Sparse-Flows: Pruning Continuous-depth Model

Lucas Liebenwein*, Ramin Hasani*, Alexander Amini, Daniela Rus,

MIT CSAIL * Equal Contributions

I) Motivation

Neural ODEs (Chen et al. 2018) (Neural Ordinary Differential Equations)

$$\frac{\partial z(t)}{\partial t} = f(\mathbf{z}(t), t, \theta),$$
where $\mathbf{z}(t_0) = \mathbf{z}_0$

CNFs (Continuous Normalizing Flows) (Chen et al. 2018)

Base distribution: $\mathbf{z}_0 \sim p_{\mathbf{z}_0}(\mathbf{z}_0)$, Target distribution: $\mathbf{z}_n \sim p_{\mathbf{z}_n}(\mathbf{z}_n)$

Change of variable:

$$\log p(\mathbf{z}(t_n)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathrm{Tr}\left(\frac{\partial f}{\partial \mathbf{z}(t)}\right) dt$$

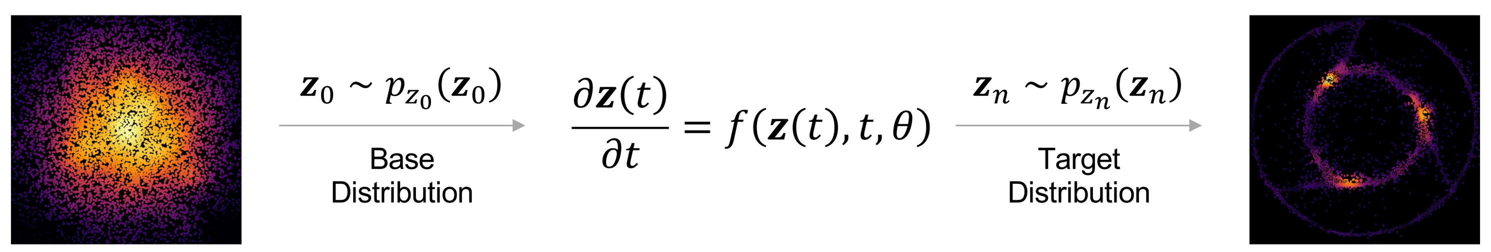
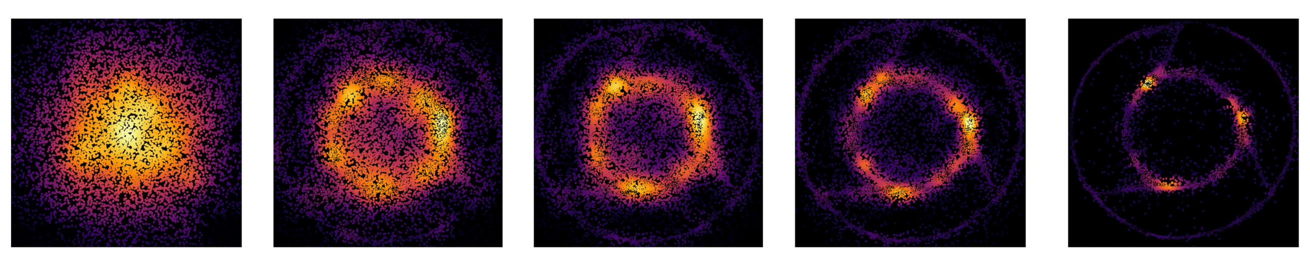


