Deep Generative Models
Lecture 12

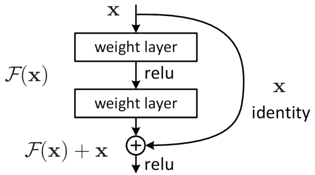
Roman Isachenko

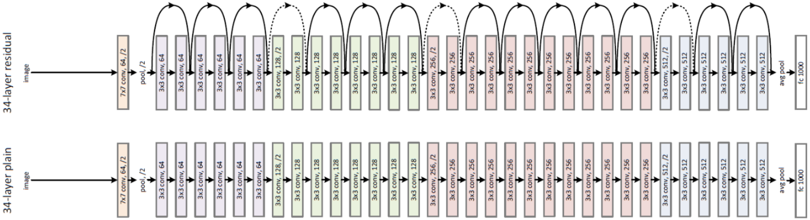
Moscow Institute of Physics and Technology

2020

Neural ODE, 2018

How did it become possible to train neural networks with hundreds of layers?
Skip connections eliminates exploding/vanishing gradients.





https://arxiv.org/pdf/1806.07366.pdf

Neural ODE, 2018

Consider ODE

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), \boldsymbol{\theta}); \quad \mathbf{z}(t_0) = \mathbf{z}_0.$$

Euler update step

$$\mathbf{z}(t + \Delta t) = \mathbf{z}(t) + \Delta t f(\mathbf{z}(t), \boldsymbol{\theta}).$$

Residual block

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f(\mathbf{z}_t, \boldsymbol{\theta}).$$

It is exactly Euler update step for solving ODE with $\Delta t = 1$!
Euler update step is unstable and trivial.

---

https://arxiv.org/pdf/1806.07366.pdf

Neural ODE, 2018

Residual block

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f(\mathbf{z}_t, \boldsymbol{\theta}).$$

What happens as we add more layers and take smaller steps?
In the limit, we parameterize the continuous dynamics of hidden units using an ODE specified by a neural network:

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t, \boldsymbol{\theta}); \quad \mathbf{z}(t_0) = \mathbf{x}; \quad \mathbf{z}(t_1) = \mathbf{y}.$$

Loss function

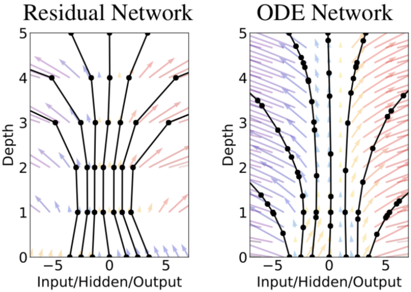
$$\begin{aligned} L(\mathbf{y}) = L(\mathbf{z}(t_1)) &= L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \boldsymbol{\theta}) dt\right) \\ &= L\left(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \boldsymbol{\theta})\right) \end{aligned}$$

https://arxiv.org/pdf/1806.07366.pdf

Neural ODE, 2018

Benefits

- memory efficient;
- adaptive computation;
- parameter efficient;
- scalable and invertible normalizing flows.



https://arxiv.org/pdf/1806.07366.pdf

Neural ODE, 2018

Loss function

$$L(\mathbf{y}) = L(\mathbf{z}(t_1)) = L\left(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \boldsymbol{\theta})\right)$$

How to train such model? How to fit $\boldsymbol{\theta}$? How to compute efficiently $\frac{\partial L}{\partial \boldsymbol{\theta}}$? – Pontryagin theorem!

Adjoint function

$$\mathbf{a}(t) = \frac{\partial L(\mathbf{z}(t))}{\partial \mathbf{z}(t)}$$

Theorem

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t), t, \boldsymbol{\theta})}{\partial \mathbf{z}(t)}$$

---

https://arxiv.org/pdf/1806.07366.pdf

Neural ODE, 2018

Theorem

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t), t, \boldsymbol{\theta})}{\partial \mathbf{z}(t)}; \quad \mathbf{a}(t) = \frac{\partial L(\mathbf{z}(t))}{\partial \mathbf{z}(t)}$$

To obtain $\mathbf{a}(t)$ along the trajectory we could solve this ODE backward in time, starting from the initial value $\mathbf{a}(t_1) = \frac{\partial L(\mathbf{z}(t_1))}{\partial \mathbf{z}(t_1)}$.

Theorem

$$\frac{dL}{d\boldsymbol{\theta}} = -\int_{t_0}^{t_1} \mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt.$$

All these gradients could be computed at once.

---

https://arxiv.org/pdf/1806.07366.pdf

Continuous NF, 2018

Discrete NF

$$\mathbf{z}_{t+1} = f(\mathbf{z}_t, \boldsymbol{\theta}); \quad \log p(\mathbf{z}_{t+1}) = \log p(\mathbf{z}_t) - \log \left| \det \frac{\partial f(\mathbf{z}_t, \boldsymbol{\theta})}{\partial \mathbf{z}_t} \right|.$$

Function f should be bijective!

