How To Train Your Neural ODE

Chris Finlay [1] **Jörn-Henrik Jacobsen** [2] **Levon Nurbekyan** [3] **Adam M Oberman** [1]

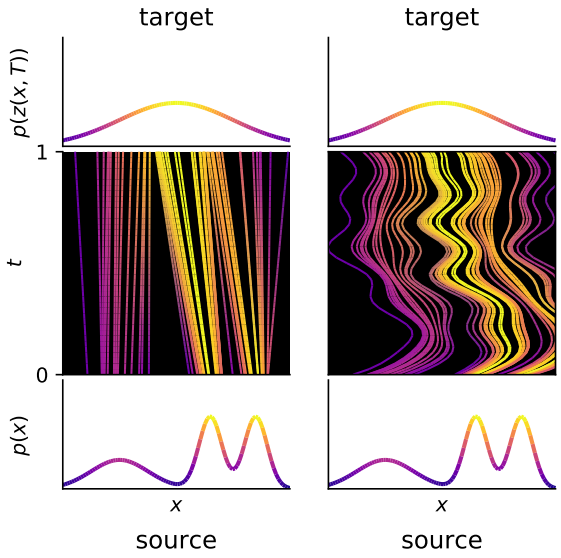
Abstract

Training neural ODEs on large datasets has not been tractable due to the necessity of allowing the adaptive numerical ODE solver to refine its step size to very small values. In practice this leads to dynamics equivalent to many hundreds or even thousands of layers. In this paper, we overcome this apparent difficulty by introducing a theoretically-grounded combination of both optimal transport and stability regularizations which encourage neural ODEs to prefer simpler dynamics out of all the dynamics that solve a problem well. Simpler dynamics lead to faster convergence and to fewer discretizations of the solver, considerably decreasing wall-clock time without loss in performance. Our approach allows us to train neural ODE based generative models to the same performance as the unregularized dynamics in just over a day on one GPU, whereas unregularized dynamics can take up to 4-6 days of training time on multiple GPUs. This brings neural ODEs significantly closer to practical relevance in large-scale applications.

1. Introduction

Recent research has bridged dynamical systems, a workhorse of mathematical modeling, with neural networks, the defacto function approximator for high dimensional data. The great promise of this pairing is that the vast mathematical machinery stemming from dynamical systems can be leveraged for modelling high dimensional problems in a dimension-independent fashion.

Connections between neural networks and ordinary differential equations (ODEs) were almost immediately noted after residual networks (He et al., 2016) were first proposed. Indeed, it was observed that there is a striking similarity between ResNets and the numerical solution of ordinary differential equations (Haber & Ruthotto, 2017; Ruthotto & Haber, 2018; Chen et al., 2018; 2019). In these works, deep networks are interepreted as discretizations of an underlying dynamical system, where time indexes the "depth" of the network and the parameters of the discretized dynamics are learned. An alternate viewpoint was taken by neural ODEs (Chen et al., 2018), where the dynamics of the neural network are approximated by an adaptive ODE solver on the fly. This latter approach is quite compelling as it does not require specifying the number of layers of the network beforehand. Furthermore, it allows the learning of homeomorphisms without any structural constraints on the function computed by the residual block.



(a) Optimal transport map (b) generic flow

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

Neural ODEs have shown great promise in the physical sciences (Köhler et al., 2019), in modeling irregular time series (Rubanova et al., 2019), mean field games (Ruthotto et al., 2019), and for generative modeling through normalizing flows with free-form Jacobians (Grathwohl et al., 2019). Recent work has even adapted neural ODEs to the stochastic setting (Li et al., 2020). Despite these successes, some hurdles still remain. In particular, although neural ODEs are memory efficient, they can take a prohibitively long time to

[1]Department of Mathematics & Statistics, McGill University, Montréal, Québec, Canada [2]Vector Institute, University of Toronto, Toronto, Ontario, Canada [3]Department of Mathematics, UCLA, California, USA. Correspondence to: Chris Finlay <christopher.finlay@mcgill.ca>.

Preprint. Under review.

train, which is arguably one of the main stumbling blocks towards their widespread adoption.

In this work we significantly reduce the training time of neural ODEs by regularizing the learned dynamics. Without further constraints on their dynamics, high dimensional neural ODEs may learn dynamics which minimize an objective function, but which generate irregular solution trajectories. See for example Figure 1b, where an unregularized flow exhibits undesirable properties due to unnecessarily fluctuating dynamics. As a solution, we propose two theoretically motivated regularization terms arising from an optimal transport viewpoint of the learned map, which encourage well-behaved dynamics (see 1b left). We empirically demonstrate that proper regularization leads to significant speed-up in training time without loss in performance, thus bringing neural ODEs closer to deployment on large-scale datasets. Our methods are validated on the problem of generative modelling and density estimation, as an example of where neural ODEs have shown impressive results, but could easily be applied elsewhere.

In summary, our proposed regularized neural ODE (RNODE) achieves the same performance as the baseline, while reducing the wall-clock training time by many hours or even days.