Image credit: Torchdyn code repository, https://github.com/DiffEqML/torchdyn

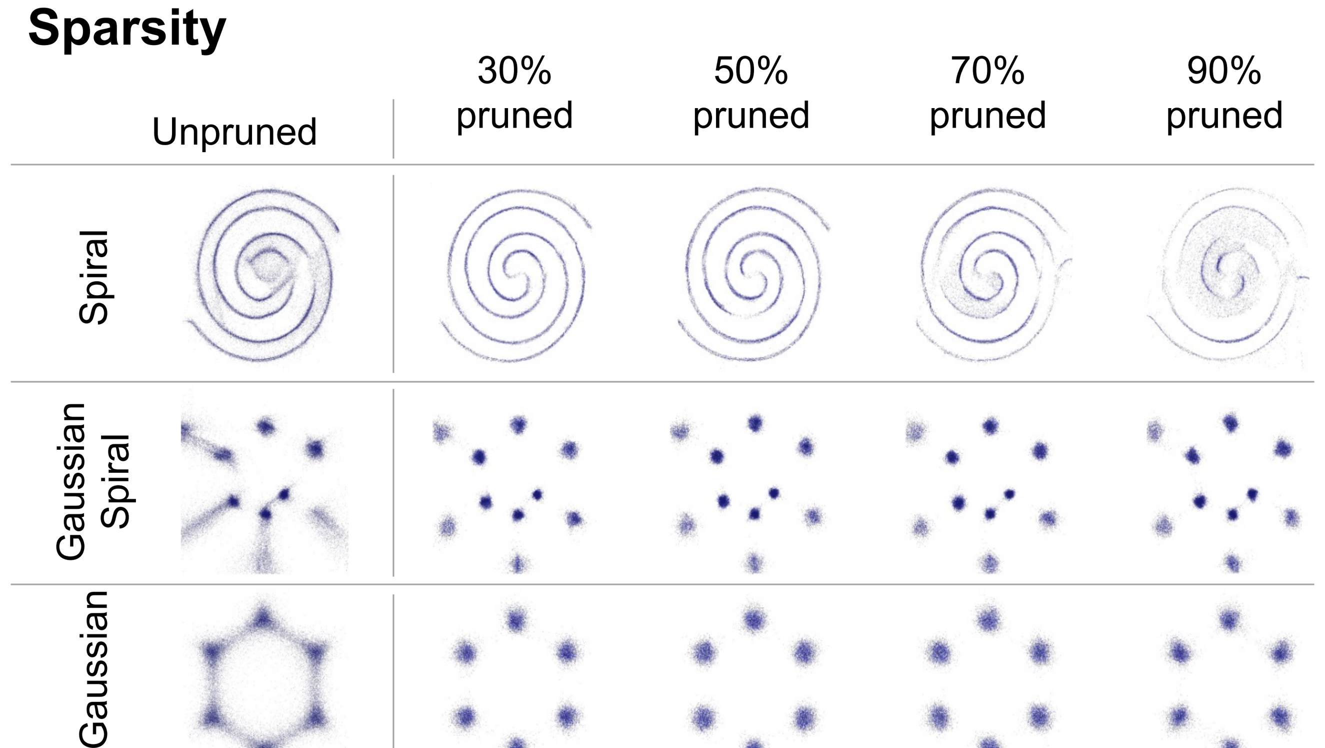


We can train CNFs by directly minimizing the negative log likelihood loss function, as long as the neural network f in the neural ODE is Lipschitz continuous.

This way we have access to the distribution at any given point during the transformation.

