NODE 1: How To Train Your Neural ODE

CS 598: Deep Generative and Dynamical Models

Instructor: Arindam Banerjee

November 2, 2021

Motivation

- Similarity between ResNets and numerical ODE solvers
- Residual models as differential equations
- Layers becomes equivalent to 'time'
- Many possible applications
  - Physical sciences, continuous time modeling
  - Generative models using flows
  - Extensions to stochastic PDEs

Neural ODEs

- Block t in ResNet: $\mathsf{x}^{t+1} = \mathsf{x}^t + \mathsf{f}(\mathsf{x}, t; \theta)$
- Consider the ODE

$$\dot{\mathsf{z}} = \mathsf{f}(\mathsf{z}, t; \theta)$$
$$\mathsf{z}(0) = \mathsf{x}$$

- Euler discretization of the ODE: $\mathsf{z}^{t+1} = \mathsf{z}^t + \tau f(\mathsf{z}^t, t; \theta)$
- Perspective: Continuous 'time' generalization of ResNets
  - $\tau = 1$ gives ResNets

Neural ODEs

- Memory footprint of Neural ODEs can be much smaller
- Training time can be long:
  - Cost of numerically integrating the ODEs
- Goal: Regularization to help solve the ODEs fast

Continuous Normalizing Flows

- Goal is develop model $p_\theta(\mathrm{x})$ to maximize likelihood, or minimize

$$J(p_\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathrm{x}_i)$$

- Parameterize $p_\theta(\mathrm{x})$ using a vector field $\mathrm{f} : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}^d$
- $z(\mathrm{x}, T)$ be the solution map by running dynamics till T
- Change of variables

$$\log p_\theta(\mathrm{x}) = \log q(z(\mathrm{x}, T)) + \log\det|\nabla \mathrm{z}(\mathrm{x}, T)|$$

- Can be done efficiently using

$$\frac{\partial}{\partial t}\log\det|\nabla \mathrm{z}(\mathrm{x}, t)| = \mathrm{div}(\mathrm{f})(\mathrm{z}(\mathrm{x}, t), t)$$

- Recall that $\mathrm{div}(\mathrm{f})(\mathrm{x}) = \sum_i \partial_{x_i} f_i(\mathrm{x}) = \mathrm{Tr}(\nabla \mathrm{f}(\mathrm{x}))$

Continuous Normalizing Flows (Contd.)

- By fundamental theorem of calculus

$$\log p_\theta(x) = \log q(z(x, T)) + \int_0^T \text{div}(f)(z(x, s), s) ds$$

- Normalizing flow with free form Jacobian
- Divergence can be estimated by an unbiased MC estimate

$$\text{div}(f)(x) = \mathbb{E}_{\epsilon \sim \mathcal{N}(0, \mathbb{I})} \left[\epsilon^T \nabla f(x) \epsilon \right]$$

- Recall:

$$\mathbb{E}_\epsilon \left[\epsilon^T \nabla f(x) \epsilon \right] = \mathbb{E}_\epsilon \left[\text{Tr}(\epsilon^T \nabla f(x) \epsilon) \right] = \mathbb{E}_\epsilon \left[\text{Tr}(\nabla f(x) \epsilon \epsilon^T) \right]$$
$$= \text{Tr}(\nabla f(x) \mathbb{E}[\epsilon \epsilon^T]) = \text{Tr}(\nabla f(x) \mathbb{I}) = \text{Tr}(\nabla f(x))$$