2. Neural ODEs & Continuous normalizing flows

Neural ODEs simplify the design of deep neural networks by formulating the forward pass of a deep network as the solution of a ordinary differential equation. Initial work along these lines was motivated by the similarity of the evaluation of one layer of a ResNet and the Euler discretization of an ODE. Suppose the block in the t-th layer of a ResNet is given by the function $\mathbf{f}(\mathbf{x}, t; \theta)$, where θ are the block's parameters. Then the evaluation of this layer of the ResNet is simply $\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{f}(\mathbf{x}^t, t; \theta)$. Now, instead consider the following ODE

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, t; \theta) \\ \mathbf{z}(0) = \mathbf{x} \end{cases} \tag{ODE}$$

The Euler discretization of this ODE with step-size τ is $\mathbf{z}^{t+1} = \mathbf{z}^t + \tau \mathbf{f}(\mathbf{z}^t, t; \theta)$, which is nearly identical to the forward evaluation of the ResNet's layer (setting step-size $\tau = 1$ gives equality). Armed with this insight, Chen et al. (2018) suggested a method for training neural networks based on (ODE) which abstain from *a priori* fixing step-size. Chen et al.'s method is a continuous-time generalization of residual networks, where the dynamics are generated by an *adaptive* ODE solver that chooses step-size on-the-fly.

Because of their adaptive nature, neural ODEs can be more flexible than ResNets in certain scenarios, such as when trading between model speed and accuracy. Moreover given a fixed network depth, the memory footprint of neural ODEs is orders of magnitude smaller than a standard ResNet during training. They therefore show great potential on a host of applications, including generative modeling and density estimation. An apparent drawback of neural ODEs is their long training time: although a learned function $\mathbf{f}(\cdot\,; \theta)$ may generate a map that solves a problem particularly well, the computational cost of numerically integrating (ODE) may be so prohibitive that it is not tractable in practice. In this paper we demonstrate this need not be so: with proper regularization, it is possible to learn $\mathbf{f}(\cdot\,; \theta)$ so that (ODE) is easily and quickly solved.

2.1. FFJORD

In density estimation and generative modeling, we wish to estimate an unknown data distribution $p(\mathbf{x})$ from which we have drawn N samples. Maximum likelihood seeks to approximate $p(\mathbf{x})$ with a parameterized distribution $p_\theta(\mathbf{x})$ by minimizing the Kullback-Leibler divergence between the two, or equivalently minimizing

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

Continuous normalizing flows (Grathwohl et al., 2019; Chen et al., 2018) parameterize $p_\theta(\mathbf{x})$ using a vector field $\mathbf{f} : \mathbb{R}^d \times \mathbb{R} \mapsto \mathbb{R}^d$ as follows. Let $\mathbf{z}(\mathbf{x}, T)$ be the solution map given by running the dynamics (ODE) for fixed time T. Suppose we are given a *known* distribution q at final time T, such as the normal distribution. Change of variables tells us that the distribution $p_\theta(\mathbf{x})$ may be evaluated through

$$\log p_\theta(\mathbf{x}) = \log q\left(\mathbf{z}(\mathbf{x}, T)\right) + \log \det \left| \nabla \mathbf{z}(\mathbf{x}, T) \right| \tag{2}$$

Evaluating the log determinant of the Jacobian is difficult. Grathwohl et al. (2019) exploit the following identity from fluid mechanics (Villani, 2003, p 114)

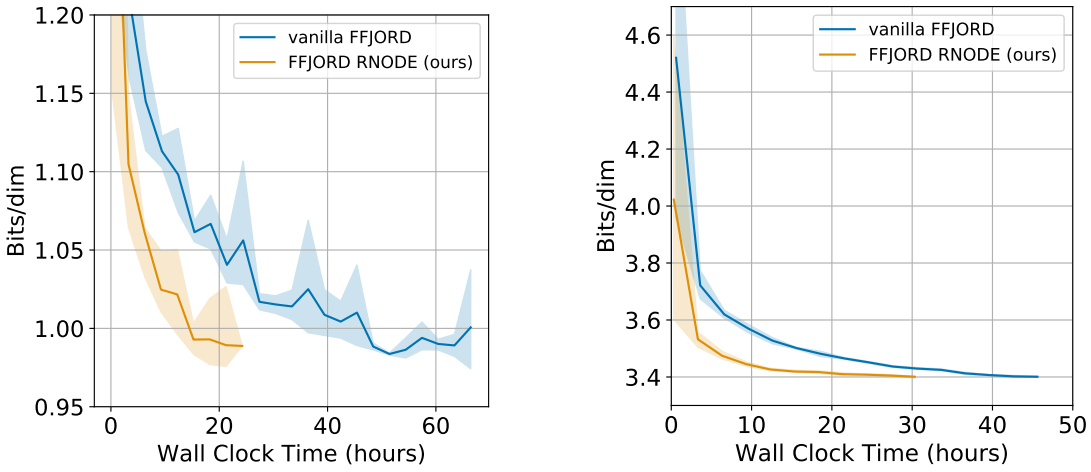
$$\frac{\partial}{\partial t} \log \det \left| \nabla \mathbf{z}(\mathbf{x}, t) \right| = \operatorname{div}\left(\mathbf{f}\right)\left(\mathbf{z}(\mathbf{x}, t), t\right) \tag{3}$$

where $\operatorname{div}(\cdot)$ is the divergence operator[1], $\operatorname{div}\left(\mathbf{f}\right)(\mathbf{x}) = \sum_i \partial_{x_i} f_i(\mathbf{x})$. By the fundamental theorem of calculus, we may then rewrite (2) in integral form

$$\log p_\theta(\mathbf{x}) = \log q\left(\mathbf{z}(\mathbf{x}, T)\right) + \int_0^T \operatorname{div}\left(\mathbf{f}\right)\left(\mathbf{z}(\mathbf{x}, s), s\right) \mathrm{d}s \tag{4}$$

Remark 2.1 (Divergence trace estimate). In (Grathwohl et al., 2019), the divergence is estimated using an unbiased

[1] In the normalizing flow literature divergence is typically written explicitly as the trace of the Jacobian, however we use $\operatorname{div}(\cdot)$ which is more common elsewhere.