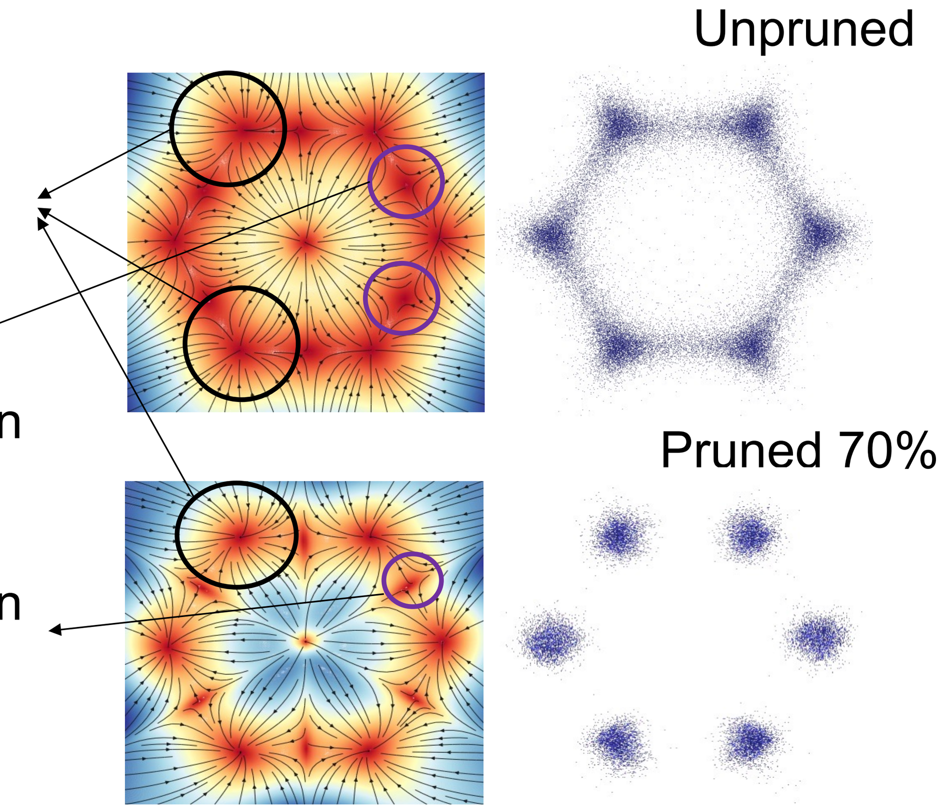
Objective

Understand Generalization Properties of CNFs using Sparsity



II) Sparsity Helps Avoid Mode-Collapse

- ✓ Vector field in this black region (that corresponds to an actual mode), does attract all samples inward toward that specific mode.
- ✓ Vector field in this purple region (which is in-between modes) attract points.
- ✓ Vector field in this purple region (which is in-between modes) **DOES NOT** attract points. Arrows direct samples to the actual models in the dataset



Measure of mode-collapse

The percentage of good quality samples [Srivastava et al. 2017]

I. Draw samples from a trained normalizing flow
II. A sample is of good quality if is within n (e.g., 2, 3 or 5) std from its nearest mode
III. Report the % of good samples as a measure of how well the generative model captures modes

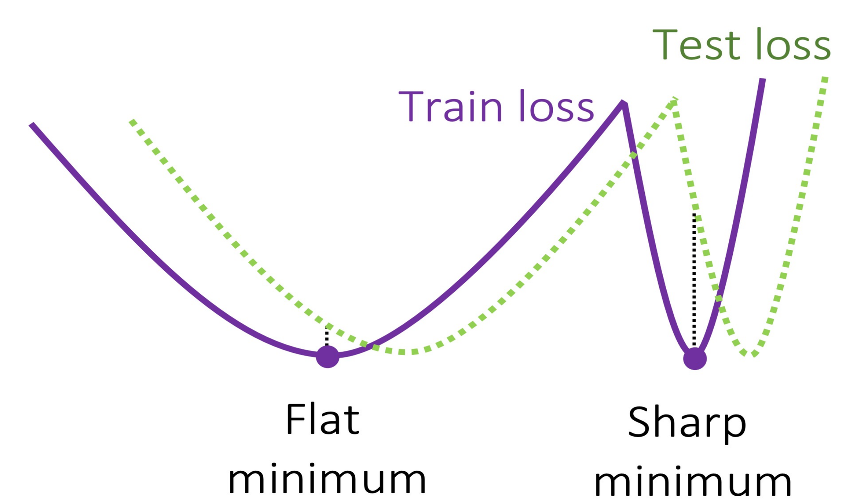
	% of Good Quality Samples	
Prune ratio (%)	Std = 2	std = 3
0.0	72.89%	89.34%
20.0	79.46%	93.45%
25.6	81.40%	94.90%
52.0	84.30%	96.85%
71.2	81.83%	94.86%
75.1	68.20%	86.10%
98.0	36.01%	76.54%

III) Sparsity Helps avoid Sharp Minima

Why pruning helps generalization?

Let's do an empirical Hessian-based investigation on the objective function of the normalizing flows in density estimation

For Neural ODEs, pruning decreases the value of the Hessian's eigenvalues, and as a result, flattens the loss which leads to better generalization [Keskar et al. (2017)].



We used PyHessian [Yao et al. 2020] to analyze the Hessian H w.r.t. the parameters of the CNF.

Inspired by the Hessian analysis in [Erichson et al. 2021]:

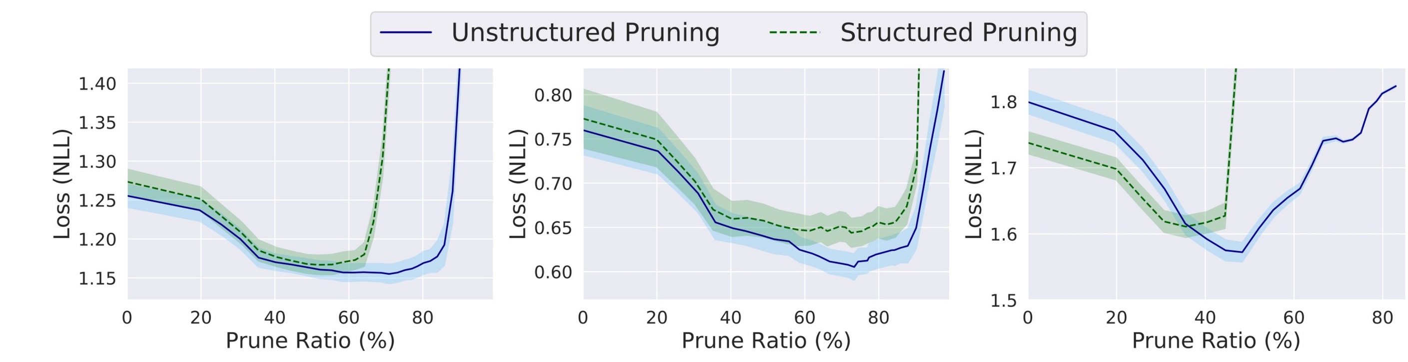
I. Compute maximum eigenvalue $\lambda_{max}(H)$
II. Hessian's Trace $\mathrm{tr}(H)$
III. Condition number $\kappa(H) = \frac{\lambda_{max}}{\lambda_{min}}$

- ✓ Smaller $\lambda_{max}(H)$ and $\mathrm{tr}(H)$ → flatter local minima
- ✓ Smaller κ → more robust network [Bottou and Bousquest, 2008]

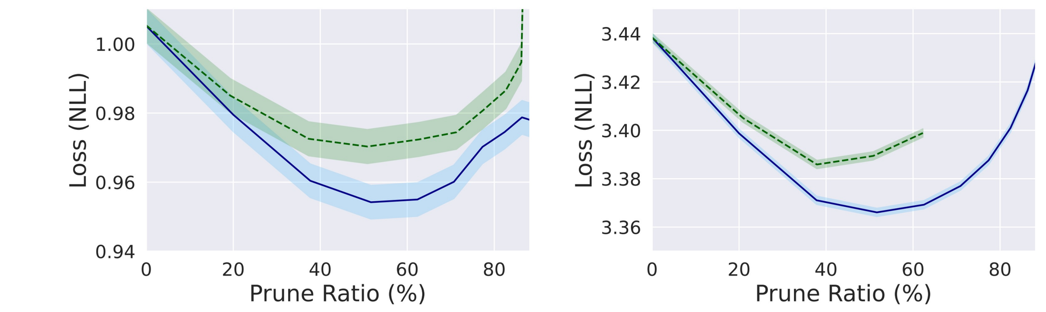
Gaussians-Spiral - Hessian Analysis (Structured Pruning)

Model	NLL	$\lambda_{max}(H)$	$\mathrm{tr}(H)$	$\kappa(H)$
Unpruned FFJORD	0.880	0.0130	0.121	0.34k
Sparse Flows(PR=25%)	0.692	0.0076	0.058	0.76k
Sparse Flows(PR=48%)	**0.634**	**0.0049**	0.047	0.22k
Sparse Flows(PR=67%)	**0.646**	**0.0052**	0.051	0.75k
Sparse Flows(PR=82%)	**0.657**	**0.0053**	0.053	1.69k
Sparse Flows(PR=94%)	0.740	0.0086	0.070	0.11k
Sparse Flows(PR=96%)	0.986	0.0100	0.095	0.23k

