Enhanced Distribution Modelling via Augmented Architectures For Neural ODE Flows

Etrit Haxholli, Marco Lorenzi

Enhanced Distribution Modelling via Augmented Architectures For Neural ODE Flows

Etrit Haxholli[*]
Epione Research Group
Inria, Université Cote d'Azur
etrit.haxholli@inria.fr

Marco Lorenzi
Epione Research Group
Inria, Université Cote d'Azur
marco.lorenzi@inria.fr

Abstract

While the neural ODE formulation of normalizing flows such as in FFJORD 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 special augmented transformation dynamics which preserve the topology of the space. Furthermore, we derive the Jacobian determinant of the general augmented form by generalizing 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, CIFAR-10 and CelebA (32×32), 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., 2017; 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., 2019). 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., 2019), 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.

[*]The code is available at: https://github.com/ehaxholli/ANODEF

DLDE-III Workshop in the 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

In (Dupont et al., 2019), it is shown that there exist functions which neural ODEs are not capable of representing. To tackle this 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. Additional background material is provided in Appendix A.
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 (Lecun, 1998), CIFAR-10 (Krizhevsky et al., 2010) and CelebA (Liu et al., 2015), show that the proposed ANODE flow defined by AFFJORD outperforms FFJORD in terms of density estimation, thus highlighting the improved flexibility and representation properties of the underlying vector field.

2 Proposed Framework

In what follows, we first 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. Then, we derive the expression of the loss function for general augmented neural ODE flows, and finally introduce our model as a special case of such augmented neural ODEs.

The Cable Rule. If we define a chain of transformations

$$\boldsymbol{z}_i = g_i(\boldsymbol{z}_{i-1}) \text{ for } i \in \{1, ..., n\}, \tag{1}$$

due to the chain rule we have:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{z}_0} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}_{n-1}} \frac{d\boldsymbol{z}_{n-1}}{d\boldsymbol{z}_0} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}_{n-1}} \frac{\partial \boldsymbol{z}_{n-1}}{\partial \boldsymbol{z}_{n-2}} ... \frac{\partial \boldsymbol{z}_1}{\partial \boldsymbol{z}_0} = \frac{\partial g_n(\boldsymbol{z}_{n-1})}{\partial \boldsymbol{z}_{n-1}} \frac{\partial g_{n-1}(\boldsymbol{z}_{n-2})}{\partial \boldsymbol{z}_{n-2}} ... \frac{\partial g_1(\boldsymbol{z}_0)}{\partial \boldsymbol{z}_0}. \tag{2}$$

We can choose each g_i to infinitesimally modify its input, i.e., $g_i(\boldsymbol{z}_{i-1}) = \boldsymbol{z}_{i-1} + \epsilon f_i(\boldsymbol{z}_{i-1}, t_{i-1}, \boldsymbol{\theta})$, so that the chain in Expression 1 transforms $\boldsymbol{z}_0$ continuously. 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 $\frac{d\boldsymbol{z}_n}{d\boldsymbol{z}_0}$ in Equation (2) converges to $\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$, which as we show in Appendix B, satisfies the differential equation below:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{3}$$

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 (3) 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 forward sensitivity formula for ODEs which provides the dynamics of Jacobian of the state with respect to the parameters $\boldsymbol{\theta}$ of the flow (Zhao & Mousseau, 2013). This relation is highlighted and explained in more detail in Appendix B, where in addition, the solution to Equation (3) is provided in Equation (38) by utilizing the Magnus Expansion, (Magnus, 1954; Blanes et al., 2009). Furthermore, we provide the continuous generalization of the total derivative decomposition, and we derive the instantaneous change of variables from the cable rule respectively in Appendix C and E.

AFFJORD, a Special Case of Augmented Neural ODE Flows. Since Neural Ordinary Differential Equations (NODEs) learn representations that preserve the topology of the input space, (Dupont et al., 2019) introduce the Augmented Neural ODEs whose dynamics can be described as follows:

$$w(T) = \begin{bmatrix} \boldsymbol{z}(T) \\ \boldsymbol{z}^*(T) \end{bmatrix} = \begin{bmatrix} \boldsymbol{z}(0) \\ \boldsymbol{z}^*(0) \end{bmatrix} + \int_0^T \begin{bmatrix} f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta}) \\ g(\boldsymbol{z}^*(t), \boldsymbol{z}(t), \boldsymbol{\phi}) \end{bmatrix} dt, \tag{4}$$

where $\boldsymbol{z}^*$ are the augmented dimensions. In this general form, the model is unsuitable to be used in the context of normalizing flows 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. We can use the chain rule to express the Jacobian determinant of this transformation as

$$\left|\det\left(\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right)\right| = \left|\det\left(\frac{d\boldsymbol{z}(T)}{d\left[\boldsymbol{z}(T), \boldsymbol{z}^*(T)\right]}\frac{d\left[\boldsymbol{z}(T), \boldsymbol{z}^*(T)\right]}{d\left[\boldsymbol{z}(0), \boldsymbol{z}^*(0)\right]}\frac{d\left[\boldsymbol{z}(0), \boldsymbol{z}^*(0)\right]}{d\boldsymbol{z}(0)}\right)\right|. \tag{5}$$

and further develop the middle term in Expression 5 via the cable rule (Equation (38), Appendix B) to give the expression of the determinant of the Jacobian of general augmented neural ODE flows. Unfortunately however, computing this expression is not feasible in general.
In order to mitigate these issues, we set $\boldsymbol{z}^*(0)$ to be the zero vector and define g to be independent of the original dimensions. Therefore, $[\boldsymbol{z}(0), \boldsymbol{z}^*(0)]$ is transformed by $h = (f, g)$, as follows:

$$w(T) = \begin{bmatrix}\boldsymbol{z}(T)\\ \boldsymbol{z}^*(T)\end{bmatrix} = \begin{bmatrix}\boldsymbol{z}(0)\\ \boldsymbol{z}^*(0)\end{bmatrix} + \int_0^T \begin{bmatrix}f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})\\ g(\boldsymbol{z}^*(t), \boldsymbol{\phi})\end{bmatrix} dt. \tag{6}$$