(a) MNIST (b) CIFAR10

Figure 2. Log-likelihood (measured in bits/dim) on the validation set as a function of wall-clock time. Rolling average of three hours, with 90% confidence intervals.

Monte-Carlo trace estimate (Hutchinson, 1990; Avron & Toledo, 2011),

$$\operatorname{div}(\mathbf{f})(\mathbf{x}) = \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,1)}\left[\epsilon^\intercal \nabla \mathbf{f}(\mathbf{x})\epsilon\right] \tag{5}$$

By using the substitution (4), the task of maximizing log-likelihood shifts from choosing p_θ to minimize (1), to learning the flow generated by a vector field $\mathbf{f}$. This results in a normalizing flow with a free-form Jacobian and reversible dynamics, and was named FFJORD by Grathwohl et al..

2.2. The need for regularity

The vector field learned through FFJORD that maximizes the log-likelihood is not unique, and raises troubling problems related to the regularity of the flow. For a simple example, refer to Figure 1, where we plot two normalizing flows, both mapping a toy one-dimensional distribution to the unit Gaussian, and where both maximize the log-likelihood of exactly the same sample of particles. Figure 1a presents a "regular" flow, where particles travel in straight lines that travel with constant speed. In contrast, Figure 1b shows a flow that still maximizes the log-likelihood, but that has undesirable properties, such as rapidly varying local trajectories and non-constant speed.

From this simple motivating example, the need for regularity of the vector field is apparent. Without placing demands on the vector field $\mathbf{f}$, it is entirely possible that the learned dynamics will be poorly conditioned. This is not just a theoretical exercise: because the dynamics must be solved with a numerical integrator, poorly conditioned dynamics will lead to difficulties during numerical integration of (ODE).

Indeed, later we present results demonstrating a clear correlation between the number of time steps an adaptive solver takes to solve (ODE), and the regularity of $\mathbf{f}$.

How can the regularity of the vector field be measured? One motivating approach is to measure the force experienced by a particle $\mathbf{z}(t)$ under the dynamics generated by the vector field $\mathbf{f}$, which is given by the total derivative of $\mathbf{f}$ with respect to time

$$\frac{\mathrm{d}\mathbf{f}(\mathbf{z},t)}{\mathrm{d}t} = \nabla \mathbf{f}(\mathbf{z},t)\cdot\dot{\mathbf{z}} + \frac{\partial \mathbf{f}(\mathbf{z},t)}{\partial t} \tag{6}$$

$$= \nabla \mathbf{f}(\mathbf{z},t)\cdot\mathbf{f}(\mathbf{z},t) + \frac{\partial \mathbf{f}(\mathbf{z},t)}{\partial t} \tag{7}$$

Well conditioned flows will place constant, or nearly constant, force on particles as they travel. Thus, in this work we propose regularizing the dynamics with two penalty terms, one term regularizing $\mathbf{f}$ and the other $\nabla \mathbf{f}$. The first penalty, presented in Section 3, is a measure of the distance travelled under the flow $\mathbf{f}$, and can alternately be interpreted as the kinetic energy of the flow. This penalty term is based off of numerical methods in optimal transport, and encourages particles to travel in straight lines with constant speed. The second penalty term, discussed in Section 4, performs regularization on the Jacobian of the vector field. Taken together the two terms ensure that the force experienced by a particle under the flow is constant or nearly so.

These two regularizers will promote dynamics that follow numerically easy-to-integrate paths, thus greatly speeding up training time.

3. Optimal transport maps & Benamou-Brenier

There is a remarkable similarity between density estimation using continuous time normalizing flows, and the calculation of the optimal transport map between two densities using the Benamou-Brenier formulation (Benamou & Brenier, 2000; Santambrogio, 2015). While a review of optimal transport theory is far outside the scope of this paper, here we provide an informal summary of key ideas relevant to continuous normalizing flows. The quadratic-cost optimal transport map between two densities $p(\mathbf{x})$ and $q(\mathbf{x})$ is a map $\mathbf{z} : \mathbb{R}^d \mapsto \mathbb{R}^d$ minimizing the transport cost

$$M(\mathbf{z}) = \int \|\mathbf{x} - \mathbf{z}(\mathbf{x})\|^2 p(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{8}$$

subject to the constraint that $\int_A q(\mathbf{y})\, \mathrm{d}\mathbf{y} = \int_{\mathbf{z}^{-1}(A)} p(\mathbf{x})\, \mathrm{d}\mathbf{x}$, in other words that the measure of any set A is preserved under the map $\mathbf{z}$. In a seminal work, Benamou & Brenier (2000) showed that rather than solving for minimizers of (8) directly, an indirect (but computationally efficient) method is available by writing $\mathbf{z}(\mathbf{x}, T)$ as the solution map of a flow under a vector field $\mathbf{f}$ (as in (ODE)) for time T, by minimizing

$$\min_{\mathbf{f},\rho} \quad \int_0^T \int \|\mathbf{f}(\mathbf{x}, t)\|^2 \rho_t(\mathbf{x})\, \mathrm{d}\mathbf{x}\mathrm{d}t \tag{9a}$$

$$\text{subject to} \quad \frac{\partial \rho_t}{\partial t} = -\operatorname{div}\left(\rho_t \mathbf{f}\right), \tag{9b}$$

$$\rho_0(\mathbf{x}) = p, \tag{9c}$$

$$\rho_T(\mathbf{z}) = q. \tag{9d}$$

The objective function (9a) is a measure of the *kinetic energy* of the flow. The constraint (9b) ensures probability mass is conserved. The latter two constraints guarantee the learned distribution agrees with the source p and target q. Note that the kinetic energy (9a) is an upper bound on the transport cost, with equality only at optimality.

