Enhanced Distribution Modelling via Augmented Architectures For Neural ODE Flows

Etrit Haxholli, Marco Lorenzi

Augmented Neural ODE Flows

Anonymous Author
Anonymous Institution

Abstract

Normalizing flows based on neural ODEs, as implemented in FFJORD, provide a powerful theoretical framework for density estimation and data generation. While the neural ODE formulation enables us to calculate the determinants of free form Jacobians in $\mathcal{O}(D)$ time, the flexibility of the transformation underlying neural ODEs has been shown to be suboptimal. In this paper, we present AFFJORD, a neural ODE-based normalizing flow which enhances the representation power of FFJORD by defining the neural ODE through augmented transformation dynamics. To derive the Jacobian determinant of the general augmented form, we generalise the chain rule in the continuous sense into the *cable rule*, which expresses the forward sensitivity of ODEs with respect to their initial conditions. The cable rule gives an explicit expression for the Jacobian of a neural ODE transformation, and provides an elegant proof of the instantaneous change of variable. Our experimental results on density estimation in synthetic and high dimensional data, such as MNIST and CIFAR-10, show that AFFJORD outperforms the baseline FFJORD through the improved flexibility of the underlying vector field.

1 INTRODUCTION

Normalizing flows are diffeomorphic random variable transformations providing a powerful theoretical framework for generative modeling and probability density estimation (Rezende & Mohamed, 2015). While the practical application of normalizing flows is generally challenging due to computational bottlenecks, most notably regarding the $\mathcal{O}(D^3)$ computation cost of the Jacobian determinant, different architectures have been proposed in order to scale normalizing flows to high dimensions while at the same time ensuring the flexibility and bijectivity of the transformations (Rezende & Mohamed, 2015; Dinh et al., 2016; Kobyzev et al., 2021). The common strategy consists of placing different architectural restrictions on the model, to enforce special Jacobian forms, with less computationally demanding determinants.

A noteworthy approach is based on neural ODEs, (Chen et al., 2018), as they enable us to calculate the determinants of free form Jacobians in $\mathcal{O}(D)$ time, (Grathwohl et al., 2018). More specifically, the rule for the instantaneous change of variable (Chen et al., 2018) provides an important theoretical contribution to normalizing flows, as it yields a closed form expression of the Jacobian determinant of a neural ODE transformation.

In this case, calculating the Jacobian determinant simplifies to calculating the integral of the divergence of the vector field along the transformation trajectory. Such models are known as Continuous Normalizing Flows (CNFs). In (Grathwohl et al., 2018), these ideas are further explored and computational simplifications are introduced, notably the use of Hutchinson's trace estimator, (Hutchinson, 1990). The resulting model is named FFJORD.

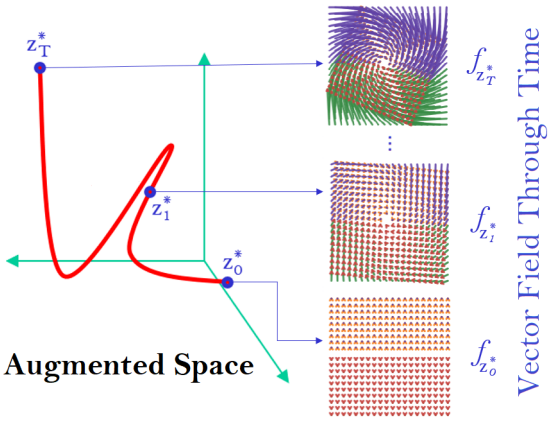

Figure 1: Vector field evolution in AFFJORD as a function of the augmented dimensions $\boldsymbol{z}_t^*$.

(Dupont et al., 2019) show that there exist functions which neural ODEs are not capable of representing. To tackle this

Preliminary work. Under review by AISTATS 2023. Do not distribute.

issue and enhance the expressiveness of the neural ODE transformation, they propose to lift the data into a higher dimensional space, on which the neural ODE is applied, to subsequently project the output back to the original space. Such augmented neural ODEs (ANODEs) have been experimentally shown to lead to improved flexibility and generalisation properties than the non-augmented counterpart.

Inspired by (Dupont et al., 2019), in this paper we develop a theoretical framework to increase the flexibility of ODE flows, such as FFJORD, through dimension augmentation. To this end, we derive an explicit formula for Jacobians corresponding to neural ODE transformations. This formula represents the continuous generalization of the chain rule, which we name *the cable rule*, that ultimately allows the derivation of the Jacobian expression and determinant for the composition of the operations defining the ANODE flow: augmentation, neural ODE transformation, and projection.

To enable computational feasibility and ensure that the ANODE transformation is diffeomorphic, we allow the augmented dimensions to parameterize the vector field acting on the original dimensions (but not vice-versa). We name the resulting model augmented FFJORD (AFFJORD). In AFFJORD, the evolution of time is high dimensional, and is learnt via the vector field defined by the augmented component, contrasting the linear time evolution of FFJORD. This setup coincides with the framework introduced by (Zhang et al., 2019), but in the context of normalizing flows. Our experiments on 2D data of toy distributions and image datasets such as MNIST and CIFAR-10, show that the proposed ANODE flow defined by AFFJORD significantly outperforms FFJORD in terms of density estimation, thus highlighting the improved fexibility and representation properties of the underlying vector field.

2 BACKGROUND AND RELATED WORK

Normalizing Flows: The Normalizing Flow framework was previously defined in (Tabak & Vanden-Eijnden, 2010; Tabak & Turner, 2013), and was popularised by (Rezende & Mohamed, 2015) and by (Dinh et al., 2014) respectively in the context of variational inference and density estimation. A Normalizing Flow is a transformation defined by a sequence of invertible and differentiable functions mapping a simple base probability distribution (e.g., a standard normal) into a more complex one. Let $\boldsymbol{Z}$ and $\boldsymbol{X} = g(\boldsymbol{Z})$ be random variables where g is a diffeomorphism with inverse h. If we denote their probability density functions by $f_{\boldsymbol{Z}}$ and $f_{\boldsymbol{X}}$, based on the change of variable theorem we get:

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = f_{\boldsymbol{Z}}(\boldsymbol{z})|\frac{d\boldsymbol{z}}{d\boldsymbol{x}}| = f_{\boldsymbol{Z}}(h(\boldsymbol{x}))|\frac{dh(\boldsymbol{x})}{d\boldsymbol{x}}|. \quad (1)$$

In general, we want to optimize the parameters of h such that we maximize the likelihood of sampled points $\boldsymbol{x}_1, ..., \boldsymbol{x}_n$. Once these parameters are optimized, then we can give as input any test point $\boldsymbol{x}$ on the right hand side of Equation 1, and calculate its probability density. For the generative task, being able to easily recover g from h is essential, as the generated point $\boldsymbol{x}_g$ will take form $\boldsymbol{x}_g$=$g(\boldsymbol{z}_s)$, where $\boldsymbol{z}_s$ is a sampled point from the base distribution $f_{\boldsymbol{Z}}$.
For increased modeling flexibility, we can use a chain (flow) of transformations, $\boldsymbol{z}_i = g_i(\boldsymbol{z}_{i-1}), i \in [n]$. In this case due to chain rule we have:

$$f_{\boldsymbol{Z}_n}(\boldsymbol{z}_n) = f_{\boldsymbol{Z}_0}(\boldsymbol{z}_0)|\frac{d\boldsymbol{z}_0}{d\boldsymbol{z}_n}| =$$

$$f_{\boldsymbol{Z}_0}(\boldsymbol{z}_0)|\frac{d\boldsymbol{z}_0}{d\boldsymbol{z}_1}||\frac{d\boldsymbol{z}_1}{d\boldsymbol{z}_2}|...|\frac{d\boldsymbol{z}_{n-1}}{d\boldsymbol{z}_n}|$$

$$= f_{\boldsymbol{Z}_0}(h_0(...h_n(\boldsymbol{z}_n))\Pi_{i=1}^n|\frac{dh_i(\boldsymbol{z}_i)}{d\boldsymbol{z}_i}| \quad (2)$$

The interested reader can find a more in-depth review of normalizing flows in (Kobyzev et al., 2021) and (Papamakarios et al., 2021).

Neural ODE Flows: Neural ODEs, (Chen et al., 2018), are continuous generalizations of residual networks:

$$\boldsymbol{z}_{t_{i+1}} = \boldsymbol{z}_i + \epsilon f(\boldsymbol{z}_{t_i}, t_i, \boldsymbol{\theta})$$

$$\rightarrow \boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau), \tau, \boldsymbol{\theta})d\tau, \text{ as } \epsilon \rightarrow 0. \quad (3)$$

(Pontrjagin et al., 1962; Chen et al., 2018) show that the gradients of neural ODEs can be computed via the adjoint method, with constant memory cost with regards to "depth":

$$\frac{dL}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}dt =$$

$$= -\int_T^0 \frac{\partial L}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}dt, \quad (4)$$

where $\frac{\partial L}{\partial \boldsymbol{z}(t)}$ can be calculated simultaneously by

$$\frac{\partial L}{\partial \boldsymbol{z}(t)} = \frac{\partial L}{\partial \boldsymbol{z}(T)} - \int_T^t \frac{\partial L}{\partial \boldsymbol{z}(\tau)}\frac{\partial f(\boldsymbol{z}(\tau), \tau, \boldsymbol{\theta}(\tau))}{\partial \boldsymbol{z}(\tau)}d\tau. \quad (5)$$

In addition, they also derive the expression for the instantaneous change of variable, which enables one to train continuous normalizing flows:

$$\log p(\boldsymbol{z}(0)) = \log p(\boldsymbol{z}(T)) + \int_0^T tr\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}dt, \quad (6)$$

where $\boldsymbol{z}(0)$ represents a sample from the data. A surprising benefit is that one does not need to calculate the determinant of the Jacobian of the transformation anymore, but simply the trace of a matrix. These models are collectively called Neural ODE flows (NODEFs) or simply Continuous Normalizing Flows (CNFs). In (Grathwohl et al., 2018), these ideas are further explored and computational simplifications

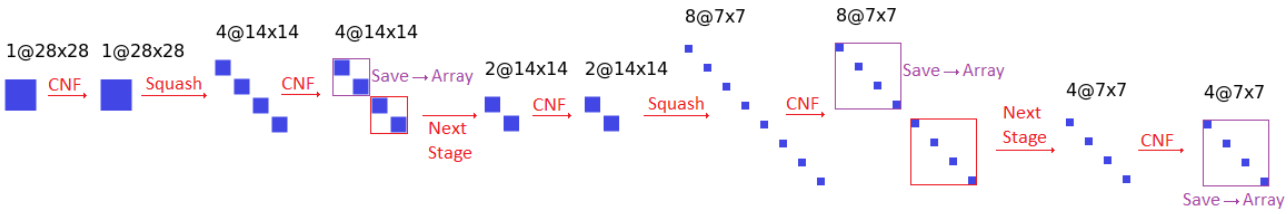

Figure 2: Example of the multiscale architecture on MNIST dataset

are introduced, notably the use of Hutchinson's trace estimator, (Hutchinson, 1990; Adams et al., 2018), as an unbiased stochastic estimator of the trace in the likelihood expression in Equation 6. The resulting model is named FFJORD.