We name this model AFFJORD (Augmented FFJORD). We are not interested in $\boldsymbol{z}^*(T)$ as our interest lies on the transformed $\boldsymbol{z}(0)$, that is $\boldsymbol{z}(T)$. 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. A more detailed explanation proving that such a choice alleviates the problems discussed above is provided in Appendix F. 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). In the case that the base distribution is multivariate normal, as shown in Appendix F.2, the expression of the loss function is:

$$L = \frac{||\boldsymbol{Z}(T)||^2}{2} - \int_0^T tr\left(\frac{\partial f(\boldsymbol{z}(t), \boldsymbol{z}^*(t))}{\partial \boldsymbol{z}(t)}\right) dt \tag{7}$$

Multiscale Architecture in Augmented Neural ODE Flows. In the implementation of the model we modify the multiscale architecture in order to accommodate the augmented dimensions. We provide a more detailed description and an illustration of this modification in Appendix I.

3 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, CIFAR-10 and CelebA(32×32). The goal is to train a neural network to generate a vector field $h = (f, g)$ (as in Eq. (6)) that transforms our data distribution into a standard multivariate normal one. The augmented dimensions are 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 $\boldsymbol{\phi}$. In all cases, the architecture of the main field f (in Eq. (6)) and the training time in AFFJORD are identical to that of FFJORD for the sake of fairness. Detailed information about the experiments such as hyper-parameters and network architectures can be found in Appendix I. Limitations and future prospects are discussed in Appendix K.

3.1 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 1. In both cases we use the Runge-Kutta 4 solver with 40 time steps (160 function evaluations). While both FFJORD and AFFJORD are capable of modelling multi-modal and discontinuous distributions, Figure 1 shows that AFFJORD has higher flexibility in modeling the complex data distributions considered, in comparison to FFJORD. We can notice in Table 1 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.

3.2 Image Datasets

We show that AFFJORD outperforms FFJORD on MNIST, CIFAR-10 and CelebA (32×32). There are several architectures of FFJORD that can be used for this application, hence we first optimize the architecture of FFJORD and then add the augmented structure in AFFJORD. Out of the architectures introduced in (Grathwohl et al., 2019) that we tested, the one that performed best was the multiscale 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. 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 we enable the evolution of the vector fields $g(\boldsymbol{z}^*(t), \boldsymbol{\phi})$ through time via 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. Number 20 was chosen as during fine-tuning, the best performance was reached in this setting. 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.

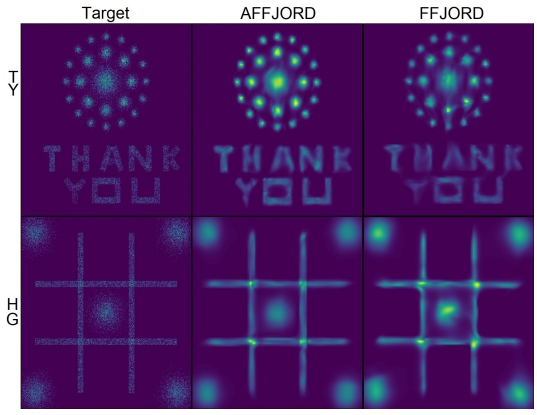

Figure 1: Probability density modeling capabilities of AFFJORD (second column) and FFJORD (third column) on 2D data of toy distributions.

Table 1: Experimental Results for Density Estimation Models, in Bits/Dim for MNIST, CIFAR-10 and CelebA (32×32), and in NLL for the 2D data. Lower Is Better. The Multiscale Architecture Is Used in All Image Data Cases.

Model	MNIST	CIFAR10	CelebA	HG (2D)	TY (2D)
Real NVP	1.06	3.49	-	-	-
Glow	1.05	3.35	-	-	-
FFJORD	0.96 $\pm$.00	3.37 $\pm$.00	3.28$\pm$.00	1.052 $\pm$.007	0.674 $\pm$.017
AFFJORD	$\mathbf{0.95} \pm .01$	$\mathbf{3.32} \pm .01$	$\mathbf{3.23} \pm .00$	$\mathbf{1.035} \pm .008$	$\mathbf{0.636} \pm .02$

As we show in Table 1, AFFJORD outperforms FFJORD on MNIST, CIFAR-10 and CelebA (32×32). Based on the conducted experiments the farther FFJORD is from optimal performance, the greater the improvements brought by AFFJORD are. The calculation of results is done as in (Grathwohl et al., 2019), where for each run the best evaluation result over epochs is taken. After 5 runs, for each model, the scores are averaged and reported in Table 1. In the case of MNIST, both models were trained for roughly 9 days, while in the case of CIFAR-10 and CelebA (32×32), they were trained for approximately 14 days. The results corresponding to the Real NVP and Glow models, are taken from the original papers: (Dinh et al., 2017) and (Kingma & Dhariwal, 2018).
Like its predecessor FFJORD, AFFJORD can generate samples by backintegrating. The generative procedure is explained in more details in Appendix I. Generated samples from both AFFJORD and FFJORD can be found in Appendix J.

4 Conclusion

We presented the generalization of the total derivative decomposition in the continuous sense as well as the continuous generalization of the chain rule. Motivated by the latter, we propose a new type of continuous normalizing flow, namely AFFJORD which outperforms FFJORD in the experiments we conducted.

Acknowledgments and Disclosure of Funding

This work has been supported by the French government, through the 3IA Côte d'Azur Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-19-P3IA-0002. The authors are grateful to the OPAL infrastructure from Université Côte d'Azur for providing resources and support.