The optimal flow $\mathbf{f}$ minimizing (9) has several particularly appealing properties. First, particles induced by the optimal flow $\mathbf{f}$ *travel in straight lines*. Second, particles travel with *constant speed*. Moreover, under suitable conditions on the source and target distributions, the optimal solution map is unique (Villani, 2008). Therefore the solution map $\mathbf{z}(\mathbf{x}, t)$ is entirely characterized by the initial and final positions: $\mathbf{z}(\mathbf{x}, t) = (1 - \frac{t}{T})\mathbf{z}(\mathbf{x}, 0) + \frac{t}{T}\mathbf{z}(\mathbf{x}, T)$. Consequently, given an optimal $\mathbf{f}$ it is extraordinarily easy to solve (ODE) numerically with minimal computational effort.

3.1. Linking normalizing flows to optimal transport

Now suppose we wish to minimize (9a), with $q(\mathbf{z})$ a unit normal distribution, and $p(\mathbf{x})$ a data distribution, unknown to us, but from which we have drawn N samples, and which we model as a discrete distribution of Dirac masses. Enforcing the initial condition is trivial because we have sampled from p directly. The continuity equation need not be enforced because we are tracking a finite number of sampled particles. However the final time condition $\rho_T = q$ cannot be implemented directly, since we do not have direct control on the form $\rho_T(\mathbf{z})$ takes. Instead, introduce a Kullback-Leibler term to (9a) penalizing discrepancy between ρ_T and q. This penalty term has an elegant simplification when $p(x)$ is modeled as a distribution of a finite number of masses, as is done in generative modeling. Setting $\rho_0 = p_\theta$ a brief derivation yields

$$\mathrm{KL}(\rho_T || q) = -\frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) \tag{10}$$

With this simplification (9a) becomes

$$J_\lambda(\mathbf{f}) = \frac{\lambda}{N}\sum_{i=1}^{N}\int_0^T \|\mathbf{f}(\mathbf{z}_i, t)\|^2 \mathrm{d}t - \frac{1}{N}\sum_{i=1}^{N} \log p_\theta(\mathbf{x}_i) \tag{11}$$

The connection between the Benamou-Brenier formulation of the optimal transport problem on a discrete set of points and continuous normalizing flows is apparent: the optimal transport problem (11) is a regularized form of the continuous normalizing flow optimization problem (1). We therefore expect that adding a kinetic energy regularization term to FFJORD will encourage solution trajectories to prefer straight lines with constant speed.

4. Unbiased Frobenius norm regularization of the Jacobian

Refering to equation (7), one can see that even if $\mathbf{f}$ is regularized to be small, via a kinetic energy penalty term, if the Jacobian is large then the force experienced by a particle may also still be large. As a result, the error of the numerical integrator can be large, which may lead an adaptive solver to make many function evaluations. This relationship is apparent in Figure 3, where we empirically demonstrate the correlation between the number of function evaluations of $\mathbf{f}$ taken by the adaptive solver, and the size of the Jacobian norm of $\mathbf{f}$. The correlation is remarkably strong: dynamics governed by a poorly conditioned Jacobian matrix require the adaptive solver to take many small time steps.

Moreover, in particle-based methods, the kinetic energy term forces dynamics to travel in straight lines *only on data seen during training*, and so the regularity of the map is only guaranteed on trajectories taken by training data. The issue here is one of generalization: the map may be

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

irregular on off-distribution or perturbed images, and cannot be remedied by the kinetic energy term during training alone. In the context of generalization, Jacobian regularization is analagous to gradient regularization, which has been shown to improve generalization (Drucker & LeCun, 1992; Novak et al., 2018).

For these reasons, we also propose regularizing the Jacobian through its Frobenius norm. The Frobenius norm $\|\cdot\|_F$ of a real matrix A can be thought of as the ℓ_2 norm of the matrix A vectorized

$$\|A\|_F = \sqrt{\sum_{i,j} a_{ij}^2} \tag{12}$$

Equivalently it may be computed as

$$\|A\|_F = \sqrt{\mathrm{tr}(AA^\mathsf{T})} \tag{13}$$

and is the Euclidean norm of the singular values of a matrix. In trace form, the Frobenius norm lends itself to estimation using a Monte-Carlo trace estimator (Hutchinson, 1990; Avron & Toledo, 2011). For real matrix B, an unbiased estimate of the trace is given by

$$\mathrm{tr}(B) = \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,1)}\left[\epsilon^\mathsf{T} B \epsilon\right] \tag{14}$$

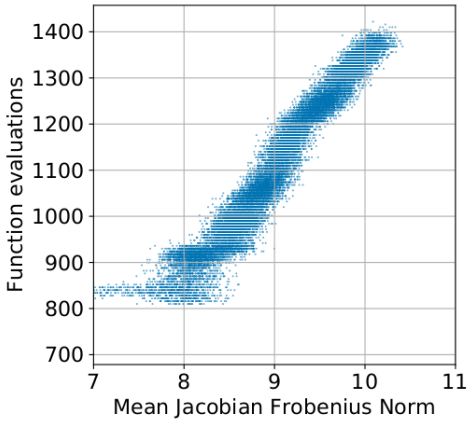


Figure 3. Number of function evaluations vs Jacobian Frobenius norm of flows on CIFAR10 during training with vanilla FFJORD, using an adaptive ODE solver.

where ϵ is drawn from a unit normal distribution. Thus the squared Frobenius norm can be easily estimated by setting $B = AA^\mathsf{T}$.