Multiscale Architectures: In (Dinh et al., 2016) a multiscale architecture for normalizing flows is implemented, which transforms the data shape from $[c, s, s]$ to $[4c, \frac{s}{2}, \frac{s}{2}]$, where c is the number of channels and s is the height and width of the image. Effectively this operation trades spatial size for additional channels, and after each transformation a normalizing flow is applied on half the channels, while the other half are saved in an array. This process can be repeated as many times as the width and height of the transformed image remain even numbers. In the end, all saved channels are concatenated to construct an image with the original dimensions. A visual description of this process can be found in Figure 2.

Augmented Neural ODEs: Considering that transformations by neural ODEs are diffeomorphisms (hence homeomorphisms), (Dupont et al., 2019) show that Neural Ordinary Differential Equations (NODEs) learn representations that preserve the topology of the input space, and prove that this implies the existence of functions that Neural ODEs cannot represent. To address these limitations, they introduce the Augmented Neural ODEs:

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{z}(t)\\ \boldsymbol{z}^*(t)\end{bmatrix} = h\left(\begin{bmatrix}\boldsymbol{z}(t)\\ \boldsymbol{z}^*(t)\end{bmatrix}, t\right) = \begin{bmatrix}f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})\\ g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})\end{bmatrix}$$
$$\text{for } \begin{bmatrix}\boldsymbol{z}(0)\\ \boldsymbol{z}^*(0)\end{bmatrix} = \begin{bmatrix}\boldsymbol{x}\\ 0\end{bmatrix}, \tag{7}$$

where $\boldsymbol{z}^*(t)$ is the augmented component, and $h = [f, g]$ is the vector field to be learnt.

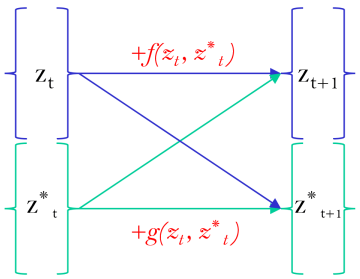

Figure 3: Architecture of discretized augmented neural ODEs.

In addition to being more expressive models, (Dupont et al., 2019) show that augmented neural ODEs are empirically more stable, generalize better and have a lower computational cost than Neural ODEs. A schematic of their architecture in the discrete case can be found in Figure 3.

3 PROPOSED FRAMEWORK

In the first subsection, we introduce our model AFFJORD, which is a special case of augmented neural ODEs. In the second subsection we give the generalisation of the chain rule in the continuous sense which we refer to as the cable rule, which is analogous to forward sensitivity. Next we give an intuitive proof of the continuous backpropagation, by showing its equivalence to the continuous generalisation of the total derivative decomposition. Then, using the cable rule, we give a more detailed explanation on why the instantaneous change of variable still holds in our model. In the final section, the augmented multiscale architecture is described, as it is implemented in the experimental section.

3.1 Proposed Model: Augmented FFJORD (AFFJORD)

A neural ODE of the type $f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})$, where $\boldsymbol{z}^*(t) = \boldsymbol{z}^*(t) + \int_0^t g(\boldsymbol{z}^*(\tau), \boldsymbol{\phi})d\tau$, can equivalently be written as $h_f(\boldsymbol{z}(t), t, \boldsymbol{\theta}) = f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})$. Thus, motivated by (Dupont et al., 2019), we propose to lift each data point $\boldsymbol{z}(0) \in \mathbb{R}^n$ to a higher dimensional space $\mathbb{R}^{n+m}$, by augmentation via an m dimensional vector $\boldsymbol{z}^*(0)$, which in practice is set to be the zero vector. Therefore, the joint vector $[\boldsymbol{z}(0), \boldsymbol{z}^*(0)]$ is transformed by the vector field $h = (f, g)$:

$$w(T) = \begin{bmatrix}\boldsymbol{z}(T)\\ \boldsymbol{z}^*(T)\end{bmatrix} = \begin{bmatrix}\boldsymbol{z}(0)\\ \boldsymbol{z}^*(0)\end{bmatrix} + \int_0^T \begin{bmatrix}f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})\\ g(\boldsymbol{z}^*(t), \boldsymbol{\phi})\end{bmatrix} dt. \tag{8}$$

We are not interested in $z^*(T)$ as our interest lies on the transformed $z(0)$, that is $z(T)$. Since by definition such coupled dynamics are contained in the formulation of neural ODEs, this implies that the instantaneous change of formula still holds, and the transformation is injective. This sort of augmentation can be seen as a special case of augmentation introduced in (Dupont et al., 2019), where the augmented dimensions depend only on themselves.

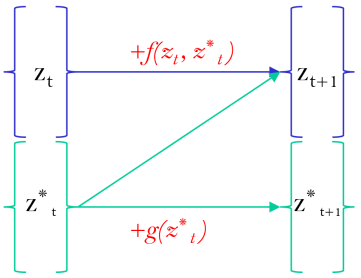

Figure 4: Architecture of the discretized augmented neural ODE flows implemented in AFFJORD.

The augmented dimensions $\boldsymbol{z}^*(t)$ can also be seen as time dependent weights of the non-autonomous f whose evolution is determined by the autonomous ODE g, hence giving f greater flexibility in time (Zhang et al., 2019).

3.2 The Continuous Generalization of the Chain Rule

If we define the transformation $\boldsymbol{z}_i = f_i(\boldsymbol{z}_{i-1})$ for $i \in \{1, ..., n\}$, due to the chain rule we have:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{z}_0} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}_{n-1}} \frac{d\boldsymbol{z}_{n-1}}{d\boldsymbol{z}_0} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}_{n-1}} \frac{\partial \boldsymbol{z}_{n-1}}{\partial \boldsymbol{z}_{n-2}} ... \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}_0} = \\ = \frac{\partial f_n(\boldsymbol{z}_{n-1})}{\partial \boldsymbol{z}_{n-1}} \frac{\partial f_{n-1}(\boldsymbol{z}_{n-2})}{\partial \boldsymbol{z}_{n-2}} ... \frac{\partial f_1(\boldsymbol{z}_0)}{\partial \boldsymbol{z}_0}. \tag{9}$$

The continuous counterpart of this transformation can be expressed as: $\boldsymbol{z}_i = \boldsymbol{z}_{i-1} + \epsilon f(\boldsymbol{z}_{i-1}, t_{i-1}, \boldsymbol{\theta})$. It is clear that $\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(t), t, \boldsymbol{\theta}) d\tau$ is the limit of the previous iterative definition when $\epsilon \to 0$. Then the expression $\frac{d\boldsymbol{z}_n}{d\boldsymbol{z}_0}$ converges to $\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$, which as we show in Appendix A, satisfies the differential equation below:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{10}$$

Using the Magnus expansion, (Magnus, 1954; Blanes et al., 2009):

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(T)}, \text{ for } \boldsymbol{\Omega}(t) = \sum_{k=1}^{\infty} \boldsymbol{\Omega}_k(t), \tag{11}$$

where $\boldsymbol{\Omega}_i(t)$ are the terms of the Magnus expansion (See Appendix A). As this expression gives the generalisation of the chain rule in the continuous sense, we refer to it as the cable rule. We notice that Equation 10 gives the dynamics of the Jacobian of the state with respect to the initial condition $\boldsymbol{z(0)}$. The cable rule is therefore analogous to the standard forward sensitivity formula for ODEs which provides the Jacobian with respect to the parameters $\boldsymbol{\theta}$ of the flow (Zhao & Mousseau, 2013). This relation is highlighted and explained in more detail in Appendix A. On this note, in Appendix G, we derive the cable rule via Equation 5. Furthermore, in Appendix E, we derive the instantaneous change of variables from the cable rule.

3.3 The Continuous Generalisation of the Total Derivative Decomposition and Continuous Backpropagation

In the case that $f = f(\boldsymbol{x}(\boldsymbol{\theta}), \boldsymbol{y}(\boldsymbol{\theta}))$, then $\frac{df}{d\boldsymbol{\theta}} = \frac{\partial f}{\partial \boldsymbol{x}} \frac{d\boldsymbol{x}}{d\boldsymbol{\theta}} + \frac{\partial f}{\partial \boldsymbol{y}} \frac{d\boldsymbol{y}}{d\boldsymbol{\theta}}$, as $\boldsymbol{\theta}$ contributes to both $\boldsymbol{x}$ and $\boldsymbol{y}$, which in turn determine f. If $\boldsymbol{z}(T) = \boldsymbol{z}(0) + \int_0^T f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t) dt$, then $\boldsymbol{\theta}$ controls the vector field at each time point during integration, hence we expect that these infinitesimal contributions of the transformation from $\boldsymbol{z}(0)$ to $\boldsymbol{z}(T)$ should be integrated. Indeed, as we prove in Appendix B, the following holds:

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}} dt, \tag{12}$$

from which for $\boldsymbol{\theta}(t) = \boldsymbol{\theta}$, and for some function $L = L(\boldsymbol{z}(T))$, we deduce:

$$\frac{dL}{d\boldsymbol{\theta}} = \frac{\partial L}{\partial \boldsymbol{z}(T)} \frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(T)} \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt \tag{13}$$

thus giving an alternative and intuitive proof of continuous backpropagation, (Pontrjagin et al., 1962; Chen et al., 2018). In Appendix H, we show that the adjoint method can be used to prove Equation 12, hence the continuous total derivative decomposition is equivalent to continuous backpropagation. An alternate derivation of Equation 12 can be found in the Appendix of (Massaroli et al., 2020)

3.4 AFFJORD as a Special Case of Augmented Neural ODE Flows

As discussed in subsection 3.1, the ODE dynamics in Equation 8, can be seen as a special case of the following joint ODE transformation:

$$w(T) = \begin{bmatrix} \boldsymbol{z}(T) \\ \boldsymbol{z}^*(T) \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}(0) \\ \boldsymbol{z}^*(0) \end{bmatrix} + \int_0^T \begin{bmatrix} f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta}) \\ g(\boldsymbol{z}^*(t), \boldsymbol{z}(t), \boldsymbol{\phi}) \end{bmatrix} dt. \tag{14}$$