References

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. *ArXiv Preprint*.

Blanes, S., Casas, F., Oteo, J., & Ros, J. (2009). The magnus expansion and some of its applications. *Physics Reports*.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*.

Dinh, L., Krueger, D., & Bengio, Y. (2015). Nice: Non-linear independent components estimation. In *Workshop paper, International Conference on Learning Representations*.

Dinh, L., Sohl-Dickstein, J., & Bengio, S. (2017). Density estimation using real nvp. In *International Conference on Learning Representations*.

Dupont, E., Doucet, A., & Teh, Y. W. (2019). Augmented neural odes. In *Advances in Neural Information Processing Systems*.

Finlay, C., Jacobsen, J.-H., Nurbekyan, L., & Oberman, A. (2020). How to train your neural ode. In *International Conference on Machine Learning*.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., & Duvenaud, D. (2019). Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*.

Ho, J., Chen, X., Srinivas, A., Duan, Y., & Abbeel, P. (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International Conference on Learning Representations*.

Hutchinson, M. F. (1990). A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*.

Kingma, D. P., & Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*.

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*.

Krizhevsky, A., Nair, V., & Hinton, G. (2010). Cifar-10 (canadian institute for advanced research). *URL http://www. cs. toronto. edu/kriz/cifar. html*.

Lecun, Y. (1998). The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*. URL `https://cir.nii.ac.jp/crid/1571417126193283840`

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Magnus, W. (1954). On the exponential solution of differential equations for a linear operator. *Communications on Pure and Applied Mathematics*.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., & Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*.

Pontrjagin, L., Boltyanskii, V., Gamkrelidze, R., Mishchenko, E., & Brown, D. (1962). The mathematical theory of optimal processes. In *International series of monographs in pure and applied mathematics*.

Rezende, D. J., & Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*.

Tabak, E. G., & Turner, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*.

Tabak, E. G., & Vanden-Eijnden, E. (2010). Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*.

Zhang, T., Yao, Z., Gholami, A., Gonzalez, J. E., Keutzer, K., Mahoney, M. W., & Biros, G. (2019). Anodev2: A coupled neural ode framework. In *Advances in Neural Information Processing Systems*.

Zhao, H., & Mousseau, V. A. (2013). Extended forward sensitivity analysis for uncertainty quantification. *Nuclear Technology*.

A 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., 2015) 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:

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = f_{\boldsymbol{Z}}(\boldsymbol{z}) \left| \det \left(\frac{d\boldsymbol{z}}{d\boldsymbol{x}} \right) \right| = f_{\boldsymbol{Z}}(h(\boldsymbol{x})) \left| \det \left(\frac{dh(\boldsymbol{x})}{d\boldsymbol{x}} \right) \right|. \tag{8}$$

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 (8), and calculate its likelihood. 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, $\boldsymbol{z}_{i-1} = g_i(\boldsymbol{z}_i),\ i \in [n]$, that is $\boldsymbol{z}_i = h_i(\boldsymbol{z}_{i-1}),\ i \in [n]$. In this case due to chain rule we have:

$$f_{\boldsymbol{Z}_0}(\boldsymbol{z}_0) = f_{\boldsymbol{Z}_n}(\boldsymbol{z}_n) \left| \det \left(\frac{d\boldsymbol{z}_n}{d\boldsymbol{z}_0} \right) \right| = f_{\boldsymbol{Z}_n}(h_n(...h_1(\boldsymbol{z}_0))) \prod_{i=1}^{n} \left| \det \left(\frac{dh_i(\boldsymbol{z}_{i-1})}{d\boldsymbol{z}_{i-1}} \right) \right|, \tag{9}$$

where $\boldsymbol{z}_0$ is a data point. 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:

$$\boldsymbol{z}_{t_{i+1}} = \boldsymbol{z}_i + \epsilon f(\boldsymbol{z}_{t_i}, t_i, \boldsymbol{\theta}) \rightarrow \boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau), \tau, \boldsymbol{\theta}) d\tau, \text{ as } \epsilon \rightarrow 0. \tag{10}$$

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

$$\frac{dL}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt, \tag{11}$$

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

$$\frac{\partial L}{\partial \boldsymbol{z}(t)} = \frac{\partial L}{\partial \boldsymbol{z}(T)} - \int_T^t \frac{\partial L}{\partial \boldsymbol{z}(\tau)} \frac{\partial f(\boldsymbol{z}(\tau), \tau, \boldsymbol{\theta}(\tau))}{\partial \boldsymbol{z}(\tau)} d\tau. \tag{12}$$

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

$$\log p(\boldsymbol{z}(0)) = \log p(\boldsymbol{z}(T)) + \int_0^T tr \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} dt, \tag{13}$$

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., 2019), these ideas are further explored and computational simplifications 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 (13). The resulting model is named FFJORD.

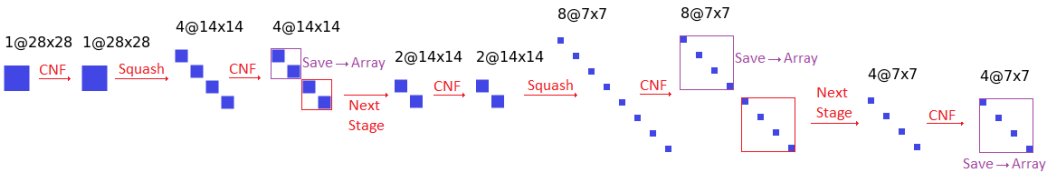

Figure 2: Example of the multiscale architecture on MNIST dataset

Multiscale Architectures: In (Dinh et al., 2017), 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:

$$\frac{d}{dt}\begin{bmatrix}\boldsymbol{z}(t)\\ \boldsymbol{z}^*(t)\end{bmatrix} = h(\begin{bmatrix}\boldsymbol{z}(t)\\ \boldsymbol{z}^*(t)\end{bmatrix}, t) = \begin{bmatrix}f(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})\\ g(\boldsymbol{z}(t), \boldsymbol{z}^*(t), \boldsymbol{\theta})\end{bmatrix}$$
$$\text{for } \begin{bmatrix}\boldsymbol{z}(0)\\ \boldsymbol{z}^*(0)\end{bmatrix} = \begin{bmatrix}\boldsymbol{x}\\ 0\end{bmatrix}, \tag{14}$$

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

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.