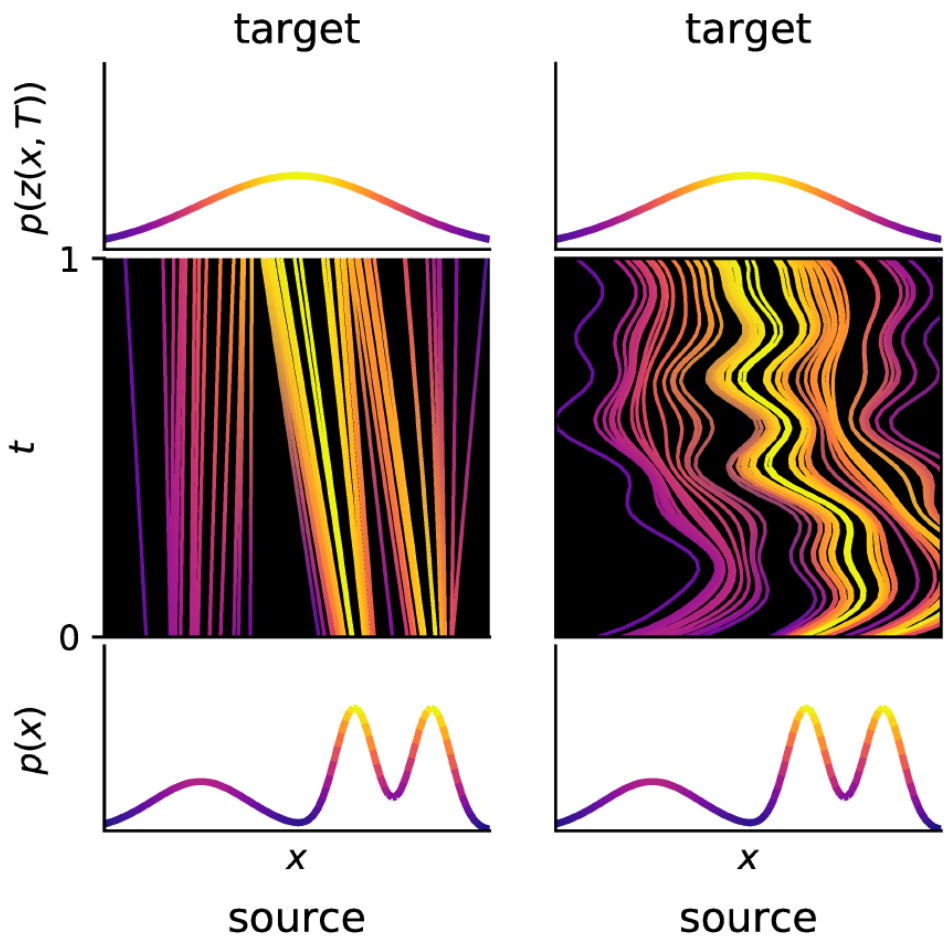
Regularizing the Flow

- Without regularization, optimal vector field f is not unique
- f may be poorly conditioned
  - Rapidly varying local trajectories and non-constant speed
  - Leads to difficulty during numerical integration of the ODE
- Regularization: Field places nearly constant force on the particles
- Force experienced by particle $z(t)$

$$\frac{df(z,t)}{dt} = \nabla f(z,t) \cdot \dot{z} + \frac{\partial f(z,t)}{\partial t}$$
$$= \nabla f(z,t) \cdot f(z,t) + \frac{\partial f(z,t)}{\partial t}$$

- Regularize two terms
  - f: so the distance traveled is small, based on optimal transport
  - ∇f: so the (Frobeius) norm is small, regularized Jacobian

Unregularized vs. Optimal Transport Flows



(a) Optimal transport map (b) generic flow

Figure 1. Optimal transport map and a generic normalizing flow.

Regularized Dynamics: Optimal Transport

- Quadratic cost optimal transport minimizes transportation cost

$$M(z) = \int \|x - z(x)\|^2 p(x) dx \ , \quad s.t. \int_A q(z) dz = \int_{z^{-1}(A)} p(x) dx)$$

- Alternative (Benamou-Brenier) approach:
  - $z(x, T)$ is solution map of vector field f (ODE)
  - Optimal transport by minimizing

$$\min_{f,\rho} \int_0^T \int \|f(x,t)\|^2 \rho_t(x) dx dt$$

$$\text{s.t.} \quad \frac{\partial \rho_t}{\partial t} = -\text{div}(\rho_t f) \qquad \text{(probability mass conservation)}$$

$$\rho_0(x) = p \qquad \text{(source marginal)}$$

$$\rho_T(z) = q \qquad \text{(target marginal)}$$

- Particles under the optimal flow travel in straight lines
- Optimal solution is unique, under assumptions

Regularized Dynamics: Optimal Transport (Contd.)