In this general form, the model is unsuitable to be used in practice for two reasons:
1) The transformation of $\boldsymbol{z}(0)$ to $\boldsymbol{z}(T)$ is not necessarily injective. Indeed, the transformation from $[\boldsymbol{z}(0), \boldsymbol{z}^*(0)]$ to $[\boldsymbol{z}(T), \boldsymbol{z}^*(T)]$ is injective due to the Picard–Lindelöf Theorem, however, for two data points $\boldsymbol{z'}(0)$ and $\boldsymbol{z''}(0)$, their images $\boldsymbol{z'}(T)$, $\boldsymbol{z''}(T)$ might be identical as long as their respective augmented dimensions $\boldsymbol{z'}^*(T)$, $\boldsymbol{z''}^*(T)$ differ.
2) The Jacobian determinant of this general transformation is computationally intractable. Using the chain rule we can express the Jacobian determinant of this transformation as

$$\left|\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right| = \\ \left|\frac{d\boldsymbol{z}(T)}{d\left[\boldsymbol{z}(T), \boldsymbol{z}^*(T)\right]} \frac{d\left[\boldsymbol{z}(T), \boldsymbol{z}^*(T)\right]}{d\left[\boldsymbol{z}(0), \boldsymbol{z}^*(0)\right]} \frac{d\left[\boldsymbol{z}(0), \boldsymbol{z}^*(0)\right]}{d\boldsymbol{z}(0)}\right|. \tag{15}$$

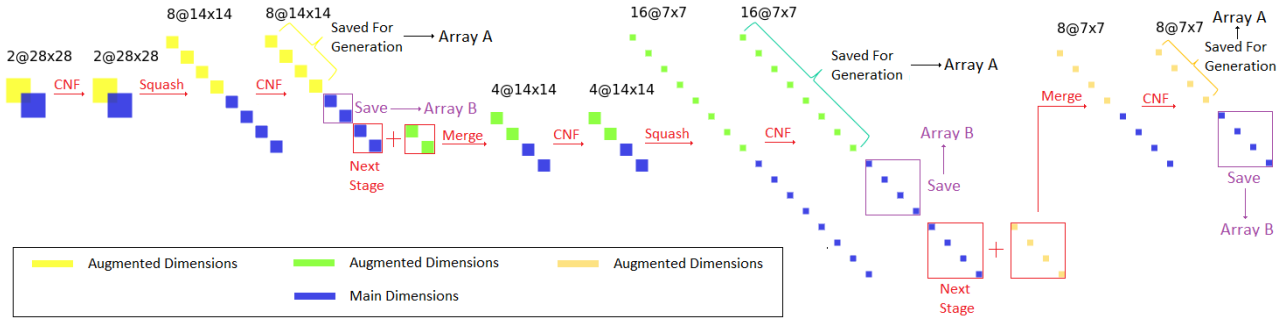

Figure 5: Example of the augmented multiscale architecture on MNIST dataset

The middle term on the RHS of Equation 15 can be further developed via the cable rule, to give the general expression of the determinant of the Jacobian of augmented neural ODE flows, which is not computationally feasible in general.

As explained in Section 3.1, the special case (AFFJORD) formulated in Equation 8, mitigates the issues mentioned above. Regarding issue 1), we have the following:

Proposition 3.1. *The architecture of AFFJORD ensures that the transformation is injective.*

Proof. Indeed, as $\boldsymbol{z}^*(t)$ is not dependent on external factors, and since $\boldsymbol{z}^*(0)$ is constant regarding $\boldsymbol{z}(0)$, the end result $\boldsymbol{z}^*(T)$ will always be the same. Hence, for $\boldsymbol{z'}(0)$ and $\boldsymbol{z''}(0)$ their images $\boldsymbol{z'}(T)$ and $\boldsymbol{z''}(T)$ must be different, since their equality would imply $[\boldsymbol{z'}(T), \boldsymbol{z}^*(T)] = [\boldsymbol{z''}(T), \boldsymbol{z}^*(T)]$ contradicting the Picard–Lindelöf Theorem. □

Issue 2) is mitigated as the right side of Equation 15 simplifies to

$$\left| [I, 0] \begin{bmatrix} e^{\int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt + \boldsymbol{\Omega}_2^{[z]}(T) + \dots} & \bar{\boldsymbol{B}}(T) \\ 0 & \bar{\boldsymbol{D}}(T) \end{bmatrix} \begin{bmatrix} I \\ 0 \end{bmatrix} \right|,$$

for two block matrices $\bar{\boldsymbol{B}}(T)$ and $\bar{\boldsymbol{D}}(T)$. This is proven in Appendix F. In this case, the $[z]$ in $\boldsymbol{\Omega}_2^{[z]}(T)$ denotes the restriction of the Magnus expansion to the original dimensions. In case that the base distribution is multivariate normal, from

$$-\log p(\boldsymbol{z}(0)) = -\log \left[p(\boldsymbol{z}(T)) \left| e^{\int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt + \dots} \right| \right]$$

we derive the following loss function:

$$L = \frac{||\boldsymbol{Z}(T)||^2}{2} - \int_0^T tr \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t))}{\partial \boldsymbol{z}(t)} dt. \qquad (16)$$

3.5 Multiscale Architecture in Augmented Neural ODE Flows

With reference to Figure 5, the multiscale architecture in the augmented case performs an Augmented Continuous Flow on the data as described in Subsection 3.1, then squeezes the channels (the augmented as well as the original channels) as in the original multiscale architecture. However after the second transformation the augmented channels are removed and stored in a separated array (Array A). The data channels are treated as before, that is, half of them are saved (Array B), and the other half are squeezed again. After this process is finished we add new augmented channels, to repeat the cycle. Note that in order to generate data by the inverse transformation, we need to retrieve the transformed augmented dimensions previously stored (i.e., Array A).

4 EXPERIMENTS

We compare the performance of AFFJORD with respect to the base FFJORD on 2D data of toy distributions, as well as on standard benchmark datasets such as MNIST and CIFAR-10. In the case of the 2D data, we use the implementation of CNFs provided in (Chen et al., 2018), since using the Hutchinson's trace estimator and GPUs as in FFJORD provides no computational benefits in low dimensions. In this case we also use the non-adaptive Runge-Kutta 4 ODE solver, while for MNIST and CIFAR-10 we use Dopri5, as well as the FFJORD implementation of (Grathwohl et al., 2018). We use a batch size of 200 for MNIST and CIFAR-10 and a batch size of 512 for the toy datasets. In all cases, we use a learning rate of 10^{-3}. All experiments were performed on a single GPU.

4.1 Toy 2D Datasets

In order to visualise the performance of the model, we first test FFJORD and AFFJORD on 2D data of toy distributions, depicted in Figure 6. In both cases we use the Runge-Kutta 4 solver with 20 time steps (80 function evaluations). In these examples, we used a hypernet architecture, where the augmented dimensions were fed to a hypernet (denoted as hyp) in order to generate the weights of the field of the main dimensions. The expression of the vector field to be learnt is the following: $\frac{d\boldsymbol{z}}{dt} = f\big([\boldsymbol{z}(t), \boldsymbol{z}^*(t)], \boldsymbol{\theta}(t) = hyp(\boldsymbol{z}^*(t), \boldsymbol{w})\big)$, where $\frac{d\boldsymbol{z}^*(t)}{dt} = g(\boldsymbol{z}^*(t), \boldsymbol{\phi})$. Thus, the learnable parameters are $\boldsymbol{w}$ and $\boldsymbol{\phi}$.

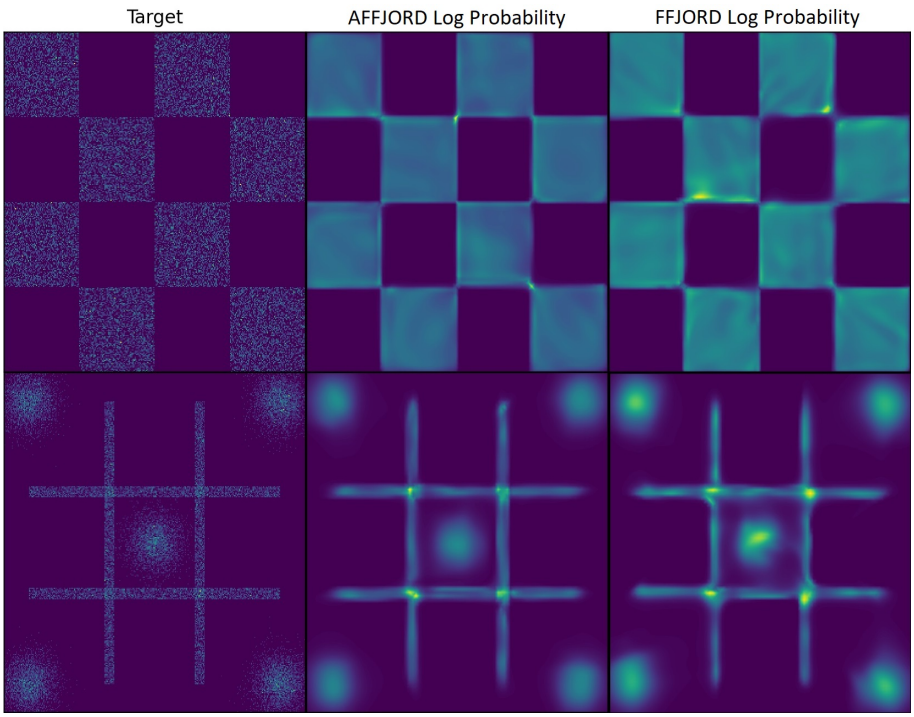

Figure 6: Probability density modeling capabilities of FFJORD (third column) and AFFJORD (second column) on 2D data of toy distributions.

While both FFJORD and AFFJORD are capable of modelling multi-modal and discontinuous distributions, Figure 6 shows that AFFJORD has higher flexibility in modeling the complex data distributions considered, in comparison to FFJORD. In the first row, the target is the checkerboard distribution, where the datapoints in each square are distributed uniformly. AFFJORD is more capable of capturing this characteristic of the data.

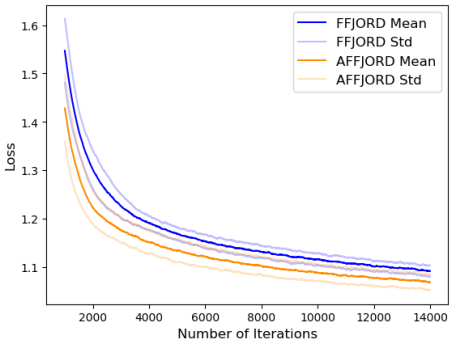

Figure 7: Performance of FFJORD and AFFJORD on the Hash-Gaussian toy 2D dataset.