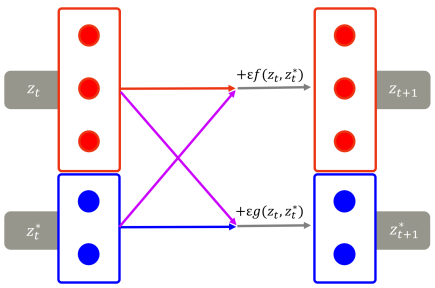

Figure 3: Architecture of discretized augmented neural ODEs.

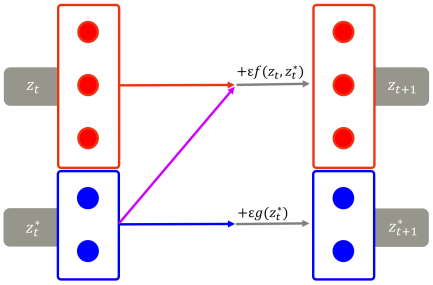

Figure 4: The discretized augmented neural ODE flows implemented in AFFJORD.

B Cable Rule: The Continuous Generalization 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

$$\frac{dz_n}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}}\frac{dz_{n-1}}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}}\frac{\partial z_{n-1}}{\partial z_{n-2}}...\frac{\partial z_1}{\partial z_0} = \frac{\partial f_n(z_{n-1})}{\partial z_{n-1}}\frac{\partial f_{n-1}(z_{n-2})}{\partial z_{n-2}}...\frac{\partial f_1(z_0)}{\partial z_0} \tag{15}$$

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

$$\frac{dz_n}{dz_0} = \frac{\partial z_n}{\partial z_{n-1}}\frac{\partial z_{n-1}}{\partial z_{n-2}}...\frac{\partial z_1}{\partial z_0} = (I + \epsilon\frac{\partial f(z_{n-1})}{\partial z_{n-1}})(I + \epsilon\frac{\partial f(z_{n-2})}{\partial z_{n-2}})...(I + \epsilon\frac{\partial f(z_0)}{\partial z_0}) \tag{16}$$

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 16, we obtain

$$\frac{dz_n}{dz_0} = I + \sum_{i=0}^{n-1} \frac{\partial f(z_i)}{\partial z_i}\epsilon + \sum_{i=0}^{n-1}\sum_{j<i} \frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\epsilon^2 + ... + \frac{\partial f(z_0)}{\partial z_0}...\frac{\partial f(z_n)}{\partial z_n}\epsilon^n \tag{17}$$

$$= I + S_1^n + S_2^n + ... + S_{n+1}^n, \tag{18}$$

where

$$S_2^n = \sum_{i=0}^{n-1}\sum_{j<i} \frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\epsilon^2,\ S_3^n = \sum_{i=0}^{n-1}\sum_{j<i}\sum_{k<j} \frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\frac{\partial f(z_k)}{\partial z_k}\epsilon^3, ... \tag{19}$$

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

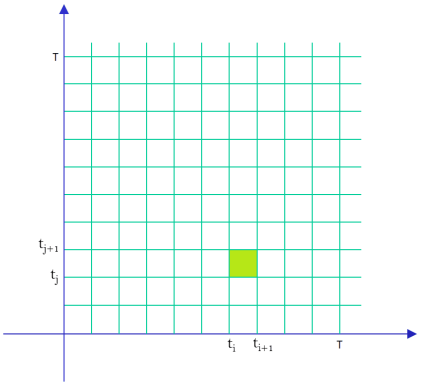

Figure 5: Rectangle (i, j) in discretized $[0, T] \times [0, T]$

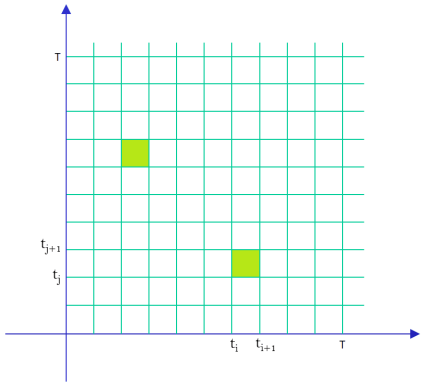

Figure 6: Symmetry of g_2^n

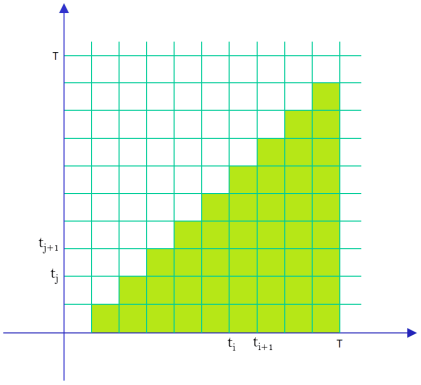

Figure 7: Rectangles participating in S_2^n

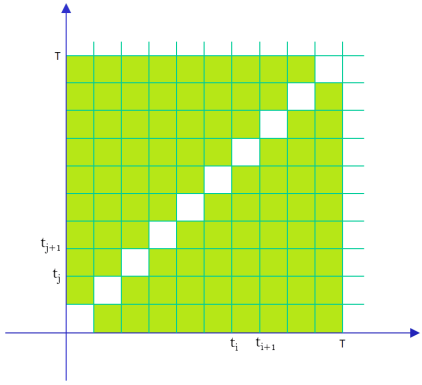

Figure 8: Rectangles in $\bar{S}_2^n$

$$g_2^n(x, y) = \frac{\partial f(z_{t_i})}{\partial z_{t_i}}\frac{\partial f(z_{t_j})}{\partial z_{t_j}},\ \text{for } x \in [t_i, t_{i+1}], y \in [t_j, t_{j+1}]. \tag{20}$$