- Solution map $\mathrm{z}(\mathrm{x}, t) = (1 - t/T)\mathrm{z}(\mathrm{x}, 0) + t/T\,\mathrm{z}(\mathrm{x}, T)$
  - Makes it easy to solve (ODE)
- For normalizing flows, $q(\mathrm{z})$ is normal, $p(\mathrm{x})$ is data distribution
  - Source marginal is satisfied, data drawn from true distribution
  - Target marginal is approximated by $KL(\rho_T \| q)$, maximum likelihood
  - Probability mass conservation not needed, tracking finite particles
- Formulation becomes

$$J_\lambda(\mathrm{f}) = \frac{\lambda}{N} \sum_{i=1}^{N} \int_0^T \|\mathrm{f}(\mathrm{z}_i, t)\|^2 dt - \frac{1}{N} \sum_{i=1}^{N} \log p_\theta(\mathrm{x}_i)$$

- Regularized form of the continuous normalizing flow problem

Regularizing the Jacobian

- Transport regularization focuses on the training set
- Want the 'smoothness' of the flow to generalize ı
- Regularize the Jacobian, based on Frobenius norm
- Efficient computation using vector-Jacobian product $\epsilon^T \nabla \mathrm{f}(\mathrm{z})$

$$\|\nabla \mathrm{f}(\mathrm{z})\|_F^2 = \mathbb{E}_{\epsilon \sim N(0,\mathbb{I})} \|\epsilon^T \nabla f(\mathrm{z})\|^2$$

Regularized Neural ODE (RNODE)

- Overall optimization problem to be solved

$$\min_{\mathsf{f}} \frac{1}{Nd} \sum_{i=1}^{N} -\log q(\mathsf{z}(\mathsf{x}_i, T)) - \int_0^T \mathrm{div}(\mathsf{z}(\mathsf{x}_i, s), s) ds$$
$$+ \lambda_K \int_0^T \|\mathsf{f}(\mathsf{z}(\mathsf{x}_i, s), s)\|^2 ds + \lambda_J \int_0^T \|\nabla_{\mathsf{z}} \mathsf{f}(\mathsf{z}(\mathsf{x}_i, s), s)\|_F^2 ds$$

- Compute the three integrals by augmenting the ODE

$$\dot{\mathsf{z}} = \mathsf{f}(\mathsf{z}, t)$$
$$\dot{l} = \mathrm{div}(\mathsf{f})(\mathsf{z}, t)$$
$$\dot{E} = \|\mathsf{f}(\mathsf{z}, t)\|^2$$
$$\dot{n} = \|\nabla \mathsf{f}(\mathsf{z}, t)\|_F^2$$
$$\mathsf{z}(0) = \mathsf{x}, E(0) = l(0) = n(0) = 0$$

Algorithm: RNODE

Algorithm 1 RNODE: regularized neural ODE training of FFJORD

Input: data $X = \{\mathbf{x}_i\}$, $i = 1, \cdots, N$, dynamics $\mathbf{f}(\cdot\,; \theta)$, final time T, regularization strength λ_J and λ_K
initialize θ
while θ not converged **do**
  Sample ϵ from standard normal distribution
  Sample minibatch $\{\mathbf{x}_j\}$ of size m from X
  Set $\mathbf{z}_j(0) = \mathbf{x}_j$, $l_j(0) = E_j(0) = n_j = 0$
  Numerically solve up to time T the system