Turning to the Jacobian $\nabla \mathbf{f}(\mathbf{z})$ of a vector valued function $\mathbf{f} : \mathbb{R}^d \mapsto \mathbb{R}^d$, recall that the vector-Jacobian product $\epsilon^\mathsf{T} \nabla \mathbf{f}(\mathbf{z})$ may be quickly computed through reverse-mode automatic differentiation. Therefore an unbiased Monte-Carlo estimate of the Frobenius norm of the Jacobian is readily available

$$\|\nabla \mathbf{f}(\mathbf{z})\|_F^2 = \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,1)} \epsilon^\mathsf{T} \nabla \mathbf{f}(\mathbf{z}) \nabla \mathbf{f}(\mathbf{z})^\mathsf{T} \epsilon \tag{15}$$

$$= \mathop{\mathbb{E}}_{\epsilon\sim\mathcal{N}(0,1)} \|\epsilon^\mathsf{T} \nabla \mathbf{f}(\mathbf{z})\|^2 \tag{16}$$

Conveniently, in the FFJORD framework the quantity $\epsilon^\mathsf{T} \nabla \mathbf{f}(\mathbf{z})$ must be computed during the estimate of the probability distribution under the flow, in the Monte-Carlo estimate of the divergence term (5). Thus Jacobian Frobenius norm regularization is available with essentially *no extra computational cost*.

5. Algorithm description

All together, we propose modifying the objective function of the FFJORD continuous normalizing flow (Grathwohl et al., 2019) with the two regularization penalties of Sections 3 & 4. The proposed method is called RNODE, short for regularized neural ODE. Pseudo-code of the method is presented in Algorithm 1. The optimization problem to be

Table 1. Log-likelihood (in bits/dim) and training time (in hours) on MNIST and CIFAR10. Our implementations report results after 100 epochs of training. Results in brackets are after an additional 100 epochs. For comparison we report both the results of the original FFJORD paper and our own independent run of FFJORD ("vanilla"), as well as results reported in other flow-based generative modeling papers. Our method (FFJORD with RNODE) has comparable log-likelihood as FFJORD but is significantly faster.

	MNIST		CIFAR10	
	BITS/DIM	TIME	BITS/DIM	TIME
FFJORD, ORIGINAL (GRATHWOHL ET AL., 2019)	0.99	-	3.40	$\geq$ 5 DAYS
FFJORD, VANILLA	0.974	68.47	3.363	91.33
FFJORD RNODE (OURS)	0.973	24.37	3.398 (3.370)	31.84 (63.85)
REALNVP (DINH ET AL., 2017)	1.06	-	3.49	-
I-RESNET (BEHRMANN ET AL., 2019)	1.05	-	3.45	-
GLOW (KINGMA & DHARIWAL, 2018)	1.05	-	3.35	-
FLOW++ (HO ET AL., 2019)	-	-	3.28 / 3.09	-
RESIDUAL FLOW (CHEN ET AL., 2019)	0.970	-	3.280	-

solved is

$$\begin{aligned}\min_{\mathbf{f}} \frac{1}{Nd}\sum_{i=1}^{N} &- \log q(\mathbf{z}(\mathbf{x}_i, T)) \\ &- \int_0^T \operatorname{div}(\mathbf{f})(\mathbf{z}(\mathbf{x}_i, s), s)\,\mathrm{d}s \\ &+ \lambda_K \int_0^T \|\mathbf{f}(\mathbf{z}(\mathbf{x}_i, s), s)\|^2\,\mathrm{d}s \\ &+ \lambda_J \int_0^T \|\nabla_{\mathbf{z}}\,\mathbf{f}(\mathbf{z}(\mathbf{x}_i, s), s)\|_F^2\,\mathrm{d}s \end{aligned} \tag{17}$$

where $\mathbf{z}(\mathbf{x}, t)$ is determined by numerically solving (ODE). Note that we take the mean over number of samples *and* input dimension. This is to ensure that the choice of regularization strength λ_K and λ_J is independent of dimension size and sample size.

To compute the three integrals and the log-probability under q of $\mathbf{z}(\mathbf{x}, T)$ at final time T, we augment the dynamics of the ODE with three extra terms, so that the entire system solved by the numerical integrator is

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, t) \\ \dot{l} = \operatorname{div}(\mathbf{f})(\mathbf{z}, t) \\ \dot{E} = \|\mathbf{f}(\mathbf{z}, t)\|^2 \\ \dot{n} = \|\nabla\,\mathbf{f}(\mathbf{z}, t)\|_F^2 \\ \mathbf{z}(0) = \mathbf{x}, \quad E(0) = l(0) = n(0) = 0 \end{cases} \tag{RNODE}$$

Here E, l, and n are respectively the kinetic energy, the log determinant of the Jacobian, and the integral of the Frobenius norm of the Jacobian.

