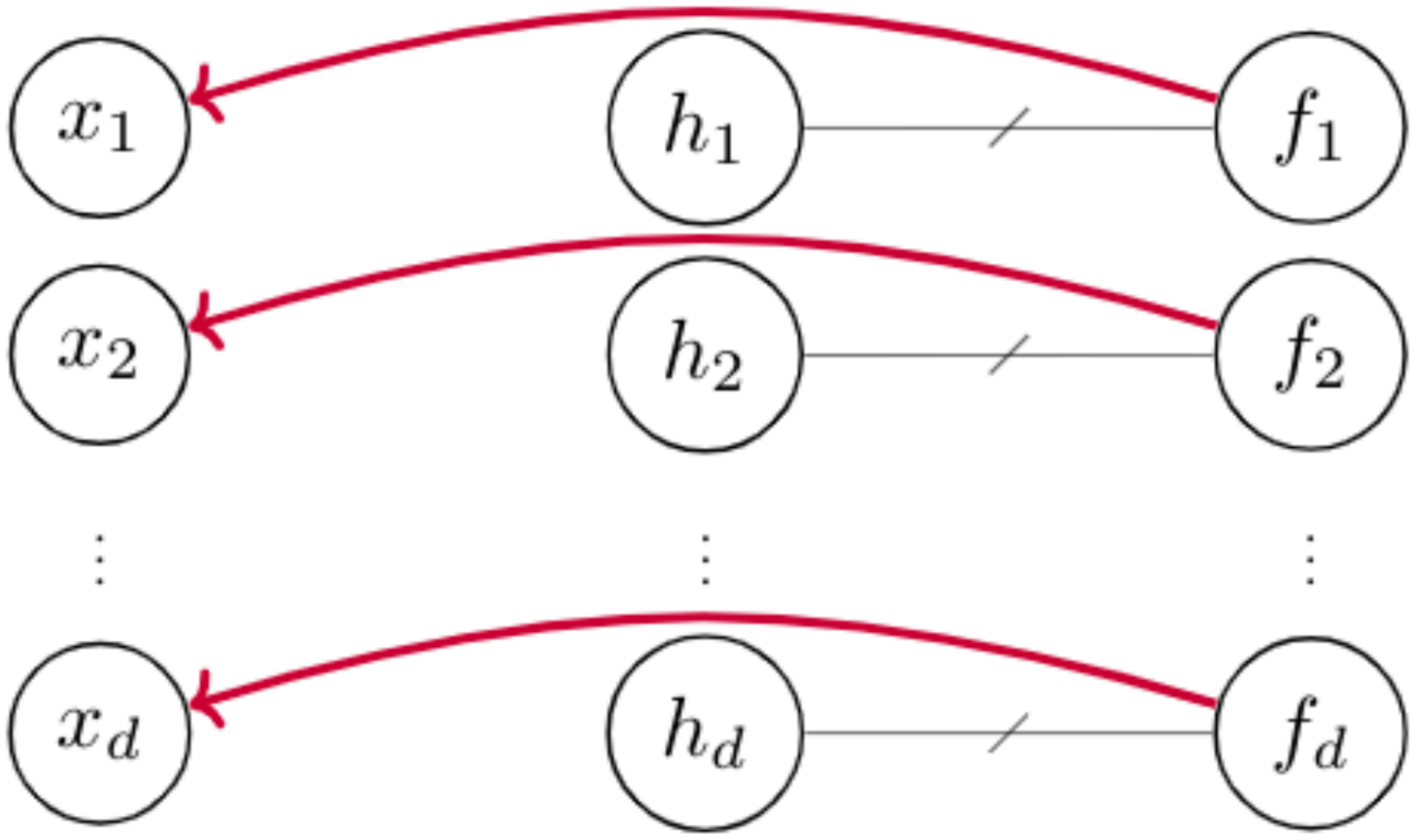
$$f_i(x) = \tau_i([x_i, h_i])$$



HollowNet Jacobians

Can get exact dimension-wise derivatives by **disconnecting** some dependencies in backward pass.