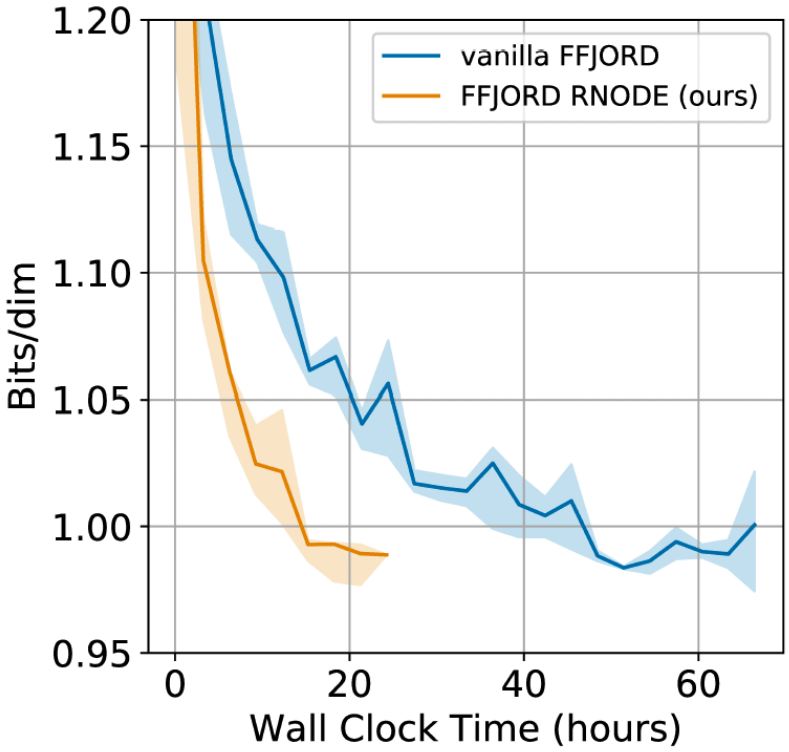
$$\begin{cases} \dot{\mathbf{z}}_j = \mathbf{f}(\mathbf{z}_j, t; \theta) \\ \dot{l}_j = \epsilon^\mathsf{T} \nabla \mathbf{f}(\mathbf{z}_j, t; \theta)\epsilon \\ \dot{E}_j = \|\mathbf{f}(\mathbf{z}_j, t; \theta)\|^2 \\ \dot{n}_j = \|\epsilon^\mathsf{T} \nabla \mathbf{f}(\mathbf{z}_j, t; \theta)\|^2 \end{cases}$$

  Compute

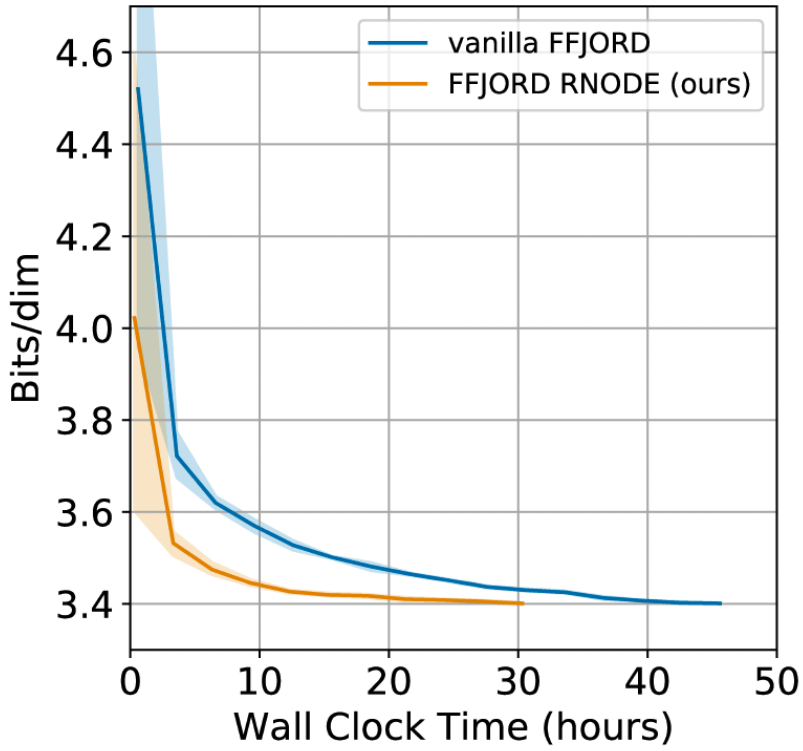
$$L(\theta) = \frac{1}{m}\sum_{j=1}^{m} - \log q(z_j(T)) - l_j(T) + \lambda_J n_j(T) + \lambda_K E_j(T)$$