Theorem

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\text{trace}\left(\frac{\partial f}{\partial \mathbf{z}(t)} \right).$$

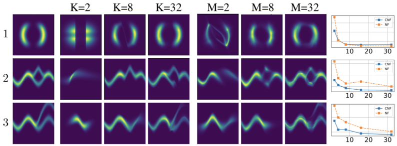
Function f is not necessary bijective! (uniformly Lipschitz continuous in $\mathbf{z}$ and continuous in t).

---

https://arxiv.org/pdf/1806.07366.pdf

Continuous NF, 2018

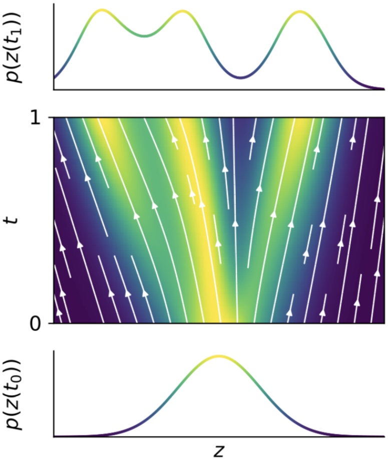
$$\log p(\mathbf{z}(t_1)) = \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \operatorname{trace}\left(\frac{\partial f}{\partial \mathbf{z}}\right) dt.$$



(a) Target (b) NF (c) CNF (d) Loss vs. K/M

https://arxiv.org/pdf/1806.07366.pdf

FFJORD, 2018



https://arxiv.org/pdf/1810.01367.pdf

FFJORD, 2018

Hutchinson's trace estimator

$$\mathrm{trace}(A) = \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T A \boldsymbol{\epsilon}\right]; \quad \mathbb{E}[\boldsymbol{\epsilon}] = 0; \quad \mathrm{Cov}(\boldsymbol{\epsilon}) = I.$$

$$\begin{aligned}
\log p(\mathbf{z}(t_1)) &= \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathrm{trace}\left(\frac{\partial f}{\partial \mathbf{z}}\right) dt \\
&= \log p(\mathbf{z}(t_0)) - \int_{t_0}^{t_1} \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}} \boldsymbol{\epsilon}\right] dt \\
&= \log p(\mathbf{z}(t_0)) - \mathbb{E}_{p(\boldsymbol{\epsilon})} \int_{t_0}^{t_1} \left[\boldsymbol{\epsilon}^T \frac{\partial f}{\partial \mathbf{z}} \boldsymbol{\epsilon}\right] dt.
\end{aligned}$$

This reduces the cost from quadratic to linear.

---

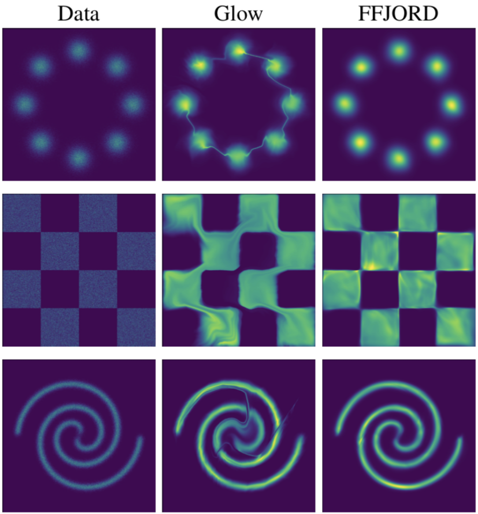
https://arxiv.org/pdf/1810.01367.pdf

FFJORD, 2018

	Method	Train on data	One-pass Sampling	Exact log-likelihood	Free-form Jacobian
	Variational Autoencoders	✓	✓	✗	✓
	Generative Adversarial Nets	✓	✓	✗	✓
	Likelihood-based Autoregressive	✓	✗	✓	✗
Change of Variables	Normalizing Flows	✗	✓	✓	✗
Change of Variables	Reverse-NF, MAF, TAN	✓	✗	✓	✗
Change of Variables	NICE, Real NVP, Glow, Planar CNF	✓	✓	✓	✗
Change of Variables	**FFJORD**	✓	✓	✓	✓

https://arxiv.org/pdf/1810.01367.pdf

FFJORD, 2018



https://arxiv.org/pdf/1810.01367.pdf

References

- Neural Ordinary Differential Equations
  https://arxiv.org/abs/1806.07366
  Summary: New interpretation of resnets as special case of ode. Discrete sequence of layers are replaced with continuous dynamic. ODESolver is used for backpropagation. Pontryagin theorem gives the analog of the chain rule. Continuous version of normalizing flow is constructed.
- **FFJORD:** Free-form Continuous Dynamics for Scalable Reversible Generative Models
  https://arxiv.org/abs/1810.01367
  Summary: Continuous version of NF is investigated. Jacobian computation cost is reduced to O(D) by using Hutchinson's trace estimator.