On the second row AFFJORD is capable of separating the Gaussian distribution in the center from the square that surrounds it. Furthermore, it separates the Gaussians in the corners from the hash symbol properly. In Figures 7 and 8 , we show the results of the validation loss per iteration for both models. We can notice that the loss of our model is roughly two standard deviations lower than the one of FFJORD. For each model the experiment was repeated 30 times. The experiments provided here show that AFFJORD is characterized by high flexibility of the vector field. Indeed, in FFJORD the vector field changes more slowly, whereas in AFFJORD the field is able to change almost abruptly, due to this greater flexibility in time [1].

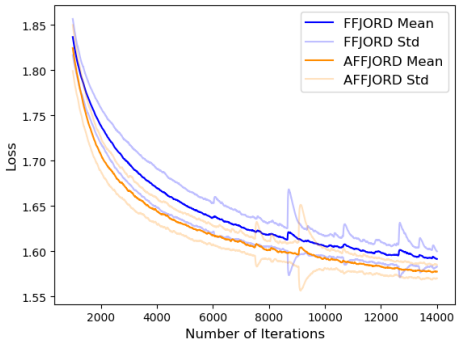

Figure 8: Performance of FFJORD and AFFJORD on the Checkerboard toy 2D dataset.

[1]Videos showing the comparison of dynamics between FFJORD and AFFJORD can be found at https://imgur.com/gallery/kMGCKve

It is important to emphasize that AFFJORD retains the ability to generate samples from the learnt distribution, by simply integrating in the opposite direction. Indeed, for a given sample $\boldsymbol{z}_s(T)$ from the base distribution, we augment it to $\boldsymbol{z}^*(T)$, as $\boldsymbol{z}^*(T)$ is the same for all data points. Then, we can simply integrate backwards this concatenated vector $[\boldsymbol{z}_s(T), \boldsymbol{z}^*(T)]$ to $[\boldsymbol{z}_s(0), \boldsymbol{z}^*(0)]$, drop the generated augmented dimensions $\boldsymbol{z}^*(0)$, and simply keep the generated data point $\boldsymbol{z}_s(0)$.

4.2 MNIST and CIFAR-10

We show that AFFJORD outperforms FFJORD on MNIST and CIFAR-10. There are several architectures of FFJORD that can be used for this application. Out of the architectures that we tested, the one that performed best was the multi-scale one, with three convolutional layers with 64 channels each. The number of CNF blocks was 1, and time was implemented by simply concatenating it as a channel into the data.

Table 1: Experimental Results for Density Estimation Models, in Bits/Dim for MNIST and CIFAR-10. Lower Is Better. The Multiscale Architecture Is Used in All Cases. In Parenthesis We Give the Best Results Out of All Repetitions of the Experiment.

MODEL	MNIST	CIFAR10
Real NVP	1.06	3.49
Glow	1.05	3.35
FFJORD	**0.98** (0.98)	3.37 (3.37)
AFFJORD (ours)	**0.98** (**0.97**)	**3.33** (**3.30**)

When time was implemented via a hypernet, we observed that the training time increased and performance decreased, especially in the case of CIFAR-10. For AFFJORD we use the exact same base architecture, however, as in the case of 2D toy data, we add a hypernet which takes the augmented dimensions as an input, and outputs the weights of the main component. The augmented dimensions are concatenated as a channel to the main channel of the data. The formulas for both the main field and the augmented component remain unchanged from the case of the 2D toy data, that is, $\frac{d\boldsymbol{z}}{dt} = f\big([\boldsymbol{z}(t), \boldsymbol{z}^*(t)], \boldsymbol{\theta}(t) = hyp(\boldsymbol{z}^*(t), \boldsymbol{w})\big)$ and $\frac{d\boldsymbol{z}^*(t)}{dt} = g(\boldsymbol{z}^*(t), \boldsymbol{\phi})$. The main difference here is that $g(\boldsymbol{z}^*(t), \boldsymbol{\phi})$ is a fully connected network with one hidden layer, which does not take as input all the dimensions of $\boldsymbol{z}^*(t)$ but merely 20 of them. The width of the hidden layer is also 20, as it is the output. We fix 10 of these 20 dimensions and feed them to a linear hypernet with weight matrix shape $[10, p]$ to output the p weights for the main component. It should be emphasized that architecture of FFJORD in the main dimensions remains unchanged in AFFJORD for fair comparison, and we only fine-tuned the augmented structure in addition. Additional details about the experimental settings can be found in Appendix I.

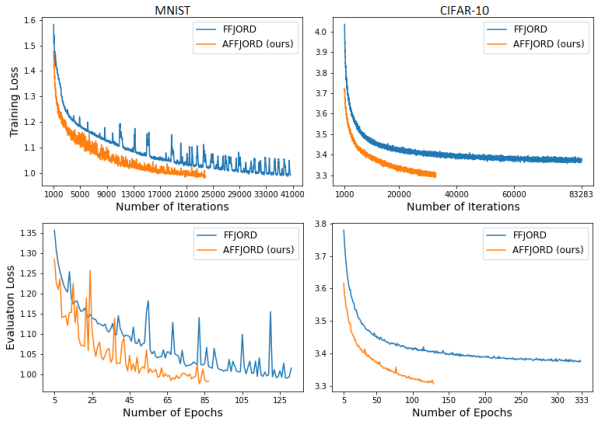

Figure 9: Graphs of training and evaluation losses of FFJORD and AFFJORD on MNIST as well as CIFAR-10. We run the experiments 3 times, and the graphs above represent the best individual results for each model. Lower is better.

As we show in Table 1, AFFJORD slightly outperforms FFJORD on MNIST, as both models reach optimal performance, as seen from the generated samples in Figure 10.

Figure 10: Samples generated from AFFJORD: MNIST and CIFAR-10 .

However, our model outperforms FFJORD on the CIFAR-10 dataset, as illustrated in Table 1, as well as in Figure 9.

In the case of MNIST, both models were trained for roughly 5 days, while in the case of CIFAR-10, they were trained for approximately 12 days. The results corresponding to the Real NVP and Glow models, are taken from the original papers: (Dinh et al., 2016) and (Kingma & Dhariwal, 2018).

As in the previous case, AFFJORD can generate samples by backintegrating. However, due to the use of the augmented multiscale architecture, where for each cycle we replace the augmented dimensions, these replaced augmented dimensions must be saved in an array for the backward generative pass. Examples of samples from AFFJORD are shown in Figure 10 for both MNIST and CIFAR-10 datasets. Additional generated samples from both AFFJORD and FFJORD can be found in Appendix D.

5 LIMITATIONS AND FUTURE WORK

Number of Function Evaluations (NFE). As originally reported in (Grathwohl et al., 2018), the number of function evaluations is one of the main bottlenecks of Neural ODE-based models. Interestingly, if the concatenation architecture is used in AFFJORD, that is, we only replace the concatenated time channel in FFJORD with the augmented channel in AFFJORD, then the number of function evaluations decreases significantly. This aspect is illustrated in Figure 11. However, the hypernet architecture of AFFJORD is prone to the issue of the number of function evaluations. Indeed, forward passes in AFFJORD-hypernet can be challenging, as the learnt vector field becomes too stiff such that the number of function evaluations diverges during training.

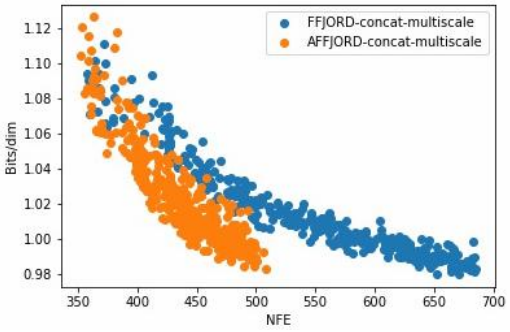

Figure 11: Number of function evaluations of FFJORD and AFFJORD when the 'concatenate' architecture is used.

Addition of Self-Attention (Ho et al., 2019), describe three modeling inefficiencies in prior work on flow models. One such factor is the lack of expressive power of convolutional layers used in normalizing flow models. Considering the performance improvements demonstrated in (Ho et al., 2019), in the future we intend to test the improvement in performance brought by the addition of self-attention in AFFJORD.

Data Dependent Augmented Dimensions As described in Section 3.1, several simplifications are made in the architecture of the general augmented neural ODE flows, in order to ensure immediate bijectivity and reduce the computational complexity. However, other possible architectures exist, where for example $\boldsymbol{z}^*(0)$ is dependent on $\boldsymbol{z}(0)$, and $g(\boldsymbol{z}^*(t)) = -\boldsymbol{z}^*(t)$. This would ensure that the augmented dimensions converge to zero, providing bijectivity. Since all augmented dimensions would be different during training, this would imply that the data is lifted to a higher plane, enabling richer transformations. Finding an approximation of the loss in Equation 15 remains a challenge for the future however.

Jacobian Regularization

Theoretically speaking, all performance enhancing modifications that can be applied to FFJORD are also applicable to AFFJORD. Such a modification which reduces the training time of FFJORD is presented in (Finlay et al., 2020), where both the vector field and its Jacobian are regularized. Thus, an interesting research direction in the future would be to test how the performance of AFFJORD is affected by such amendments.

6 CONCLUSION

We have presented the generalization of the total derivative decomposition in the continuous sense as well as the continuous generalization of the chain rule, to which we refer as the cable rule. The cable rule is analogous to the forward sensitivity of ODEs in the sense that, it gives the dynamics of the Jacobian of the state with respect to the initial conditions, whereas forward sensitivity gives the dynamics of the Jacobian of the state with respect to the parameters of the flow. Motivated by this contribution, we propose a new type of continuous normalizing flow, namely Augmented FFJORD (AFFJORD), which outperforms the CNF state-of-art-approach, FFJORD, in the experiments we conducted on the task of density estimation on both 2D toy data, and on high dimensional datasets such as MNIST and CIFAR-10.