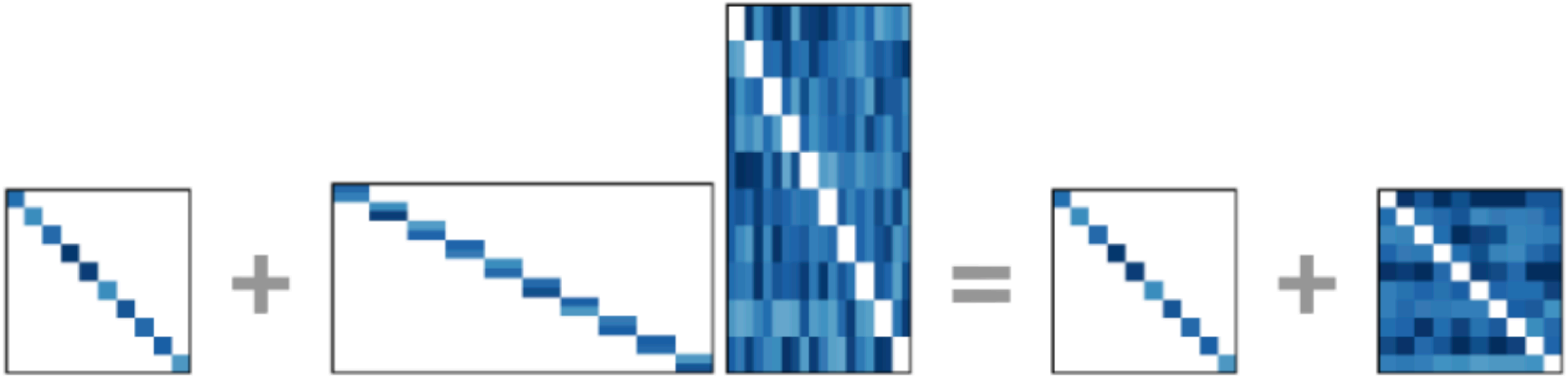
i.e. `detach` in PyTorch or `stop_gradient` in TensorFlow.



HollowNet Jacobians

Can factor Jacobian into:

- A **diagonal** matrix (dimension-wise dependencies).
- A **hollow** matrix (all interactions).



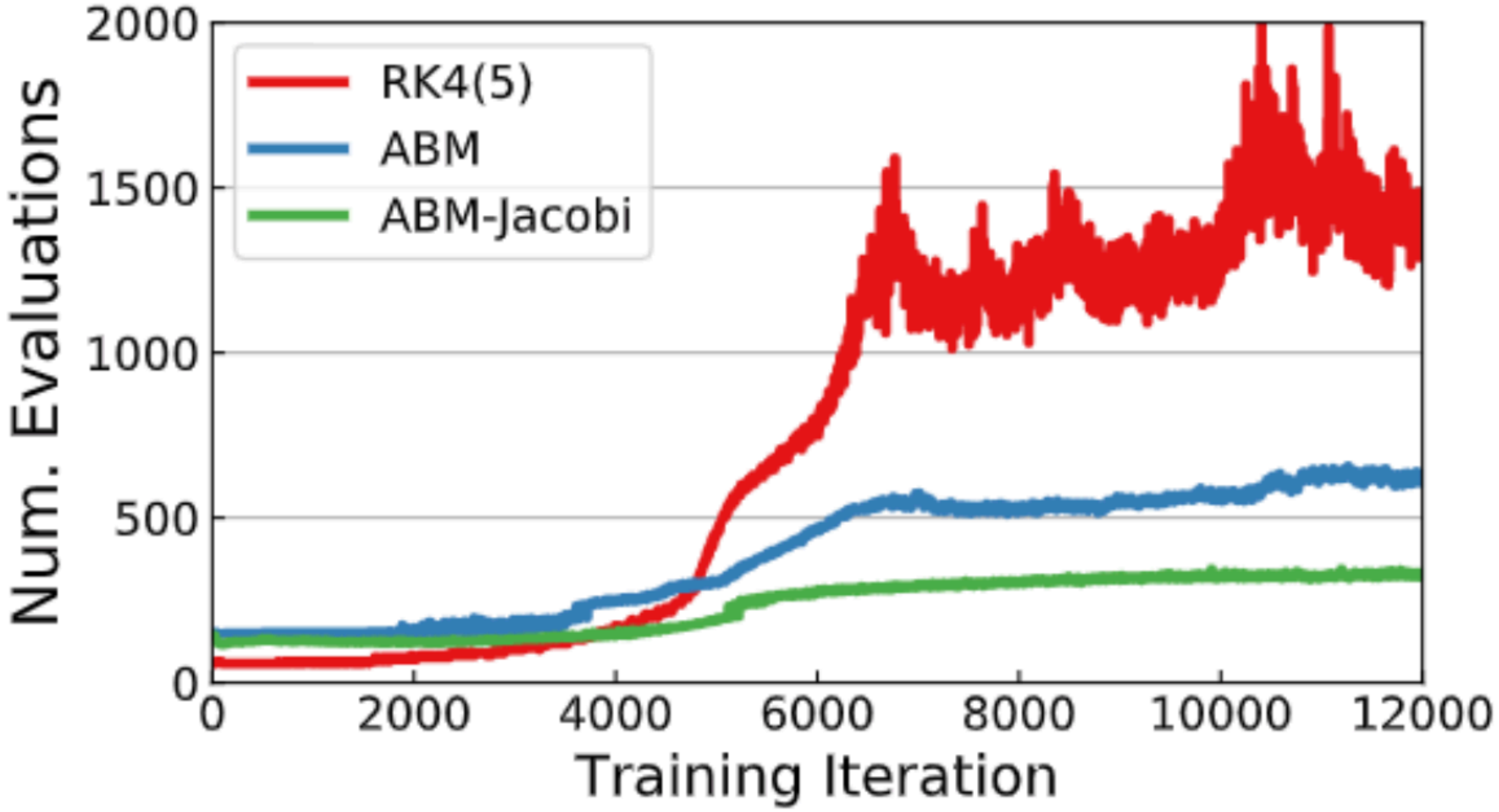
$$\frac{d}{dx}f = \frac{\partial}{\partial x}\tau(x,h) + \frac{\partial}{\partial h}\tau(x,h)\frac{\partial}{\partial x}h(x) = \text{diagonal} + \text{hollow}$$