IV) Preview of Experimental Results



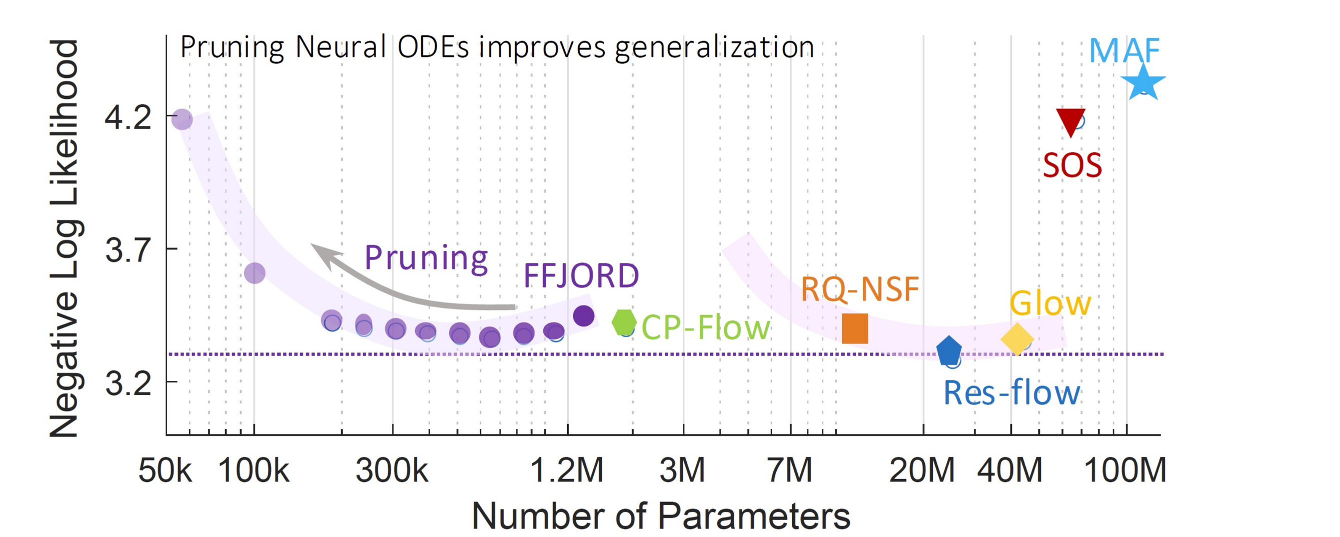
(a) Gaussians (b) Gaussian Spiral (c) Spiral



(a) MNIST. (b) CIFAR10.

Table 2: Negative test log-likelihood (NLL) in nats of tabular datasets from (Papamakarios et al., 2017) and corresponding architecture size in number of parameters (#params). Sparse Flow (based on FFJORD) with lowest NLL and competing baseline with lowest NLL are bolded.

Model	POWER		GAS		HEPMASS		MINIBOONE		BSDS300	
	nats	#params	nats	#params	nats	#params	nats	#params	nats	#params
MADE (Germain et al., 2015)	3.08	6K	-3.56	6K	20.98	147K	15.59	164K	-148.85	621K
Real NVP (Dinh et al., 2016)	-0.17	212K	-8.33	216K	18.71	5.46M	13.84	5.68M	-153.28	22.3M
MAF (Papamakarios et al., 2017)	-0.24	59.0K	-10.08	62.0K	17.70	1.47M	11.75	1.64M	-155.69	6.21M
Glow (Kingma and Dhariwal, 2018)	-0.17	N/A	-8.15	N/A	18.92	N/A	11.35	N/A	-155.07	N/A
CP-Flow (Huang et al., 2020)	-0.52	5.46M	-10.36	2.76M	16.93	2.92M	10.58	379K	-154.99	2.15M
TAN (Oliva et al., 2018b)	**-0.60**	N/A	**-12.06**	N/A	**13.78**	N/A	11.01	N/A	**-159.80**	N/A
NAF (Huang et al., 2018)	**-0.62**	**451K**	**-11.96**	**443K**	15.09	10.7M	**8.86**	**8.03M**	-157.73	42.3M
SOS (Jaini et al., 2019)	**-0.60**	**212K**	**-11.99**	**256K**	15.15	4.43M	**8.90**	**6.87M**	-157.48	9.09M
FFJORD (Grathwohl et al., 2019)	-0.35	43.3K	-8.58	279K	17.53	547K	10.50	821K	-128.33	6.70M
Sparse Flow	-0.45	30K	-10.79	194K	16.53	340K	10.84	397K	-145.62	4.69M
	-0.50	23K	-11.19	147K	15.82	160K	10.81	186K	-148.72	3.55M
	-0.53	**13K**	**-11.59**	**85K**	**15.60**	**75K**	**9.95**	**32K**	-150.45	2.03M
	-0.52	10K	-11.47	64K	15.99	46K	10.54	18K	**-151.34**	**1.16M**



Conclusions

- ✓ Pruning improves generalization in Neural ODEs and continuous flows
- ✓ Pruning helps avoid mode-collapse in Continuous Flows
- ✓ Pruning flattens the loss surface of continuous normalizing flows
- ✓ Maybe for continuous flows pruning is all you need?

Contact: Ramin Hasani, rhasani@mit.edu, Lucas Liebenwein, lucasl@mit.edu

Acknowledgements: BOEING, Aurora Flight Sciences A Boeing Company

GitHub https://github.com/ lucaslie/torchprune