  Compute $\nabla_\theta L(\theta)$ using the adjoint sensitivity method by numerically solving the adjoint equations
  Update $\theta \leftarrow \theta - \tau \nabla_\theta L(\theta)$
end while

Running Time: RNODE vs. FFJORD



(a) MNIST



(b) CIFAR10

Jacobian Norm vs. Function Evaluation

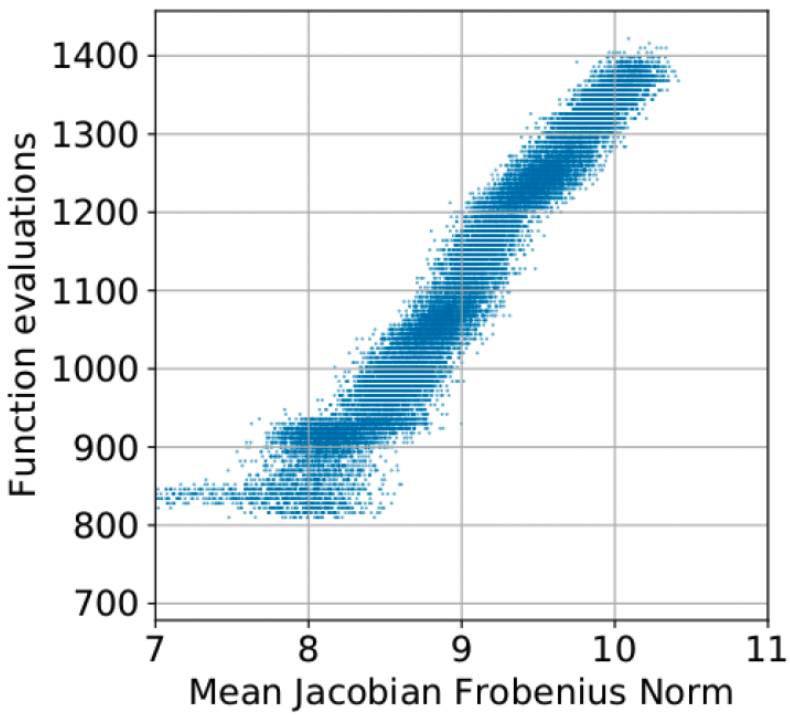
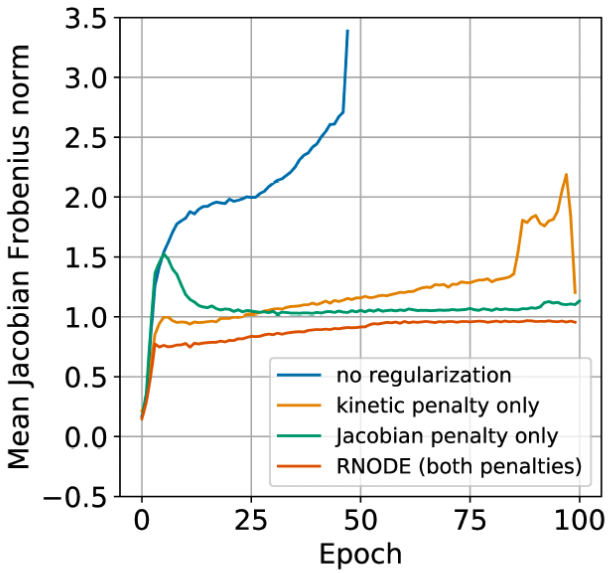


Figure 3. Number of function evaluations vs Jacobian Frobenius norm of flows on CIFAR10 during training with vanilla FFJORD, using an adaptive ODE solver.