Both the divergence term and the Jacobian Frobenius norm are approximated with Monte-Carlo trace estimates. In our implementation, the Jacobian Frobenius estamate reuses the computatian $\epsilon^{\mathsf{T}}\,\nabla f$ from the divergence estimate for efficiency. We remark that the kinetic energy term only requires the computation of a dot product. Thus just as in FFJORD, our implementation scales linearly with the number of time steps taken by the ODE solver.

Gradients of the objective function with respect to the network parameters are computed using the adjoint sensitivity method (Pontryagin et al., 1962; Chen et al., 2018).

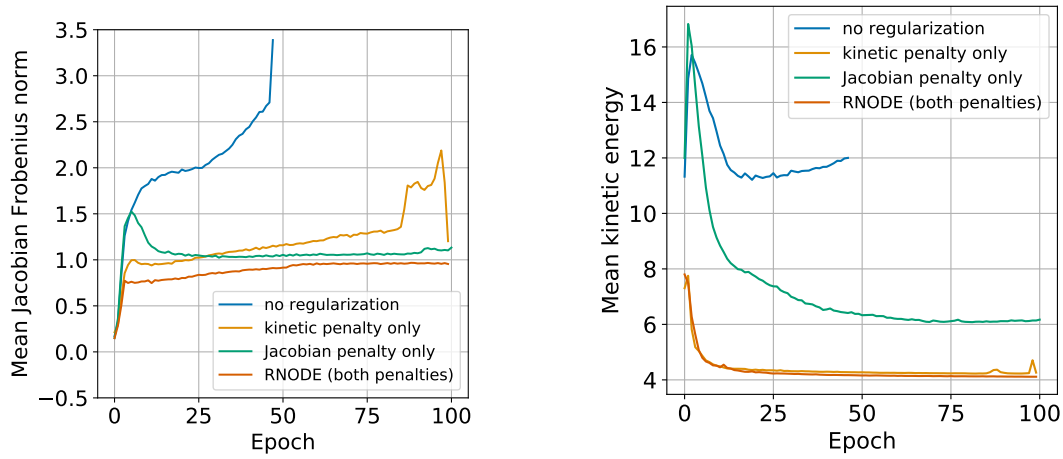
6. Experimental design

Here we demonstrate the benefits of regularizing neural ODEs on generative models for CIFAR10 (Krizhevsky & Hinton, 2009) and MNIST (LeCun & Cortes, 1998), an application where neural ODEs have shown strong empirical performance. We use an identical neural architecture to that of Grathwohl et al. (2019). The dynamics are defined by a neural network $\mathbf{f}(\mathbf{z}, t; \theta(t)) : \mathbb{R}^d \times \mathbf{R} \mapsto \mathbb{R}^d$ where $\theta(t)$ is piecewise constant in time. On MNIST we use 10 pieces; CIFAR10 uses 14. Each piece is a 4-layer deep convolutional network comprised of 3x3 kernels and softplus activation functions. Intermediary layers have 64 hidden dimensions, and time t is concatenated to the spatial input z. Weight matrices are chosen to imitate the multi-scale architecture of Real NVP (Dinh et al., 2017), in that images are 'squeezed' via a permutation to halve image height and width but quadruple the number of channels. Divergence of $\mathbf{f}$ is estimated using the Gaussian Monte-Carlo trace estimator with one sample of fixed noise per solver time-step.

We train with a batch size of 200 for 100 epochs on a single GPU[2], using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of $1\mathrm{e}{-3}$. Data is preprocessed by perturbing with uniform noise followed by the logit transform.

The reference implementation of FFJORD solves the dynamics using a Runge-Kutta 4(5) adaptive solver (Dormand

[2]GeForce RTX 2080 Ti



(a) Jacobian norm

(b) Kinetic energy

Figure 4. Ablation study of the effect of the two regularizers, comparing two measures of flow regularity during training with a fixed step-size ODE solver. Figure 4a: mean Jacobian Frobenius norm as a function of training epoch. Figure 4b: mean kinetic energy of the flow as a function of training epoch.

& Prince, 1980) with error tolerances $1e{-5}$ and initial step size $1e{-2}$. We have found that using less accurate solvers on the reference implementation of FFJORD results in numerically unstable training dynamics. In contrast, a simple fixed-grid four stage Runge-Kutta solver suffices for RNODE during training, using a step size of 0.25. The step size was determined based on a simple heuristic of starting with 0.5 and decreasing the step size by a factor of two until the discrete dynamics were stable and achieved good performance. We have also observed that RNODE improves the training time of the adaptive solvers as well, requiring many fewer function evaluations; however in Python we have found that the fixed grid solver is typically quicker at a specified number of function evaluations. At test time RNODE uses the same adaptive solver as FFJORD.

We always initialize RNODE so that $\mathbf{f}(z, t) = 0$; thus training begins with an initial identity map. The identity map is an appropriate choice because it has zero transport cost and zero Frobenius norm. Moreover the identity map is trivially solveable for any numerical solver, thus training begins without any effort required on the solver's behalf.

On MNIST both MNIST and CIFAR10 we set the kinetic energy regularization coefficient $\lambda_K = 0.01$. For the Jacobian norm regularizer, on MNIST we set $\lambda_J = 0.01$ and on CIFAR10 we set $\lambda_J = 0.03$.