Bibliography

Adams, R. P., Pennington, J., Johnson, M. J., Smith, J., Ovadia, Y., Patton, B., & Saunderson, J. (2018). Estimating the spectral density of large implicit matrices.
URL https://arxiv.org/abs/1802.03451

Blanes, S., Casas, F., Oteo, J., & Ros, J. (2009). The magnus expansion and some of its applications. *Physics Reports*, *470*(5-6), 151–238.
URL https://doi.org/10.1016%2Fj.physrep.2008.11.001

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. URL https://arxiv.org/abs/1806.07366

Dinh, L., Krueger, D., & Bengio, Y. (2014). Nice: Nonlinear independent components estimation. URL https://arxiv.org/abs/1410.8516

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2016). Density estimation using real nvp. URL https://arxiv.org/abs/1605.08803

Dupont, E., Doucet, A., & Teh, Y. W. (2019). Augmented neural odes. URL https://arxiv.org/abs/1904.01681

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., & Oberman, A. (2020). How to train your neural ODE: the world of Jacobian and kinetic regularization. In H. D. III, & A. Singh (Eds.) *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, (pp. 3154–3164). PMLR. URL https://proceedings.mlr.press/v119/finlay20a.html

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., & Duvenaud, D. (2018). Ffjord: Free-form continuous dynamics for scalable reversible generative models. URL https://arxiv.org/abs/1810.01367

Ho, J., Chen, X., Srinivas, A., Duan, Y., & Abbeel, P. (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. URL https://arxiv.org/abs/1902.00275

Hutchinson, M. (1990). A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*, *19*(2), 433–450. URL https://doi.org/10.1080/03610919008812866

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. URL https://arxiv.org/abs/1807.03039

Kobyzev, I., Prince, S. J., & Brubaker, M. A. (2021). Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(11), 3964–3979. URL https://doi.org/10.1109%2Ftpami.2020.2992934

Magnus, W. (1954). On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*, *7*(4), 649–673. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160070404

Massaroli, S., Poli, M., Park, J., Yamashita, A., & Asama, H. (2020). Dissecting neural odes. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.) *Advances in Neural Information Processing Systems*, vol. 33, (pp. 3952–3963). Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2020/file/293835c2cc75b585649498ee74b395f5-Paper.pdf

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, *22*(57), 1–64. URL http://jmlr.org/papers/v22/19-1028.html

Pontrjagin, L., Boltyanskii, V., Gamkrelidze, R., Mishchenko, E., & Brown, D. (1962). *The Mathematical Theory of Optimal Processes*. International series of monographs in pure and applied mathematics. Wiley. URL https://books.google.fr/books?id=PcH9oAEACAAJ

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. URL https://arxiv.org/abs/1505.05770

Tabak, E. G., & Turner, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, *66*(2), 145–164. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.21423

Tabak, E. G., & Vanden-Eijnden, E. (2010). Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, *8*(1), 217 – 233. URL https://doi.org/

Zhang, T., Yao, Z., Gholami, A., Gonzalez, J. E., Keutzer, K., Mahoney, M. W., & Biros, G. (2019). Anodev2: A coupled neural ode framework. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.) *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc. URL https://proceedings.neurips.cc/paper/2019/file/227f6afd3b7f89b96c4bb91f95d50f6d-Paper.pdf

Zhao, H., & Mousseau, V. A. (2013). Extended forward sensitivity analysis for uncertainty quantification. *Nuclear Technology*, *181*(1), 184–195. URL https://doi.org/10.13182/NT13-A15766

A CABLE RULE: THE CONTINUOUS GENERALISATION OF THE CHAIN RULE

We will first assume that z is one dimensional. If $z_i = f_i(z_{i-1})$ for $i \in \{1, ..., n\}$ then by the chain rule we have

$$\frac{dz_n}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}} \frac{dz_{n-1}}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} ... \frac{\partial z_1}{\partial z_0} = \frac{\partial f_n(z_{n-1})}{\partial z_{n-1}} \frac{\partial f_{n-1}(z_{n-2})}{\partial z_{n-2}} ... \frac{\partial f_1(z_0)}{\partial z_0} \tag{17}$$

Now, if we assume that $z_i = z(t_i)$ is transformed more gradually as in $z_{i+1} = z_i + \epsilon f(z_i)$, and that $t_{i+1} = t_i + \epsilon$, we get that

$$\frac{dz_n}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}} \frac{\partial z_{n-1}}{\partial z_{n-2}} ... \frac{\partial z_1}{\partial z_0} = (I + \epsilon \frac{\partial f(z_{n-1})}{\partial z_{n-1}})(I + \epsilon \frac{\partial f(z_{n-2})}{\partial z_{n-2}})...(I + \epsilon \frac{\partial f(z_0)}{\partial z_0}) \tag{18}$$

We see that $z(t) = z(0) + \int_0^t f(z(\tau))d\tau$ is the limit of the previous iterative definition $z_{i+1} = z_i + \epsilon f(z_i, \boldsymbol{\theta}(t_i))$, when $\epsilon \to 0$. For simplicity, we have written $f(z_i) = f(z_i, \boldsymbol{\theta}, t_i)$. If we decide to expand equation 18, we obtain

$$\frac{dz_n}{dz_0} = I + \sum_{i=0}^{n-1} \frac{\partial f(z_i)}{\partial z_i} \epsilon + \sum_{i=0}^{n-1} \sum_{j<i} \frac{\partial f(z_i)}{\partial z_i} \frac{\partial f(z_j)}{\partial z_j} \epsilon^2 + ... + \frac{\partial f(z_0)}{\partial z_0} ... \frac{\partial f(z_n)}{\partial z_n} \epsilon^n \tag{19}$$

$$= I + S_1^n + S_2^n + ... + S_{n+1}^n, \tag{20}$$

where

$$S_2^n = \sum_{i=0}^{n-1} \sum_{j<i} \frac{\partial f(z_i)}{\partial z_i} \frac{\partial f(z_j)}{\partial z_j} \epsilon^2, \; S_3^n = \sum_{i=0}^{n-1} \sum_{j<i} \sum_{k<j} \frac{\partial f(z_i)}{\partial z_i} \frac{\partial f(z_j)}{\partial z_j} \frac{\partial f(z_k)}{\partial z_k} \epsilon^3, ... \tag{21}$$

We will focus on the sum S_2^n for a moment. Let us define

$$g_2^n(x, y) = \frac{\partial f(z_{t_i})}{\partial z_{t_i}} \frac{\partial f(z_{t_j})}{\partial z_{t_j}}, \text{ for } x \in [t_i, t_{i+1}], y \in [t_j, t_{j+1}]. \tag{22}$$

It is clear that g_2^n is the discretization of

$$g_2(t, u) = \frac{\partial f(z_t)}{\partial z_t} \frac{\partial f(z_u)}{\partial z_u}, \text{ for } t \in [0, T], u \in [0, T]. \tag{23}$$

We can notice that $S_2^n = \sum_{i=0}^{n-1} \sum_{j<i} g_2^n(t_i, t_j)\epsilon^2$ and that $g_2^n(t_i, t_j) = g_2^n(t_j, t_i)$ is symmetric, but in S_2^n, only takes values in the rectangles under the diagonal as illustrated in Figure 14. If we decide to expand the sum S_2^n on the rectangles above the diagonal as in Figure 15 and denote it as $\bar{S}_2^n = \sum_{i=0}^{n-1} \sum_{j \neq i} g_2^n(t_i, t_j)\epsilon^2$, then due to the symmetry of $g_2^n(t_i, t_j) = g_2^n(t_j, t_i)$, we deduce that $S_2^n = \frac{1}{2}\bar{S}_2^n$. The only rectangles missing in $\bar{S}_2^n$, regarding the discretization of $[0, T] \times [0, T]$, are the ones corresponding to the cases when $i = j$, which are the rectangles in the diagonal. It is important to emphasize here that S_2^n is the sum of C_n^2 terms, while $\bar{S}_2^n$ is made out of $V_n^2 = 2!C_n^2$ terms. This is especially apparent when we notice the discretization of $[0, T] \times [0, T]$ is composed of n^2 rectangles, while the diagonal is composed of n rectangles, hence $\bar{S}_2^n$ is the sum of $n^2 - n = V_n^2$ terms. The ratio of the collective mass of the rectangles in the diagonal with respect to the entire $[0, T] \times [0, T]$ goes to zero as $n \to \infty$, hence we conclude that

$$\bar{S}_2^n = \sum_{i=0}^{n-1} \sum_{j \neq i} \frac{\partial f(z_i)}{\partial z_i} \frac{\partial f(z_j)}{\partial z_j} \epsilon^2 \to \int_{[0,T]\times[0,T]} g_2(t, u) d(t, u) = (\int_0^T \frac{\partial f(z_t)}{\partial z_t} dt)^2, \tag{24}$$

thus,

$$S_2^n = \frac{1}{2!} \bar{S}_2^n \to \frac{1}{2!} (\int_0^T \frac{\partial f(z_t)}{\partial z_t} dt)^2. \tag{25}$$

Similarly for

$$S_3^n = \sum_{i=0}^{n-1} \sum_{j<i} \sum_{k<j} \frac{\partial f(z_i)}{\partial z_i} \frac{\partial f(z_j)}{\partial z_j} \frac{\partial f(z_k)}{\partial z_k} \epsilon^3,$$

we can define

$$g_3^n(x, y, z) = \frac{\partial f(z_{t_i})}{\partial z_{t_i}} \frac{\partial f(z_{t_j})}{\partial z_{t_j}} \frac{\partial f(z_{t_k})}{\partial z_{t_k}}, \text{ for } x \in [t_i, t_{i+1}], y \in [t_j, t_{j+1}], z \in [t_k, t_{k+1}], \tag{26}$$

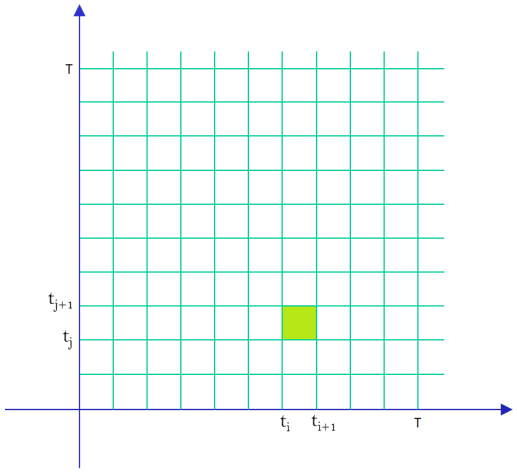

Figure 12: Rectangle (i, j) in discretized $[0, T] \times [0, T]$

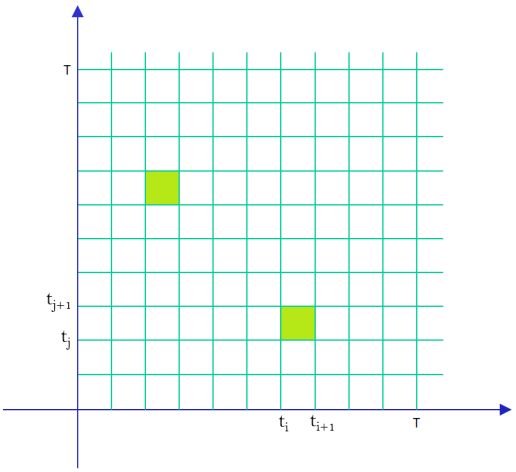

Figure 13: Symmetry of g_2^n

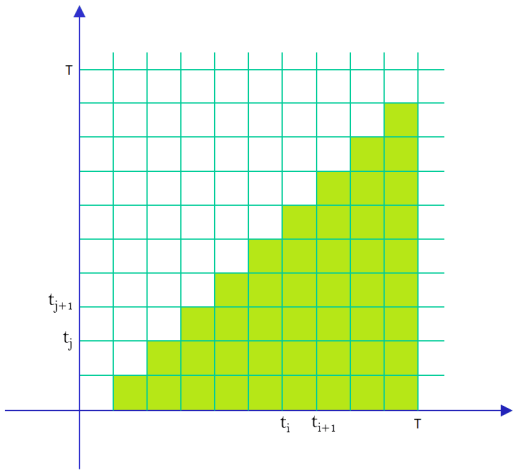

Figure 14: Rectangles participating in S_2^n

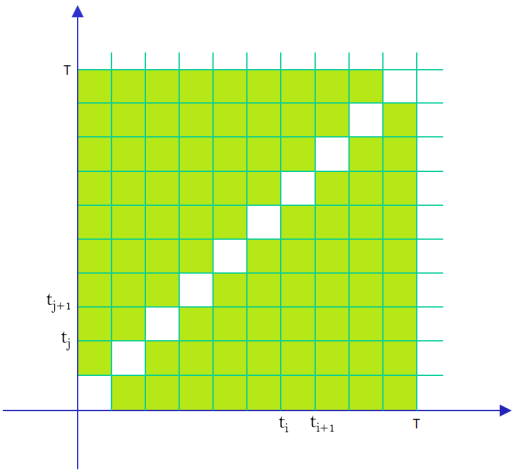

Figure 15: Rectangles participating in $\bar{S}_2^n$

and

$$g_3(t, u, v) = \frac{\partial f(z_t)}{\partial z_t} \frac{\partial f(z_u)}{\partial z_u} \frac{\partial f(z_v)}{\partial z_v}, \text{ for } t \in [0, T], u \in [0, T], v \in [0, T]. \tag{27}$$