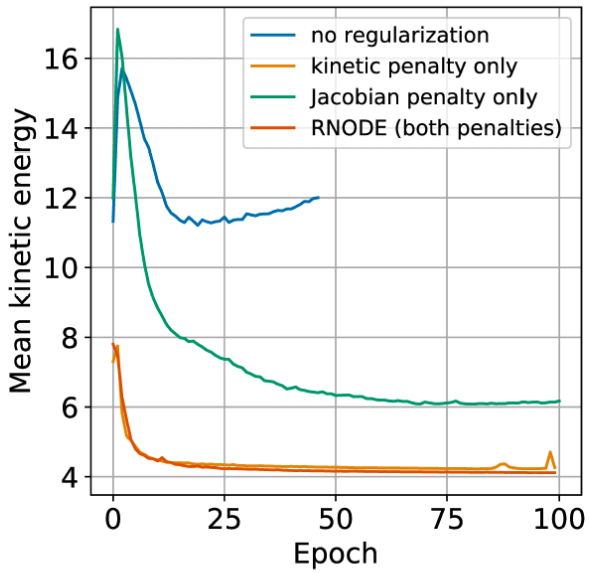
Results: Log-likelihood (bits/dim)

	MNIST		CIFAR10		IMAGENET64		CELEBA-HQ256	
	BITS/DIM	TIME	BITS/DIM	TIME	BITS/DIM	TIME		
FFJORD, ORIGINAL	0.99	-	3.40	$\geq$ 5 DAYS	-	-	-	-
FFJORD, VANILLA	0.97	68.5	3.36	91.3	X	X	-	-
FFJORD RNODE (OURS)	0.97	24.4	3.38	31.8	3.83	64.1	1.04	6.6 DAYS
REALNVP (DINH ET AL., 2017)	1.06	-	3.49	-	3.98	-	-	-
I-RESNET (BEHRMANN ET AL., 2019)	1.05	-	3.45	-	-	-	-	-
GLOW (KINGMA & DHARIWAL, 2018)	1.05	-	3.35	-	3.81	-	1.03	7 DAYS[2]
FLOW++ (HO ET AL., 2019)	-	-	3.28	-	-	-	-	-
RESIDUAL FLOW (CHEN ET AL., 2019)	0.97	-	3.28	-	3.76	-	0.99	-