Application I: Finding Fixed Points

Root finding problems $(f(x) = 0)$ can be solved using Jacobi-Newton:

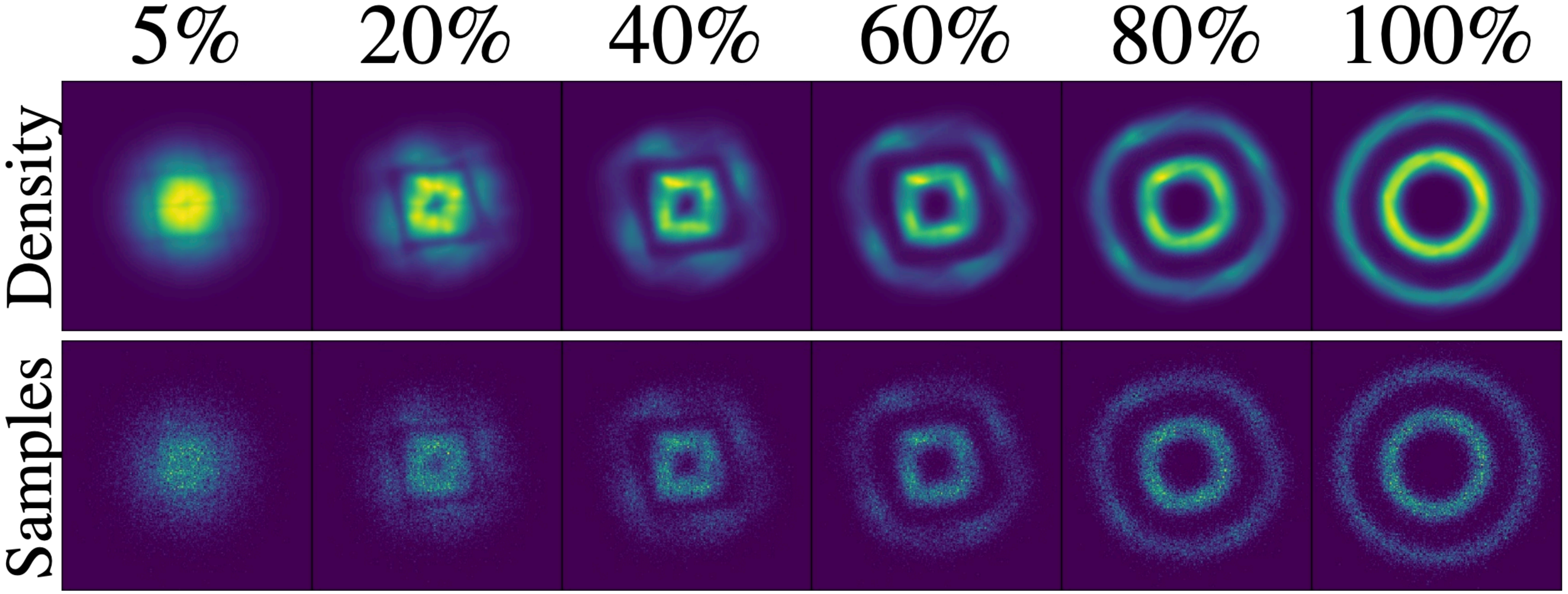
$$x_{t+1} = x_t - f(x) \qquad x_{t+1} = x_t - \left[D_{dim} f(x)\right]^{-1} f(x)$$

