Neural ordinary differential equations

-Chen, Yubanova, et.al

Presented by : Haresh Karnan

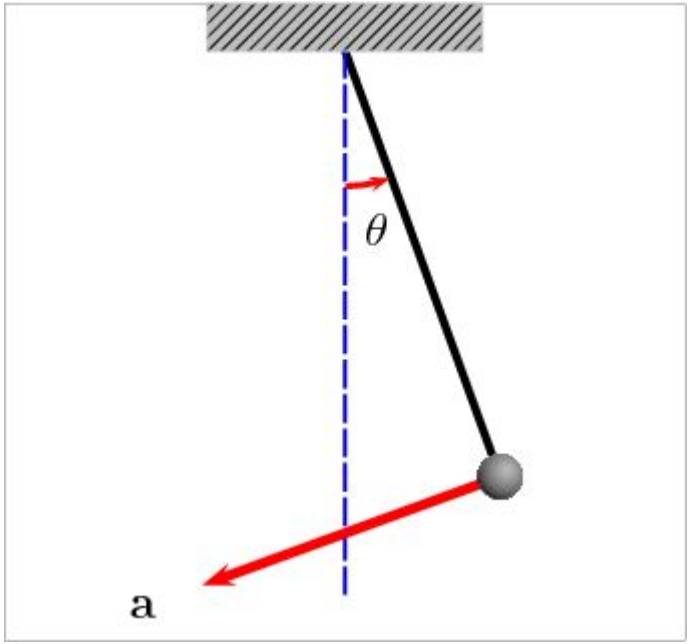
What is an ODE ?

$$\frac{d^2\theta}{dt^2} + \frac{g}{\ell}\sin\theta = 0$$

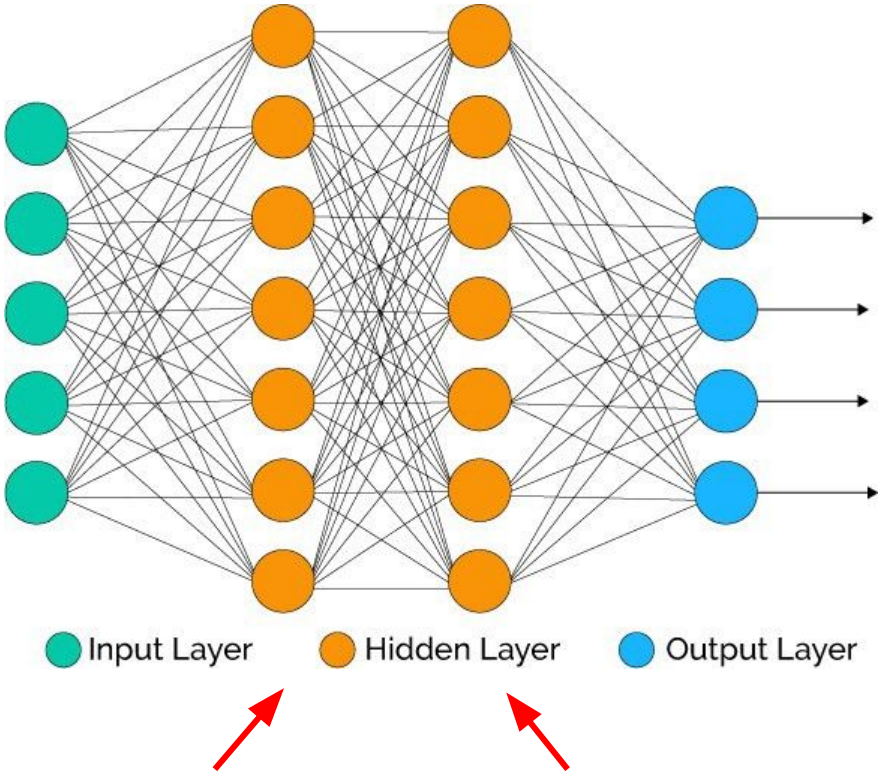
Techniques to solve an ODE :

1) Runge Kutta 4th order method
2) Heun’s method
3) Euler’s method

Simple pendulum :



Deep neural network :



$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$$

Advantages of this technique :

- Memory efficiency (no backpropagation)
- Adaptive computation (speed vs accuracy)
- Parameter efficiency (no discretization)
- Scalable and invertible normalizing flows (change of variables formula)
- Continuous time series models (continuous dynamics)

How do we compute the gradient ?