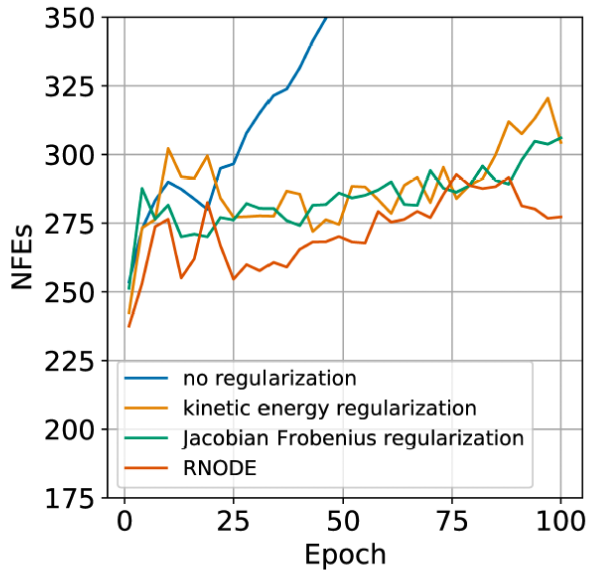
Results: Ablation Study



(a) Jacobian norm



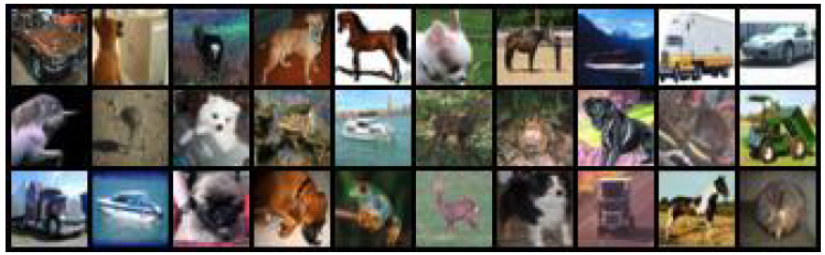
(b) Kinetic energy



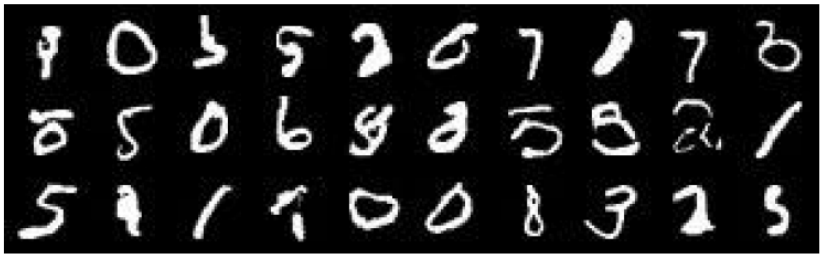
(c) Function evaluations

Results: Samples

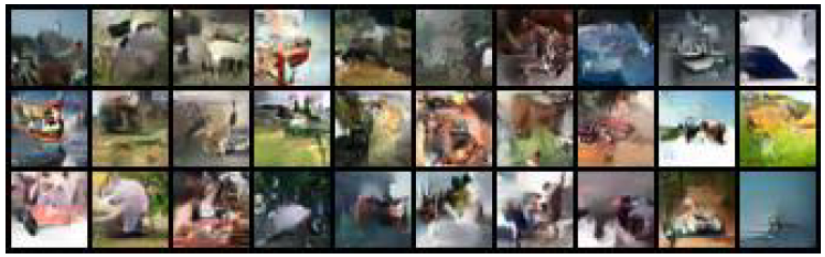
(a) real MNIST images



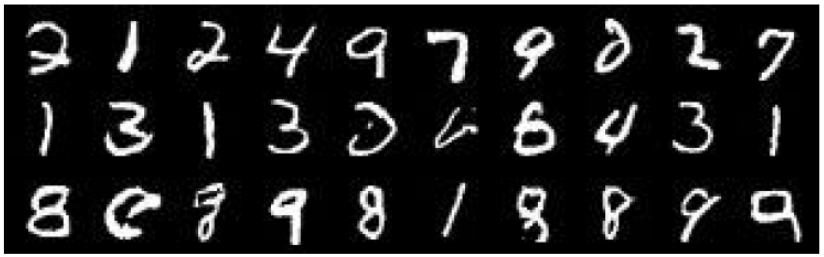
(b) real CIFAR10 images



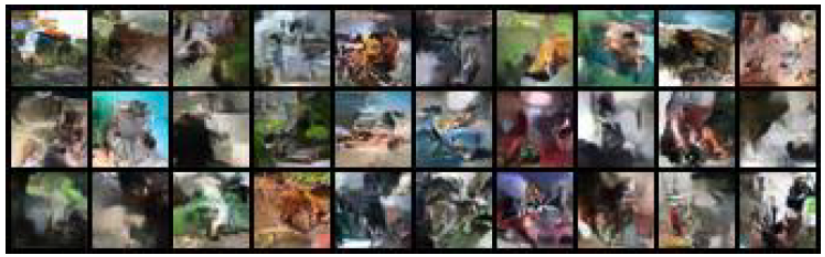
(c) vanilla FFJORD



(d) vanilla FFJORD



(e) FFJORD RNODE



(f) FFJORD RNODE

Figure 6. Quality of generated samples samples with and without regularization on MNIST, left, and CIFAR10, right.

References

- R. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud. Neural ordinary differential equations NeurIPS, 2018.
- C. Finlay, J. Jacobsen, L. Nurbekyan, A. Oberman. How to train your neural ODE, ICML, 2020.