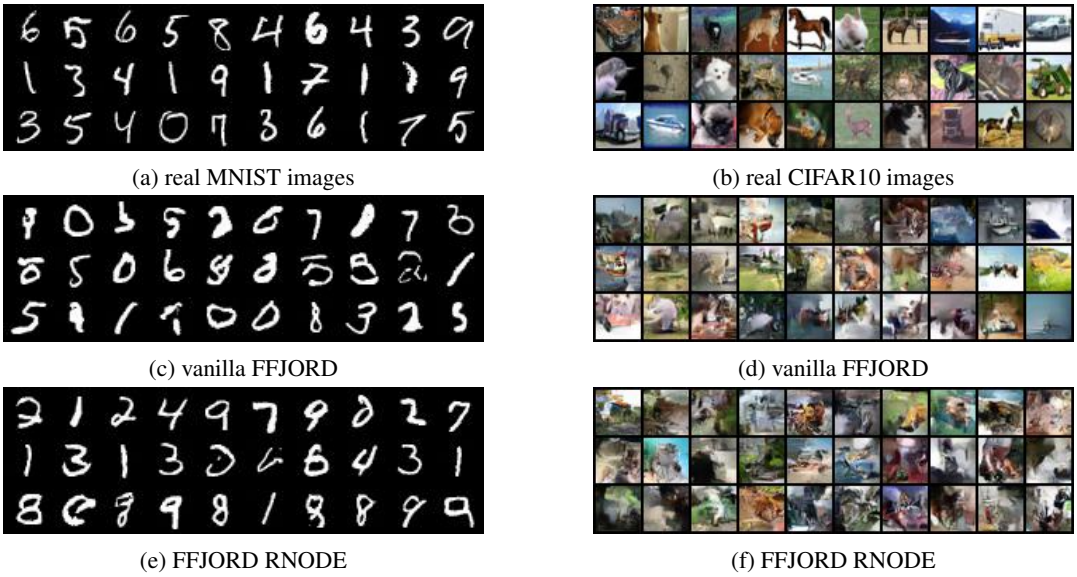
7. Results

A comparison of RNODE against FFJORD and other flow-based generative models is presented in Table 1. We report both our running of "vanilla" FFJORD and the results as originally reported in (Grathwohl et al., 2019). We highlight that RNODE runs roughly 2.8x faster than FFJORD on both datasets, while achieving or surpassing the performance of FFJORD. This can further be seen in Figure 2 where we plot bits per dimension ($-\frac{1}{d}\log_2 p(x)$, a normalized measure of log-likelihood) on the validation set as a function of training epoch, for both datasets. Visual inspection of the sample quality reveals no qualitative difference between regularized and unregularized approaches; refer to Figure 5.

Surprisingly, our run of "vanilla" FFJORD achieved slightly better performance than the results reported in (Grathwohl et al., 2019). We suspect the discrepancy in performance and run times between our implementation of FFJORD and that of the original paper is due to batch size: Grathwohl et al. use a batch size of 900 and train on six GPUs, whereas we use a batch size of 200 and train on a single GPU.

7.1. Ablation study on MNIST

In Figure 4, we compare the effect of each regularizer by itself on the training dynamics with the fixed grid ODE solver on the MNIST dataset. Without any regularization at all, training dynamics are numerically unstable and fail after just under 50 epochs. This is precisely when the Jacobian norm grows large; refer to Figure 4a. Figure 4a demonstrates that each regularizer by itself is able to control the Jacobian norm. The Jacobian regularizer is better suited to this task, although it is interesting that the kinetic energy regularizer also improves the Jacobian norm. Unsurprisingly Figure 4b demonstrates the addition of the kinetic energy regularizer encourages flows to travel a minimal distance. In addition,



(a) real MNIST images

(b) real CIFAR10 images

(c) vanilla FFJORD

(d) vanilla FFJORD

(e) FFJORD RNODE

(f) FFJORD RNODE

Figure 5. Quality of generated samples samples with and without regularization on MNIST, left, and CIFAR10, right.

we see that the Jacobian norm alone also has a beneficial effect on the distance particles travel. Overall, the results support our theoretical reasoning empirically.

8. Previous generative flows inspired by optimal transport

Zhang et al. (2018) define a neural ODE flow where the dynamics are given as the gradient of a scalar potential function. This interpretation has deep connections to optimal transport: the optimal transport map is the gradient of a convex potential function. Yang & Karniadakis (2019) continue along these lines, and define an optimal transport again as a scalar potential gradient. Yang & Karniadakis (2019) enforce that the learned map is in fact an optimal transport map by penalizing their objective function with a term measuring violations of the continuity equation. Ruthotto et al. (2019) place generative flows within a broader context of mean field games, and as an example consider a neural ODE gradient potential flow solving the optimal transport problem in up to 100 dimensions.

When a flow is the gradient of a scalar potential, the change of variables formula (4) simplifies so that the divergence term is replaced by the Laplacian of the scalar potential. Although mathematically parsimonious and theoretically well-motivated, we chose not to implement our flow as the gradient of a scalar potential function due to computational constraints: such an implementation would require 'triple backprop' (twice to compute or approximate the Laplacian, and once more for the parameter gradient). Ruthotto et al. (2019) circumvented this problem by utilizing special structural properties of residual networks to efficiently compute the Laplacian.