Adjoint method :

Compute gradients by solving a second ODE backwards in can do this easily). - scales linearly with problem size, has explicitly controls numerical errors.

$$L(\mathbf{z}(t_1)) = L\left(\int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta)dt\right) = L\left(\text{ODESolve}(\mathbf{z}(t_0), f, t_0, t_1, \theta)\right) \quad (3)$$

$$a(t) = -\partial L / \partial \mathbf{z}(t).$$

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

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

Replacing residual networks with NODE :

Table 1: Performance on MNIST. †From LeCun et al. (1998).

	Test Error	# Params	Memory	Time
1-Layer MLP†	1.60%	0.24 M	-	-
ResNet	0.41%	0.60 M	$\mathcal{O}(L)$	$\mathcal{O}(L)$
RK-Net	0.47%	0.22 M	$\mathcal{O}(\tilde{L})$	$\mathcal{O}(\tilde{L})$
ODE-Net	0.42%	0.22 M	$\mathcal{O}(1)$	$\mathcal{O}(\tilde{L})$

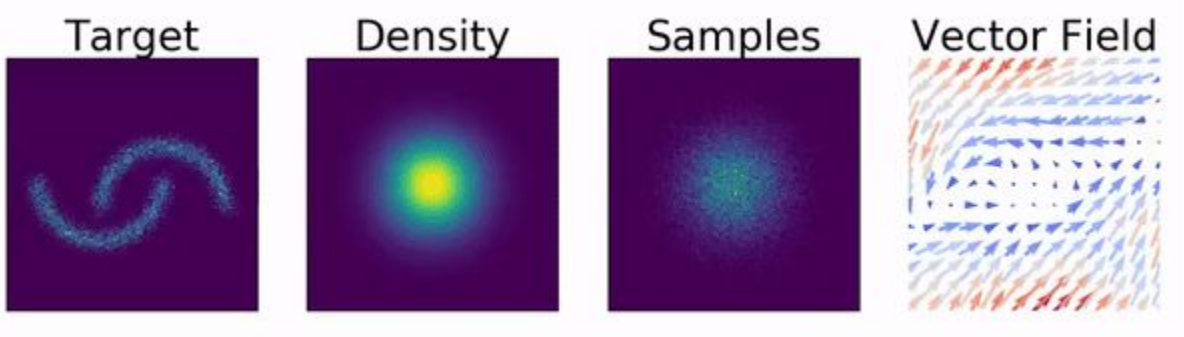
“ Supervised learning ”

Continuous normalizing flows :

Theorem 1 (Instantaneous Change of Variables). *Let $\mathbf{z}(t)$ be a finite continuous random variable with probability $p(\mathbf{z}(t))$ dependent on time. Let $\frac{d\mathbf{z}}{dt} = f(\mathbf{z}(t), t)$ be a differential equation describing a continuous-in-time transformation of $\mathbf{z}(t)$. Assuming that f is uniformly Lipschitz continuous in $\mathbf{z}$ and continuous in t, then the change in log probability also follows a differential equation,*

$$\frac{\partial \log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)}\right) \tag{8}$$

1) Density matching
2) Maximum likelihood training



Pros

Pros :

- Novel idea.
- Appendix section is very elaborate and the authors derive connections with a special case of Fokker-Planck equation.
- Appendix C provides code !.
- Cool visualization.