It is clear that g_2^n is the discretization of

$$g_2(t, u) = \frac{\partial f(z_t)}{\partial z_t}\frac{\partial f(z_u)}{\partial z_u},\ \text{for } t \in [0, T], u \in [0, T]. \tag{21}$$

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 7. If we decide to expand the sum S_2^n on the rectangles above the diagonal as in Figure 8 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

$$\bar{S}_2^n = \sum_{i=0}^{n-1}\sum_{j\neq i}\frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\epsilon^2 \to \int_{[0,T]\times[0,T]} g_2(t,u)d(t,u) = (\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^2, \tag{22}$$

thus,

$$S_2^n = \frac{1}{2!}\bar{S}_2^n \to \frac{1}{2!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^2. \tag{23}$$

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

$$g_3^n(x,y,z) = \frac{\partial f(z_{t_i})}{\partial z_{t_i}}\frac{\partial f(z_{t_j})}{\partial z_{t_j}}\frac{\partial f(z_{t_k})}{\partial z_{t_k}}, \text{ for } x\in[t_i,t_{i+1}], y\in[t_j,t_{j+1}], z\in[t_k,t_{k+1}], \tag{24}$$

and

$$g_3(t,u,v) = \frac{\partial f(z_t)}{\partial z_t}\frac{\partial f(z_u)}{\partial z_u}\frac{\partial f(z_v)}{\partial z_v}, \text{ for } t\in[0,T], u\in[0,T], v\in[0,T]. \tag{25}$$

In this case:

$$\begin{aligned} g_3^n(x,y,z) = g_3^n(x,y,z) = g_3^n(x,z,y) = g_3^n(y,x,z) = \\ = g_3^n(y,z,x) = g_3^n(z,x,y) = g_3^n(z,y,x), \end{aligned} \tag{26}$$

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:

$$\bar{S}_3^n = \sum_{i=0}^{n-1}\sum_{j\neq i}\sum_{i\neq k\neq j}\frac{\partial f(z_i)}{\partial z_i}\frac{\partial f(z_j)}{\partial z_j}\frac{\partial f(z_k)}{\partial z_k}\epsilon^3 \to \int_{[0,T]\times[0,T]\times[0,T]} g_3(t,u,v)d(t,u,v) \tag{27}$$

$$= (\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^3, \tag{28}$$

implying

$$S_3^n = \frac{1}{3!}\bar{S}_3^n \to \frac{1}{3!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^3. \tag{29}$$

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:

$$\frac{dz(T)}{dz(0)} = \frac{1}{0!}I + \frac{1}{1!}\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt + \frac{1}{2!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^2 + \frac{1}{3!}(\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt)^3 + ... \tag{30}$$

$$\frac{dz(T)}{dz(0)} = e^{\int_0^T \frac{\partial f(z_t)}{\partial z_t}dt}. \tag{31}$$

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 31 that $\frac{dz(T)}{dz(0)}$ is the solution of the following ODE:

$$\frac{d(\frac{dz(t)}{dz(0)})}{dt} = \frac{\partial f(z(t))}{\partial z(t)} \frac{dz(t)}{dz(0)} \tag{32}$$

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:

$$\boldsymbol{z}(t) = \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau))d\tau = g(\boldsymbol{z}(0), t), \tag{33}$$

therefore,

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial^2 g(\boldsymbol{z}(0),t)}{\partial t \partial \boldsymbol{z}(0)} = \frac{\partial^2 g(\boldsymbol{z}(0),t)}{\partial \boldsymbol{z}(0) \partial t} =$$
$$= \frac{d(\frac{dg(\boldsymbol{z}(0),t)}{dt})}{d\boldsymbol{z}(0)} = \frac{df(\boldsymbol{z}(t))}{d\boldsymbol{z}(0)} = \frac{df(\boldsymbol{z}(t))}{d\boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{34}$$

We pause for a moment, in order to highlight the similarity of expression 34 and the forward sensitivity:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{\theta}})}{dt} = \frac{df(\boldsymbol{z}(t),t,\boldsymbol{\theta})}{d\boldsymbol{\theta}} = \frac{\partial f(\boldsymbol{z}(t),t,\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{\theta}} + \frac{\partial f(\boldsymbol{z}(t),t,\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{35}$$

Now from Expression 34, we infer that $\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}$ is the solution of the following linear ODE:

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{36}$$

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

$$\frac{d\boldsymbol{Y}(t)}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \boldsymbol{Y}(t). \tag{37}$$

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:

$$\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(t)} \frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(0)} = e^{\boldsymbol{\Omega}(t)}, \text{ for } \boldsymbol{\Omega}(t) = \sum_{k=1}^{\infty} \boldsymbol{\Omega}_k(t), \tag{38}$$

where

$$\boldsymbol{\Omega}_1(t) = \int_0^t \boldsymbol{A}(t_1)\, dt_1, \tag{39}$$

$$\boldsymbol{\Omega}_2(t) = \frac{1}{2} \int_0^t dt_1 \int_0^{t_1} dt_2\, [\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)], \tag{40}$$

$$\boldsymbol{\Omega}_3(t) = \tag{41}$$

$$= \frac{1}{6} \int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3 \left(\big[\boldsymbol{A}(t_1), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_3)]\big] + \big[\boldsymbol{A}(t_3), [\boldsymbol{A}(t_2), \boldsymbol{A}(t_1)]\big] \right), \tag{42}$$

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

$$\boldsymbol{A}(t) = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{43}$$

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 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 $\boldsymbol{z}(t)$, furthermore let $\boldsymbol{z}(t) = \boldsymbol{z}_t(\boldsymbol{z}(0), t)$ satisfy the conditions of Clairaut's theorem. Then the following holds:*

$$\frac{d(\frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)})}{dt} = \frac{\partial f(\boldsymbol{z}(t))}{\partial \boldsymbol{z}(t)} \frac{d\boldsymbol{z}(t)}{d\boldsymbol{z}(0)}. \tag{44}$$

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. □

C The Continuous Generalization of the Total Derivative Decomposition

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.