In this case:

$$g_3^n(x, y, z) = g_3^n(x, y, z) = g_3^n(x, z, y) = g_3^n(y, x, z) = g_3^n(y, z, x) = g_3^n(z, x, y) = g_3^n(z, y, x), \tag{28}$$

where each equality corresponds to one of the $6 = 3!$ permutations of (x, y, z). As before we can expand the domain of S_3^n, by adding the rectangles in the discretization of $[0, T] \times [0, T] \times [0, T]$ such that i might be smaller than j, as well as that j could be smaller than k. We denote this expanded sum as $\bar{S}_3^n$, and by the symmetry of g_3^n, we notice that $S_3^n = \frac{1}{3!} \bar{S}_3^n$. This implies that the rectangles participating in $\bar{S}_3^n$, now cover most of $[0, T] \times [0, T] \times [0, T]$, where the only exceptions are the ones when $i = j$ or $j = k$ (or both). The number of rectangles participating in $\bar{S}_3^n$ is $V_n^3 = 3! C_n^3 = n^3 - 3n^2 + 2n$, and since the number of all rectangles in $[0, T] \times [0, T] \times [0, T]$ is n^3, this implies that $3n^2 - 2n$ rectangles are missing in sum $\bar{S}_3^n$ from the cases when $i = j$ or $j = k$ (or both). The ratio of the collective mass of such rectangles with respect to the entire $[0, T] \times [0, T] \times [0, T]$ goes to zero as $n \to \infty$, hence as before:

$$\bar{S}_3^n = \sum_{i=0}^{n-1} \sum_{j \neq i} \sum_{i \neq k \neq j} \frac{\partial f(z_i)}{\partial z_i} \frac{\partial f(z_j)}{\partial z_j} \frac{\partial f(z_k)}{\partial z_k} \epsilon^3 \to \int_{[0,T] \times [0,T] \times [0,T]} g_3(t, u, v) d(t, u, v) \tag{29}$$

$$= (\int_0^T \frac{\partial f(z_t)}{\partial z_t} dt)^3, \tag{30}$$

implying

$$S_3^n = \frac{1}{3!}\bar{S}_3^n \to \frac{1}{3!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^3. \tag{31}$$

In a similar fashion we can prove that $S_k^n \to \frac{1}{k!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^k$. Thus we conclude that:

$$\frac{dz(T)}{dz(0)} = \frac{1}{0!}I + \frac{1}{1!}\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt + \frac{1}{2!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^2 + \frac{1}{3!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^3 + ... \tag{32}$$

$$\frac{dz(T)}{dz(0)} = e^{\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt}. \tag{33}$$

Unfortunately, this result does not hold when the dimensionality of $\boldsymbol{z}(t)$ is larger than one. Indeed, in this case, $\frac{\partial f(\boldsymbol{z}_{t_i})}{\partial \boldsymbol{z}_{t_i}}$ is a matrix, hence $g_2^n(x,y) = \frac{\partial f(\boldsymbol{z}_{t_i})}{\partial \boldsymbol{z}_{t_i}}\frac{\partial f(\boldsymbol{z}_{t_j})}{\partial \boldsymbol{z}_{t_j}}$ is not necessarily symmetric, as the commutator $[\frac{\partial f(\boldsymbol{z}_{t_i})}{\partial \boldsymbol{z}_{t_i}}, \frac{\partial f(\boldsymbol{z}_{t_j})}{\partial \boldsymbol{z}_{t_j}}]$ is not necessarily zero. For this reason, inspired by the previous result we try a different approach. Indeed, we can see from Equation 33 that $\frac{dz(T)}{dz(0)}$ is the solution of the following ODE:

$$\frac{d(\frac{dz(t)}{dz(0)})}{dt} = \frac{\partial f(z(t))}{\partial z(t)}\frac{dz(t)}{dz(0)} \tag{34}$$

as the initial condition $\frac{dz(t=0)}{dz(0)} = I$. Hence we wish to prove that $\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$ satisfies the same ODE in higher dimensions as well. We notice that we can write:

$$\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau))d\tau = g(\boldsymbol{z}(0), t), \tag{35}$$

therefore,

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial^2 g(\boldsymbol{z}(0),t)}{\partial t \partial \boldsymbol{z}(0)} = \frac{\partial^2 g(\boldsymbol{z}(0),t)}{\partial \boldsymbol{z}(0)\partial t} = \frac{d(\frac{dg(\boldsymbol{z}(0),t)}{dt})}{d\boldsymbol{z}(0)} = \frac{df(\boldsymbol{z}(t))}{d\boldsymbol{z}(0)} = \frac{df(\boldsymbol{z}(t))}{d\boldsymbol{z}(t)}\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{36}$$

We pause for a moment, in order to highlight the similarity of expression 36 and the forward sensitivity:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{\theta}})}{dt} = \frac{df(\boldsymbol{z}(t),t,\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \frac{\partial f(\boldsymbol{z}(t),t,\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}\frac{d\boldsymbol{z}(t)}{d\boldsymbol{\theta}} + \frac{\partial f(\boldsymbol{z}(t),t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{37}$$

Now from Expression 36, we infer that $\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$ is the solution of the following linear ODE:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)}\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{38}$$

If we write $\boldsymbol{Y}(t) = \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$, then the equation above becomes:

$$\frac{d\boldsymbol{Y}(t)}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)}\boldsymbol{Y}(t). \tag{39}$$

The general solution of first-order homogeneous linear ODEs is given in (Magnus, 1954; Blanes et al., 2009), and in our case can be written as follows:

$$\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(t)}\frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(t)}, \text{ for } \boldsymbol{\Omega}(t) = \sum_{k=1}^{\infty}\boldsymbol{\Omega}_k(t), \tag{40}$$

where

$$\boldsymbol{\Omega}_1(t) = \int_0^t \boldsymbol{A}(t_1)\,dt_1, \tag{41}$$

$$\boldsymbol{\Omega}_2(t) = \frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2\,[\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)], \tag{42}$$

$$\boldsymbol{\Omega}_3(t) = \frac{1}{6}\int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3 \Big(\big[\boldsymbol{A}(t_1), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_3)]\big] + \big[\boldsymbol{A}(t_3), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_1)]\big]\Big), \tag{43}$$

and so on for $k > 3$, where:

$$\boldsymbol{A}(t) = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{44}$$

We notice that if $z(t)$ is one dimensional, then this result agrees with the one in the previous approach. To conclude, we have proven the following theorem:

Theorem A.1. *Let $\frac{d\boldsymbol{z}(t)}{dt} = f(\boldsymbol{z}(t), \boldsymbol{\theta}, t)$, where f is continuous in t and Lipschitz continuous in $\boldsymbol{z}(t)$. Then the following holds:*

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)} = e^{\int_0^T \frac{\partial f(\boldsymbol{z}_t)}{\partial \boldsymbol{z}_t} dt + \boldsymbol{\Omega}_2(T) + \ldots.}, \tag{45}$$

where $\boldsymbol{\Omega}_{k>1}(t)$ are the terms of the Magnus series.

Proof. Since f is continuous in t and Lipschitz continuous in $\boldsymbol{z}$ then due to Picard–Lindelöf theorem, $\boldsymbol{z}(T)$ exists and is unique. Furthermore, since f is Lipschitz continuous in $\boldsymbol{z}(t)$, then $\frac{\partial f(\boldsymbol{z}_t)}{\partial \boldsymbol{z}_t}$ exists almost everywhere. Based on the previous derivations we reach the desired conclusion. □

B THE CONTINUOUS GENERALIZATION OF THE TOTAL DERIVATIVE DECOMPOSITION

In Appendix A, we assumed that $\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau), \boldsymbol{\theta}, \tau) d\tau$, which was the the limit of $\epsilon \to 0$ of the previous iterative definition of $\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + \epsilon f(\boldsymbol{z}_i, \boldsymbol{\theta}, t_i)$. Now, we assume that the set of parameters in f is different at each discrete time, that is $\boldsymbol{z}_{i+1} = \boldsymbol{z}_i + \epsilon f_i(\boldsymbol{z}_i, \boldsymbol{\theta}_i, t_i)$, for independent sets $\boldsymbol{\theta}_i$. First, we notice that

$$\boldsymbol{z}_n = \boldsymbol{z}_n(\boldsymbol{z}(t_0), \boldsymbol{\theta}(t_0), \boldsymbol{\theta}(t_1), ..., \boldsymbol{\theta}(t_{n-1})) = \boldsymbol{z}_n(\boldsymbol{z}(t_1), \boldsymbol{\theta}(t_1), \boldsymbol{\theta}(t_2), ..., \boldsymbol{\theta}(t_{n-1})) =$$

$$...\boldsymbol{z}_n(\boldsymbol{z}(t_{k+1}), \boldsymbol{\theta}(t_{k+1}), \boldsymbol{\theta}(t_{k+2}), ..., \boldsymbol{\theta}(t_{n-1})) = ... = \boldsymbol{z}_n(\boldsymbol{z}(t_{n-1}), \boldsymbol{\theta}(t_{n-1})) = \boldsymbol{z}(t_n = T - \epsilon).$$

This can be seen from Figure 16.