- Same solution with **faster convergence.**
- We applied to implicit ODE solvers for solving stiff equations.



Application II: Continuous Normalizing Flows

- Transforms distributions through an ODE:



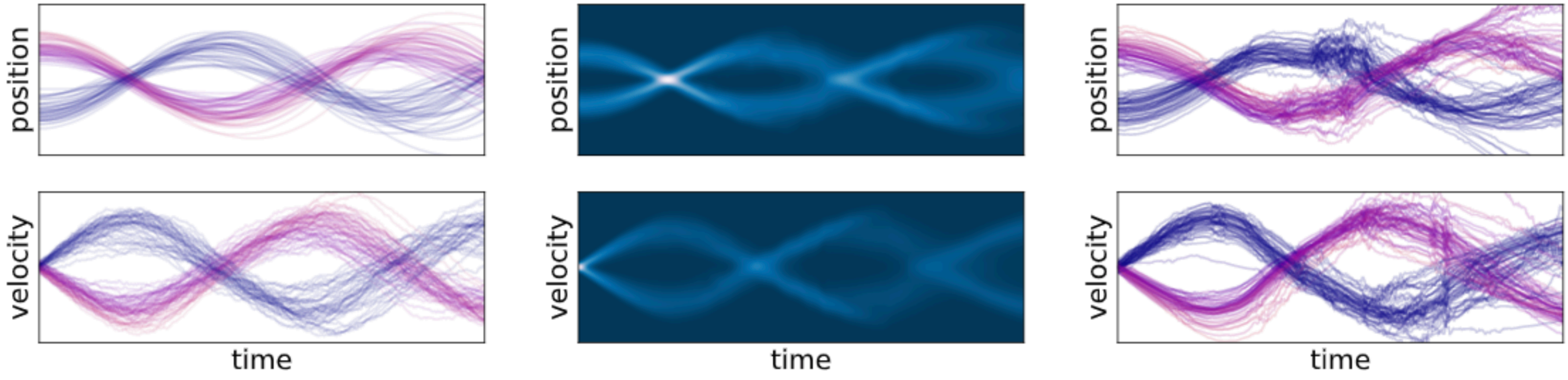
- Change in density given by divergence:

$$\frac{d \log p(x,t)}{dt} = \mathrm{tr}\left(\frac{d}{dx} f(x)\right) = \sum_{i=1}^{d} \left[D_{dim} f(x)\right]_i$$

Learning Stochastic Diff Eqs

- Fokker-Planck describes density change using D_{dim} and D^2_{dim} :

$$\frac{\partial p(t,x)}{\partial t} = \sum_{i=1}^{d} \left[-(\mathbf{D_{dim}} f)p - (\nabla p) \odot f + (\mathbf{D^2_{dim}} diag(g))p + 2(\mathbf{D_{dim}} diag(g)) \odot (\nabla p) + \frac{1}{2}\, diag(g)^2 \odot (\mathbf{D_{dim}} \nabla p) \right]_i$$



Data

Learned Density

Samples from Learned SDE

Takeaways

- Dimension-wise derivatives are costly for general functions.
- Restricting to hollow Jacobians gives cheap diagonal grads.
- Useful for PDEs, SDEs, normalizing flows, and optimization.