In Appendix B, 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 9. Thus

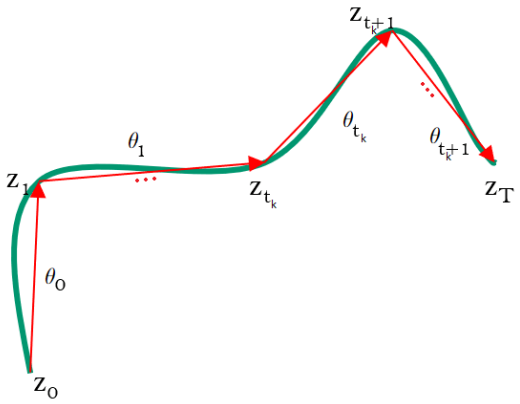

Figure 9: The dependencies in the discretisation of $\frac{\boldsymbol{z}(t)}{t} = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$. Variable $\boldsymbol{z}_{t_{k+1}}$ completely absorbs the contribution of $\boldsymbol{\theta}_{t_k}$ to $\boldsymbol{z}_T$, hence $\boldsymbol{z}_T$ is a function of $\boldsymbol{z}_{t_{k+1}}$ and the following sets of parameters $\boldsymbol{\theta}_{t_{k+1}}, \boldsymbol{\theta}_{t_{k+2}}, ..., \boldsymbol{\theta}_{t_n}$.

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \frac{d\boldsymbol{z}_n(\boldsymbol{z}(t_k + \epsilon), \boldsymbol{\theta}(t_k + \epsilon), \boldsymbol{\theta}(t_{k+2}), ..., \boldsymbol{\theta}(t_n))}{d\boldsymbol{\theta}(t_k)} \tag{45}$$

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{d\boldsymbol{z}(t_k+\epsilon)}{d\boldsymbol{\theta}(t_k)} + 0 + 0 + ... + 0 = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)}. \tag{46}$$

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:

$$\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{z}(t_k)} = \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} (I + \epsilon \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{z}(t_k)} + O(\epsilon^2)) \tag{47}$$

we get

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

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

$$\frac{\partial \boldsymbol{z}(t_k+\epsilon)}{\partial \boldsymbol{\theta}(t_k)} = \frac{\partial (\boldsymbol{z}(t_k) + f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))\epsilon + O(\epsilon^2))}{\partial \boldsymbol{\theta}(t_k)} = 0 + \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \epsilon + O(\epsilon^2). \tag{49}$$

After combining both Equation 48 and Equation 49 in expression 46, we deduce that:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}(t_k)} =$$

$$\big(\frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} - \epsilon \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k+\epsilon)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{z}(t_k)} + O(\epsilon^2)\big)\big(\frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \epsilon + O(\epsilon^2)\big), \tag{50}$$

thus

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

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 $\boldsymbol{\theta}$:

$$\frac{d\boldsymbol{z}_n}{d\boldsymbol{\theta}} = \sum_{k=0}^{n} \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{\theta}(t_k)} \frac{d\boldsymbol{\theta}(t_k)}{d\boldsymbol{\theta}} = \sum_{k=0}^{n} \frac{\partial \boldsymbol{z}_n}{\partial \boldsymbol{z}(t_k)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \frac{d\boldsymbol{\theta}(t_k)}{d\boldsymbol{\theta}} \epsilon + O(\epsilon^2) \tag{52}$$

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

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{\theta}} = \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}} dt. \tag{53}$$

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

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

$$= \int_0^T \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt = -\int_T^0 \frac{\partial \boldsymbol{z}(T)}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} dt. \tag{54}$$

Theorem 2. *Let $\frac{d\boldsymbol{z}(t)}{dt} = f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)$, where f is continuous in t and Lipschitz continuous in $\boldsymbol{z}(t)$ and $\boldsymbol{\theta}(t)$. Then the following holds:*

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

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. 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. □

D 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 $\frac{T}{2}$):

$$\boldsymbol{z}(t) = \begin{cases} \boldsymbol{z}(0) + \int_0^t f(\boldsymbol{z}(\tau), \boldsymbol{\theta}(\tau))d\tau, \ t \in [0, \frac{T}{2}] \\ \boldsymbol{z}(0) + \int_0^{\frac{T}{2}} f(\boldsymbol{z}(\tau), \boldsymbol{\theta}(\tau))d\tau + \int_{\frac{T}{2}}^t g(\boldsymbol{z}(\tau), \boldsymbol{\phi}(\tau))d\tau, \ t \in (\frac{T}{2}, T] \end{cases} \tag{56}$$