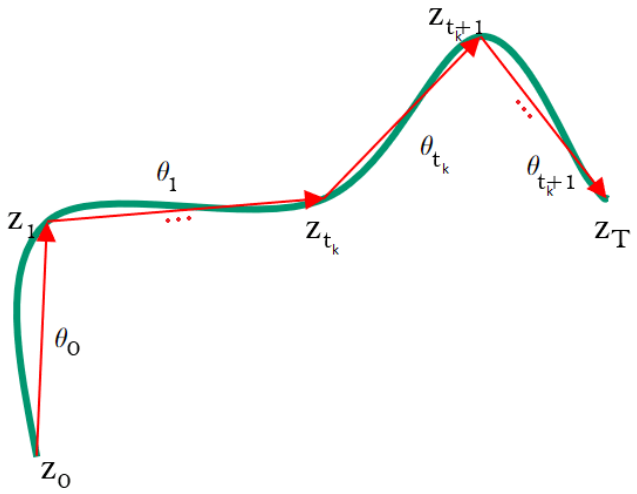

Figure 16: The dependencies in the discretisation of $\frac{\boldsymbol{z}(t)}{t} = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$. Variable $\boldsymbol{z}_{t_{k+1}}$ completely absorbs the contribution of $\boldsymbol{\theta}_{t_k}$ to $\boldsymbol{z}_T$, hence $\boldsymbol{z}_T$ is a function of $\boldsymbol{z}_{t_{k+1}}$ and the following sets of parameters $\boldsymbol{\theta}_{t_{k+1}}, \boldsymbol{\theta}_{t_{k+2}}, ..., \boldsymbol{\theta}_{t_n}$.

Thus

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \frac{d\boldsymbol{z}_n(\boldsymbol{z}(t_k + \epsilon), \boldsymbol{\theta}(t_k + \epsilon), \boldsymbol{\theta}(t_{k+2}), ..., \boldsymbol{\theta}(t_n))}{d\boldsymbol{\theta}(t_k)} \tag{46}$$

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k + \epsilon)} \frac{d\boldsymbol{z}(t_k + \epsilon)}{d\boldsymbol{\theta}(t_k)} + 0 + 0 + ... + 0 = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k + \epsilon)} \frac{\partial \boldsymbol{z}(t_k + \epsilon)}{\partial \boldsymbol{\theta}(t_k)}. \tag{47}$$

We now focus on analysing $\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)}$ and $\frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)}$. First we notice that from:

$$\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)}\frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{z}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)}(I + \epsilon\frac{\partial f(\boldsymbol{z}(t_k),\boldsymbol{\theta}(t_k))}{\partial \boldsymbol{z}(t_k)} + O(\epsilon^2)) \tag{48}$$

we get

$$\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} - \epsilon\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)}\frac{\partial f(\boldsymbol{z}(t_k),\boldsymbol{\theta}(t_k))}{\partial \boldsymbol{z}(t_k)} + O(\epsilon^2). \tag{49}$$

Regarding $\frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)}$ the following holds:

$$\frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)} = \frac{\partial(\boldsymbol{z}(t_k) + f(\boldsymbol{z}(t_k),\boldsymbol{\theta}(t_k))\epsilon + O(\epsilon^2))}{\partial \boldsymbol{\theta}(t_k)} = 0 + \frac{\partial f(\boldsymbol{z}(t_k),\boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)}\epsilon + O(\epsilon^2). \tag{50}$$

After combining both Equation 49 and Equation 50 in expression 47, we deduce that:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \big(\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} - \epsilon\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)}\frac{\partial f(\boldsymbol{z}(t_k),\boldsymbol{\theta}(t_k))}{\partial \boldsymbol{z}(t_k)} + O(\epsilon^2)\big)\big(\frac{\partial f(\boldsymbol{z}(t_k),\boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)}\epsilon + O(\epsilon^2)\big) \tag{51}$$

$$\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{\theta}(t_k)} = \frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)}\frac{\partial f(\boldsymbol{z}(t_k),\boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)}\epsilon + O(\epsilon^2) \tag{52}$$

Now assuming that $\boldsymbol{\theta}(t_k) = g(t_k,\boldsymbol{\theta})$, we can write the total derivative of $\boldsymbol{z}_n$ with respect to parameters $\boldsymbol{\theta}$:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}} = \sum_{k=0}^{n}\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{\theta}(t_k)}\frac{d\boldsymbol{\theta}(t_k)}{d\boldsymbol{\theta}} = \sum_{k=0}^{n}\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)}\frac{\partial f(\boldsymbol{z}(t_k),\boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)}\frac{d\boldsymbol{\theta}(t_k)}{d\boldsymbol{\theta}}\epsilon + O(\epsilon^2) \tag{53}$$

hence taking $\epsilon \to 0$, we conclude that

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)}\frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}}dt. \tag{54}$$

We can see that we are integrating the infinitesimal contributions of parameters $\boldsymbol{\theta}$, at each time t. In case that $\boldsymbol{\theta}(t) = g(t,\boldsymbol{\theta}) = \boldsymbol{\theta}$, then we have

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}Idt = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}dt = -\int_T^0 \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}dt. \tag{55}$$

Theorem B.1. *Let $\frac{dz(t)}{dt} = f(\boldsymbol{z}(t),\boldsymbol{\theta}(t),t)$, where f is continuous in t and Lipschitz continuous in $\boldsymbol{z}(t)$ and $\boldsymbol{\theta}(t)$. Then the following holds:*

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{\theta}(t),t)}{\partial \boldsymbol{\theta}(t)}\frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}}dt. \tag{56}$$

Proof. As before, since f is continuous in t and Lipschitz continuous in z then due to Picard–Lindelöf theorem, $z(T)$ exists and is unique. From Theorem A.1, we can establish the existence of $\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(t)}$. Furthermore, since f is Lipschitz continuous in $\theta(t)$, then $\frac{\partial f(\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)}$ exists almost everywhere. Based on the previous derivations we reach the desired conclusion. □

C GENERALIZATION OF CONTINUOUS BACKPROPAGATION INTO PIECEWISE CONTINUOUS BACKPROPAGATION

We define $\boldsymbol{z}(t)$ as before, with the only difference being that its derivative is discontinuous at a point (say $\frac{T}{2}$):

$$\boldsymbol{z}(t) = \begin{cases} \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau),\boldsymbol{\theta}(\tau))d\tau,\ t \in [0,\frac{T}{2}] \\ \boldsymbol{z}(0) + \int_0^{\frac{T}{2}} f(\boldsymbol{z}(\tau),\boldsymbol{\theta}(\tau))d\tau + \int_{\frac{T}{2}}^t g(\boldsymbol{z}(\tau),\boldsymbol{\phi}(\tau))d\tau,\ t \in (\frac{T}{2},T] \end{cases} \tag{57}$$

As in (Chen et al., 2018), we can get

$$\frac{d(\frac{\partial L}{\partial \boldsymbol{z}(t)})}{dt} = \begin{cases} -\frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{z}(t)}, \ t \in [0, \frac{T}{2}] \\ -\frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi}(t))}{\partial \boldsymbol{z}(t)}, \ t \in (\frac{T}{2}, T], \end{cases} \tag{58}$$

thus

$$\frac{\partial L}{\partial \boldsymbol{z}(t)} = \begin{cases} \frac{\partial L}{\partial \boldsymbol{z}(T)} - \int_T^t \frac{\partial L}{\partial \boldsymbol{z}(\tau)} \frac{\partial g(\boldsymbol{z}(\tau), \boldsymbol{\phi}(\tau))}{\partial \boldsymbol{z}(\tau)} d\tau, \ t \in (\frac{T}{2}, T] \\ \frac{\partial L}{\partial \boldsymbol{z}(T)} - \int_{\frac{T}{2}}^t \frac{\partial L}{\partial \boldsymbol{z}(\tau)} \frac{\partial f(\boldsymbol{z}(\tau), \boldsymbol{\theta}(\tau))}{\partial \boldsymbol{z}(\tau)} d\tau - \int_T^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(\tau)} \frac{\partial g(\boldsymbol{z}(\tau) \boldsymbol{\phi}(\tau))}{\partial \boldsymbol{z}(\tau)} d\tau, \ t \in (0, \frac{T}{2}]. \end{cases} \tag{59}$$

The approach developed in Appendix B can be used to generalize continuous backpropagation into piecewise continuous backpropagation. Indeed, identically as before:

$$\frac{dL_\epsilon}{d\boldsymbol{\theta}} = \sum_{k=0}^{n} \frac{\partial L_\epsilon}{\partial \boldsymbol{z}(t_k)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \frac{\partial \boldsymbol{\theta}(t_k)}{\partial \boldsymbol{\theta}} \epsilon + \sum_{k=0}^{n} \frac{\partial L_\epsilon}{\partial \boldsymbol{z}(t_k)} \frac{\partial g(\boldsymbol{z}(t_k), \boldsymbol{\phi}(t_k))}{\partial \boldsymbol{\phi}(t_k)} \frac{\partial \boldsymbol{\phi}(t_k)}{\partial \boldsymbol{\theta}} \epsilon + O(\epsilon^2) \tag{60}$$

Taking the limit we get:

$$\frac{dL}{d\boldsymbol{\theta}} = \int_0^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}} dt + \int_{\frac{T}{2}}^{T} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\phi}(t)}{\partial \boldsymbol{\theta}} dt \tag{61}$$

and in the same manner we get:

$$\frac{dL}{d\boldsymbol{\phi}} = \int_0^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\phi}} dt + \int_{\frac{T}{2}}^{T} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\phi}(t)}{\partial \boldsymbol{\phi}} dt, \tag{62}$$

If we set $\boldsymbol{\theta}(t) = \boldsymbol{\theta}$ and $\boldsymbol{\phi}(t) = \boldsymbol{\phi}$, then we get:

$$\frac{dL}{d\boldsymbol{\theta}} = \int_0^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt, \tag{63}$$

and

$$\frac{dL}{d\boldsymbol{\phi}} = \int_{\frac{T}{2}}^{T} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} dt. \tag{64}$$

D ADDITIONAL GENERATED SAMPLES

Below in Figure 17 and Figure 18, are presented additional examples of generated samples of MNIST and CIFAR-10 by AFFJORD, respectively.

Figure 17: Generated samples of MNIST by AFFJORD

Figure 18: Generated samples of CIFAR-10 by AFFJORD

In addition in Figure 19 and Figure 20, are presented additional examples of generated samples of MNIST and CIFAR-10 by FFJORD, respectively.

Figure 19: Generated samples of MNIST by FFJORD

Figure 20: Generated samples of CIFAR-10 by FFJORD

E DERIVING THE INSTANTANEOUS CHANGE OF VARIABLE VIA THE CABLE RULE

The trace of a commutator $[\boldsymbol{X}, \boldsymbol{Y}] = \boldsymbol{XY} - \boldsymbol{YX}$ is always zero, hence we prove below that for all terms $\boldsymbol{\Omega}_{k>1}(t)$ given in Equations 41 in Appendix A, we have $tr(\boldsymbol{\Omega}_k(t)) = 0$.