9. Discussion

In practice, RNODE is simple to implement, and only requires augmenting the dynamics (ODE) with two extra scalar equations (one for the kinetic energy term, and another for the Jacobian penalty). In the setting of FFJORD, because we may recycle intermediary terms used in the divergence estimate, the computational cost of evaluating these two extra equations is minimal. RNODE introduces two extra hyperparameters related to the strength of the regularizers; we have found these required almost no tuning.

Although the problem of classification was not considered in this work, we believe RNODE may offer similar improvements both in training time and the regularity of the classifier learned. In the classification setting we expect the computional overhead of calculating the two extra terms should be marginal relative to gains made in training time.

10. Conclusion

We have presented RNODE, a regularized method for neural ODEs. This regularization approach is theoretically well-motivated, and encourages neural ODEs to learn well-behaved dynamics. As a consequence, numerical integration of the learned dynamics is straight forward and relatively easy, which means fewer discretizations are needed to solve the dynamics. In many circumstances, this allows for the replacement of adaptive solvers with fixed grid solvers, which can be more efficient during training. This leads to a substantial speed up in training time, while still maintaining

the same empirical performance, opening the use of neural ODEs to large-scale applications.

11. Acknowledgements

L. N. was supported by AFOSR MURI FA9550-18-1-0502, AFOSR Grant No. FA9550-18-1-0167, and ONR Grant No. N00014-18-1-2527.

A. O was supported by the Air Force Office of Scientific Research under award number FA9550-18-1-0167

References

Avron, H. and Toledo, S. Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix. *J. ACM*, 58(2):8:1–8:34, 2011. doi: 10.1145/1944345.1944349. URL https://doi.org/10.1145/1944345.1944349.

Behrmann, J., Grathwohl, W., Chen, R. T. Q., Duvenaud, D., and Jacobsen, J. Invertible residual networks. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 573–582. PMLR, 2019. URL http://proceedings.mlr.press/v97/behrmann19a.html.

Benamou, J.-D. and Brenier, Y. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

Chen, T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 6572–6583, 2018. URL http://papers.nips.cc/paper/7892-neural-ordinary-differential-equations.

Chen, T. Q., Behrmann, J., Duvenaud, D., and Jacobsen, J. Residual flows for invertible generative modeling. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pp. 9913–9923, 2019. URL http://papers.nips.cc/paper/9183-residual-flows-for-invertible-generative-modeling.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Dormand, J. R. and Prince, P. J. A family of embedded Runge-Kutta formulae. *Journal of computational and applied mathematics*, 6(1):19–26, 1980.

Drucker, H. and LeCun, Y. Improving generalization performance using double backpropagation. *IEEE Trans. Neural Networks*, 3(6):991–997, 1992. doi: 10.1109/72.165600. URL https://doi.org/10.1109/72.165600.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. FFJORD: free-form continuous dynamics for scalable reversible generative models. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

Haber, E. and Ruthotto, L. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2722–2730. PMLR, 2019. URL http://proceedings.mlr.press/v97/ho19a.html.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In Bengio, S., Wallach,

H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pp. 10236–10245, 2018. URL http://papers.nips.cc/paper/8224-glow-generative-flow-with-invertible-1x1-convolutions.

Köhler, J., Klein, L., and Noé, F. Equivariant flows: sampling configurations for multi-body systems with symmetric energies. *arXiv preprint arXiv:1910.00753*, 2019.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL http://www.cs.toronto.edu/~kriz/cifar.html.

LeCun, Y. and Cortes, C. The MNIST database of handwritten digits. 1998. URL http://yann.lecun.com/exdb/mnist/.

Li, X., Wong, T. L., Chen, R. T. Q., and Duvenaud, D. Scalable gradients for stochastic differential equations. *CoRR*, abs/2001.01328, 2020. URL http://arxiv.org/abs/2001.01328.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and generalization in neural networks: an empirical study. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=HJC2SzZCW.

Pontryagin, L. S., Mishchenko, E., Boltyanskii, V., and Gamkrelidze, R. The mathematical theory of optimal processes. 1962.

Rubanova, Y., Chen, T. Q., and Duvenaud, D. K. Latent ordinary differential equations for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, pp. 5321–5331, 2019.

Ruthotto, L. and Haber, E. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pp. 1–13, 2018.

Ruthotto, L., Osher, S. J., Li, W., Nurbekyan, L., and Fung, S. W. A machine learning framework for solving high-dimensional mean field game and mean field control problems. *CoRR*, abs/1912.01825, 2019. URL http://arxiv.org/abs/1912.01825.

Santambrogio, F. *Benamou-Brenier and other continuous numerical methods*, pp. 219–248. Springer International Publishing, Cham, 2015. ISBN 978-3-319-20828-2. doi: 10.1007/978-3-319-20828-2_6. URL https://doi.org/10.1007/978-3-319-20828-2_6.

Villani, C. *Topics in Optimal Transportation*. Graduate studies in mathematics. American Mathematical Society, 2003. ISBN 9780821833124.

Villani, C. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008. ISBN 9783540710509. URL https://books.google.ca/books?id=hV8o5R7_5tkC.

Yang, L. and Karniadakis, G. E. Potential flow generator with L_2 Optimal Transport regularity for generative models. *CoRR*, abs/1908.11462, 2019. URL http://arxiv.org/abs/1908.11462.

Zhang, L., E, W., and Wang, L. Monge-Ampère flow for generative modeling. *CoRR*, abs/1809.10188, 2018. URL http://arxiv.org/abs/1809.10188.