Indeed, since the trace and the integral are interchangeable we have:

$$tr(\boldsymbol{\Omega}_2(t)) = tr\frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2 \, [\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)] = \frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2 \, tr[\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)] = 0. \tag{65}$$

$$tr(\boldsymbol{\Omega}_3(t)) = \frac{1}{6}\int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3 \left(tr\big[\boldsymbol{A}(t_1), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_3)]\big] + tr\big[\boldsymbol{A}(t_3), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_1)]\big] \right), \tag{66}$$

$$= \frac{1}{6}\int_0^t dt_1 \int_0^{t_1} (0+0) dt_2 = 0, \tag{67}$$

and so on for $k > 3$.

The only exception is the case when $k = 1$:

$$tr(\boldsymbol{\Omega}_1(t)) = tr\int_0^t \boldsymbol{A}(t_1) dt_1 = \int_0^t tr\frac{\partial f(\boldsymbol{z}(t_1), t_1, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} dt_1. \tag{68}$$

Finally, using Jacobi's formula, we conclude that:

$$\log|\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}| = \log|e^{\boldsymbol{\Omega}(T)}| = \log e^{tr(\boldsymbol{\Omega}(T))} = \int_0^T tr\frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt. \tag{69}$$

F SIMPLIFICATIONS IN SECTION 3.1

First we notice that if $z^*(0)$ does not depend on $z(0)$, then

$$|\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}| = |\begin{bmatrix} I, 0 \end{bmatrix} e^{\begin{bmatrix} \int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt & \int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)} dt \\ \int_0^T \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} dt & \int_0^T \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)} dt \end{bmatrix} + \boldsymbol{\Omega}_2(T) + \dots} \begin{bmatrix} I \\ \frac{d\boldsymbol{z}^*(0)}{d\boldsymbol{z}(0)} \end{bmatrix}|, \tag{70}$$

becomes

$$|\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}| = |\begin{bmatrix} I, 0 \end{bmatrix} e^{\begin{bmatrix} \int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt & \int_0^T \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)} dt \\ \int_0^T \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} dt & \int_0^T \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)} dt \end{bmatrix} + \boldsymbol{\Omega}_2(T) + \dots} \begin{bmatrix} I \\ 0 \end{bmatrix}|. \tag{71}$$

On the other hand, if the augmented dimensions do not depend on the main dimensions then $\frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} = 0$. This implies that

$$\boldsymbol{A}(t) = \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} & \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)} \\ \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} & \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)} \end{bmatrix} \rightarrow \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} & \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)} \end{bmatrix}. \tag{72}$$

Hence,

$$\boldsymbol{\Omega}_1(t) = \int_0^t dt_1 \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_1), \boldsymbol{z}^*(t_1), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \frac{\partial f(\boldsymbol{z}(t_1), \boldsymbol{z}^*(t_1), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_1)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_1), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_1)} \end{bmatrix} = \begin{bmatrix} \int_0^t dt_1 \frac{\partial f(\boldsymbol{z}(t_1), \boldsymbol{z}^*(t_1), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \boldsymbol{B}_1(t) \\ 0 & \boldsymbol{D}_1(t) \end{bmatrix} \tag{73}$$

$$\boldsymbol{\Omega}_2(t) =$$

$$\int_0^t dt_1 \int_0^{t_1} dt_2 \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_2), \boldsymbol{z}^*(t_2), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_2)} & \frac{\partial f(\boldsymbol{z}(t_2), \boldsymbol{z}^*(t_2), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_2)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_2), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_2)} \end{bmatrix} \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_1), \boldsymbol{z}^*(t_1), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \frac{\partial f(\boldsymbol{z}(t_1), \boldsymbol{z}^*(t_1), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_1)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_1), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_1)} \end{bmatrix}$$

$$- \int_0^t dt_1 \int_0^{t_1} dt_2 \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_1), \boldsymbol{z}^*(t_1), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \frac{\partial f(\boldsymbol{z}(t_1), \boldsymbol{z}^*(t_1), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_1)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_1), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_1)} \end{bmatrix} \begin{bmatrix} \frac{\partial f(\boldsymbol{z}(t_2), \boldsymbol{z}^*(t_2), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_2)} & \frac{\partial f(\boldsymbol{z}(t_2), \boldsymbol{z}^*(t_2), \boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_2)} \\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_2), \boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_2)} \end{bmatrix}$$

$$= \begin{bmatrix} \int_0^t dt_1 \int_0^{t_1} dt_2 [\frac{\partial f(\boldsymbol{z}(t_1), \boldsymbol{z}^*(t_1), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)}, \frac{\partial f(\boldsymbol{z}(t_2), \boldsymbol{z}^*(t_2), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t_2)}] & \boldsymbol{B}_2(t) \\ 0 & \boldsymbol{D}_2(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Omega}_2^{[z]}(t) & \boldsymbol{B}_2(t) \\ 0 & \boldsymbol{D}_2(t) \end{bmatrix}.$$

In this way we can prove that

$$\boldsymbol{\Omega}(t) = \sum_{k=1}^{\infty} \boldsymbol{\Omega}(t)_k = \begin{bmatrix} \sum_{k=1}^{\infty} \boldsymbol{\Omega}_k^{[z]}(t) & \sum_{k=1}^{\infty} \boldsymbol{B}_k(t) \\ 0 & \sum_{k=1}^{\infty} \boldsymbol{D}_k(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Omega}^{[z]}(t) & \boldsymbol{B}(t) \\ 0 & \boldsymbol{D}(t) \end{bmatrix}.$$

Considering that the exponential of a matrix whose lower left block is zero, will still have a zero lower left block, we have:

$$\frac{dz(t)}{dz(0)} = e^{\boldsymbol{\Omega}(t)} = \begin{bmatrix} e^{\boldsymbol{\Omega}^{[z]}(t)} & \bar{\boldsymbol{B}}(t) \\ 0 & \bar{\boldsymbol{D}}(t) \end{bmatrix}.$$

Finally,

$$|\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}| = |\begin{bmatrix} I, 0 \end{bmatrix} \begin{bmatrix} e^{\boldsymbol{\Omega}^{[z]}(t)} & \bar{\boldsymbol{B}}(t) \\ 0 & \bar{\boldsymbol{D}}(t) \end{bmatrix} \begin{bmatrix} I \\ 0 \end{bmatrix}| = |e^{\boldsymbol{\Omega}^{[z]}(t)}| = e^{\int_0^T tr \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt + 0 + \ldots 0 + \ldots}. \tag{74}$$

G CABLE RULE DERIVED VIA EQUATION 5

Differentiating Equation 5 in Appendix A, we get:

$$\frac{d\big(\frac{dL}{d\boldsymbol{z}(t)}\big)}{dt} = -\frac{dL}{d\boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{75}$$

Choosing L to be $\boldsymbol{z}(0)$, we have

$$\frac{d\big(\frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(t)}\big)}{dt} = -\frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{76}$$

We define $\boldsymbol{A}(t) := \frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(t)}$, and $\boldsymbol{B}(t) := \boldsymbol{A}(t)^{-1} = \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$, so that the equation above becomes

$$\frac{d\boldsymbol{A}(t)}{dt} = -\boldsymbol{A}(t) \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}, \tag{77}$$

thus

$$-\boldsymbol{B}(t) \frac{d\boldsymbol{A}(t)}{dt} = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{78}$$

Then from

$$\boldsymbol{B}(t)\boldsymbol{A}(t) = I, \tag{79}$$

we have

$$-\boldsymbol{B}(t) \frac{d\boldsymbol{A}(t)}{dt} = \frac{d\boldsymbol{B}(t)}{dt} \boldsymbol{A}(t), \tag{80}$$

thus, using this expression in Equation 78 we get

$$\frac{d\boldsymbol{B}(t)}{dt} \boldsymbol{A}(t) = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{81}$$

Finally, we get the desired result by multiplying both sides from the right with $\boldsymbol{B}(t)$.

H EQUIVALENCE BETWEEN CONTINUOUS TOTAL DERIVATIVE DECOMPOSITION AND THE CONTINUOUS BACKPROPAGATION

In Section 3.3. we derived the expression of continuous backpropagation from the continuous total derivative decomposition. However, we can also derive the the formula of the continuous total derivative decomposition (Equation 12) from continuous backpropagation (Equation 13). Indeed, for a function $f = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$, where $\boldsymbol{\theta}(t) = g(\boldsymbol{\theta}, t)$, we can see f as $h(\boldsymbol{z}(t), \boldsymbol{\theta}, t) = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$. Hence,

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{dh(\boldsymbol{z}(t), \boldsymbol{\theta}, t)}{d\boldsymbol{\theta}} dt = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{df(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)}{d\boldsymbol{\theta}} dt, \tag{82}$$

therefore

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)}{\partial \boldsymbol{\theta}(t)} \frac{d\boldsymbol{\theta}(t)}{d\boldsymbol{\theta}} dt. \tag{83}$$

Setting $L = \boldsymbol{z}(T)$, gives the desired result.

I ADDITIONAL DETAILS ABOUT THE EXPERIMENTS

As mentioned in the main paper, we optimized the architecture of FFJORD first and fine-tuned its hyper-parameters.

Table 2: Number of Total Parameters of FFJORD and AFFJORD (Concat and Hypernet Architecture) for MNIST and CIFAR-10. Lower Is Better. The Multiscale Architecture Is Used in All Cases.

Dataset	Concat		Hypernet	
	FFJORD	AFFJORD	FFJORD	AFFJORD
MNIST	400323	417629	783019	4556169
CIFAR10	679441	820815	1332361	8976115

This architecture remains unchanged in the AFFJORD model for fair comparison, and we only fine-tuned the augmented structure and dimension in addition. Indeed, as it can be seen, the results we report for FFJORD (0.98 MNIST, 3.37 CIFAR) are better than those reported on the original paper (0.99 MNIST, 3.40 CIFAR). The aforementioned improvements were a result of reducing the number of parameters during our tuning process, by reducing the number of CNF blocks from 2 to 1. The total number of parameters for different architectures of FFJORD and AFFJORD can be found on Table 2. It should be emphasized that in the hypernet architecture, the parameters of AFFJORD are contained inside the parameters in the main architecture of FFJORD, so a bigger model is not being used, simply the flexibility of evolution of the main parameters in time is being increased.

More generally, our experiments show that augmented architectures generally improve the performance of the standard non-augmented FFJORD counterparts, independently from the baseline architecture used. Furthermore, the farther the performance of FFJORD is from being optimal, the greater are the improvements when using AFFJORD.