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

$$\frac{d(\frac{\partial L}{\partial \boldsymbol{z}(t)})}{dt} = \begin{cases} -\frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{z}(t)}, \ t \in [0, \frac{T}{2}] \\ -\frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi}(t))}{\partial \boldsymbol{z}(t)}, \ t \in (\frac{T}{2}, T], \end{cases} \tag{57}$$

thus

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

The approach developed in Appendix C can be used to generalize continuous backpropagation into piecewise continuous backpropagation. Indeed, identically as before, we have the first order appoximation:

$$\frac{dL_\epsilon}{d\boldsymbol{\theta}} = \sum_{k=0}^{n} \frac{\partial L_\epsilon}{\partial \boldsymbol{z}(t_k)} \frac{\partial f(\boldsymbol{z}(t_k), \boldsymbol{\theta}(t_k))}{\partial \boldsymbol{\theta}(t_k)} \frac{\partial \boldsymbol{\theta}(t_k)}{\partial \boldsymbol{\theta}} \epsilon + \sum_{k=0}^{n} \frac{\partial L_\epsilon}{\partial \boldsymbol{z}(t_k)} \frac{\partial g(\boldsymbol{z}(t_k), \boldsymbol{\phi}(t_k))}{\partial \boldsymbol{\phi}(t_k)} \frac{\partial \boldsymbol{\phi}(t_k)}{\partial \boldsymbol{\theta}} \epsilon \tag{59}$$

Taking the limit we get:

$$\frac{dL}{d\boldsymbol{\theta}} = \int_0^{\frac{T}{2}} \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\theta}(t)}{\partial \boldsymbol{\theta}} dt + \int_{\frac{T}{2}}^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi}(t))}{\partial \boldsymbol{\theta}(t)} \frac{\partial \boldsymbol{\phi}(t)}{\partial \boldsymbol{\theta}} dt \tag{60}$$

and in the same manner we get:

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

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

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

and

$$\frac{dL}{d\boldsymbol{\phi}} = \int_{\frac{T}{2}}^T \frac{\partial L}{\partial \boldsymbol{z}(t)} \frac{\partial g(\boldsymbol{z}(t), \boldsymbol{\phi})}{\partial \boldsymbol{\phi}} dt. \tag{63}$$

E Deriving the Instantaneous Change of Variable via the Cable Rule

The trace of a commutator $[\boldsymbol{X}, \boldsymbol{Y}] = \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{X}$ is always zero, hence we prove below that for all terms $\boldsymbol{\Omega}_{k>1}(t)$ given in Equations (39), (40) and (41) in Appendix B, we have $tr(\boldsymbol{\Omega}_k(t)) = 0$.

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

$$\begin{aligned} tr(\boldsymbol{\Omega}_2(t)) &= tr\frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2 \, [\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)] \\ &= \frac{1}{2}\int_0^t dt_1 \int_0^{t_1} dt_2 \, tr[\boldsymbol{A}(t_1), \boldsymbol{A}(t_2)] = 0. \end{aligned} \tag{64}$$

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

$$+tr\big[\boldsymbol{A}(t_3),[\boldsymbol{A}(t_2),\boldsymbol{A}(t_1)]\big]\Big) \tag{65}$$

$$=\frac{1}{6}\int_0^t dt_1\int_0^{t_1}(0+0)dt_2=0, \tag{66}$$

and so on for $k>3$.

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

$$tr(\boldsymbol{\Omega}_1(t))=tr\int_0^t \boldsymbol{A}(t_1)dt_1=\int_0^t tr\frac{\partial f(\boldsymbol{z}(t_1),t_1,\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)}dt_1. \tag{67}$$

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

$$\log\left|\det\left(\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right)\right|=\log\left|\det\left(e^{\boldsymbol{\Omega}(T)}\right)\right|=\log e^{tr(\boldsymbol{\Omega}(T))}=\int_0^T tr\frac{\partial f(\boldsymbol{z}(t),t,\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}dt. \tag{68}$$

F Simplifications and the Derivation of the Loss in Section 2

F.1 Simplifications

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

$$\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}=[I,0]\,e^{\begin{bmatrix}\int_0^T\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}dt & \int_0^T\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)}dt\\ \int_0^T\frac{\partial g(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}(t)}dt & \int_0^T\frac{\partial g(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)}dt\end{bmatrix}+\boldsymbol{\Omega}_2(T)+\dots}\begin{bmatrix}I\\ \frac{d\boldsymbol{z}^*(0)}{d\boldsymbol{z}(0)}\end{bmatrix}, \tag{69}$$

becomes

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

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

$$\boldsymbol{A}(t)=\begin{bmatrix}\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} & \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)}\\ \frac{\partial g(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}(t)} & \frac{\partial g(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)}\end{bmatrix}\to\begin{bmatrix}\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)} & \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t)}\\ 0 & \frac{\partial g(\boldsymbol{z}^*(t),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t)}\end{bmatrix}. \tag{71}$$

Hence,

$$\boldsymbol{\Omega}_1(t)=\int_0^t dt_1\begin{bmatrix}\frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}^*(t_1)}\\ 0 & \frac{\partial g(\boldsymbol{z}^*(t_1),\boldsymbol{\phi})}{\partial \boldsymbol{z}^*(t_1)}\end{bmatrix}=\begin{bmatrix}\int_0^t dt_1\frac{\partial f(\boldsymbol{z}(t_1),\boldsymbol{z}^*(t_1),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t_1)} & \boldsymbol{B}_1(t)\\ 0 & \boldsymbol{D}_1(t)\end{bmatrix} \tag{72}$$

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

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

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

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

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:

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

Finally,

$$\left|\det\left(\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right)\right| = \left|\det\left([I,0]\begin{bmatrix} e^{\mathbf{\Omega}^{[z]}(t)} & \bar{\boldsymbol{B}}(t) \\ 0 & \bar{\boldsymbol{D}}(t) \end{bmatrix}\begin{bmatrix} I \\ 0 \end{bmatrix}\right)\right| =$$

$$= \left|\det\left(e^{\mathbf{\Omega}^{[z]}(t)}\right)\right| = e^{\int_0^T tr\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}dt+0+\ldots 0+\ldots}. \tag{73}$$

F.2 AFFJORD Alleviates the Issues Listed in Section 2

Regarding issue 1), we have the following:

Proposition 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 since as we proved above, $\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}$ simplifies to

$$[I,0]\begin{bmatrix} e^{\int_0^T \frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}dt+\mathbf{\Omega}_2^{[z]}(T)+\ldots} & \bar{\boldsymbol{B}}(T) \\ 0 & \bar{\boldsymbol{D}}(T) \end{bmatrix}\begin{bmatrix} I \\ 0 \end{bmatrix}, \tag{74}$$

and $= \left|\det\left(e^{\mathbf{\Omega}^{[z]}(t)}\right)\right| = e^{\int_0^T tr\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}dt}$. In case that the base distribution is multivariate normal, from

$$-\log p(\boldsymbol{z}(0)) = -\log\left[p(\boldsymbol{z}(T))\left|\det\left(\frac{d\boldsymbol{z}(T)}{d\boldsymbol{z}(0)}\right)\right|\right]$$

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

we derive the following loss function:

$$L = \frac{||\boldsymbol{Z}(T)||^2}{2} - \int_0^T tr\frac{\partial f(\boldsymbol{z}(t),\boldsymbol{z}^*(t),\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}dt. \tag{75}$$

G Cable Rule Derived via Equation (12)

Differentiating Equation (12) in Appendix B, we get:

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

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

$$\frac{d\left(\frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(t)}\right)}{dt} = -\frac{d\boldsymbol{z}(0)}{d\boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t),t,\boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{77}$$

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

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

thus

$$-\boldsymbol{B}(t)\frac{d\boldsymbol{A}(t)}{dt} = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{79}$$

Then from

$$\boldsymbol{B}(t)\boldsymbol{A}(t) = I, \tag{80}$$

we have

$$-\boldsymbol{B}(t)\frac{d\boldsymbol{A}(t)}{dt} = \frac{d\boldsymbol{B}(t)}{dt}\boldsymbol{A}(t), \tag{81}$$

thus, using this expression in Equation (79) we get

$$\frac{d\boldsymbol{B}(t)}{dt}\boldsymbol{A}(t) = \frac{\partial f(\boldsymbol{z}(t), t, \boldsymbol{\theta})}{\partial \boldsymbol{z}(t)}. \tag{82}$$

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 Appendix C. 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 (53)) from continuous backpropagation (Equation (11), Appendix A). 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,

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{\theta}} &= \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)}\frac{dh(\boldsymbol{z}(t), \boldsymbol{\theta}, t)}{d\boldsymbol{\theta}}dt \\ &= \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)}\frac{df(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)}{d\boldsymbol{\theta}}dt,\end{aligned} \tag{83}$$

therefore

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \int_0^T \frac{\partial L}{\partial \boldsymbol{z}(t)}\frac{\partial f(\boldsymbol{z}(t), \boldsymbol{\theta}(t), t)}{\partial \boldsymbol{\theta}(t)}\frac{d\boldsymbol{\theta}(t)}{d\boldsymbol{\theta}}dt. \tag{84}$$

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

I Additional Details About the Experiments

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 image data we use Dopri5, as well as the FFJORD implementation of (Grathwohl et al., 2019). We use a batch size of 200 for image data and a batch size of 512 for the toy datasets. In the case of image data, we use a learning rate of 6×10^{-4}, while for toy data the learning rate is set to 10^{-3}. All experiments were performed on a single GPU.

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

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

Dataset	Concat	
	FFJORD	AFFJORD
MNIST	400323	417629
CIFAR10	679441	820815
CelebA (32×32)	679441	820815

Dataset	Hypernet	
	FFJORD	AFFJORD
MNIST	783019	4556169
CIFAR10	1332361	8976115
CelebA (32×32)	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 (0.96 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.

I.1 Multiscale Architecture in Augmented Neural ODE Flows

With reference to Figure 10, the multiscale architecture in the augmented case performs an Augmented Continuous Flow on the data as described in Section 2, 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). The original multiscale architecture is presented in Appendix A for comparison.

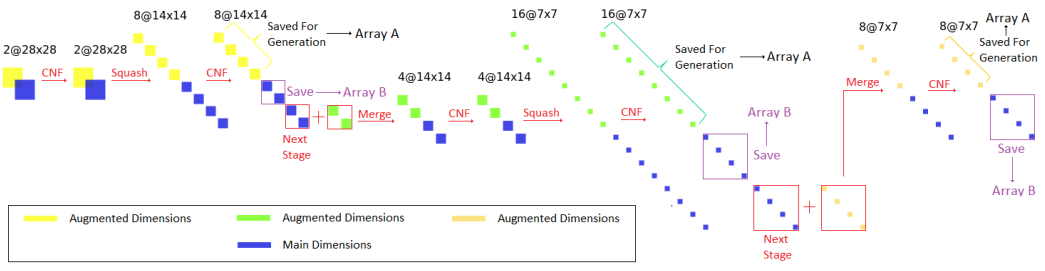

Figure 10: Example of the augmented multiscale architecture on MNIST dataset

I.2 The Generative Procedure

As we mentioned earlier, 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 $\boldsymbol{z}_s(0)$. 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. This is why in Figure 10 at the end of each scale i we save the augmented dimensions $\boldsymbol{z}^*(T)_i$.

I.3 Validation Results During Training

Additional results illustrating the training and validation loss through-out the training procedure are provided in Figures 11, 12 and 13.

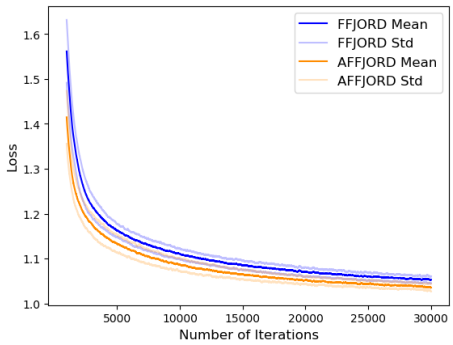

Figure 11: Performance of FFJORD and AFFJORD on the Hash-Gaussian toy 2D dataset.

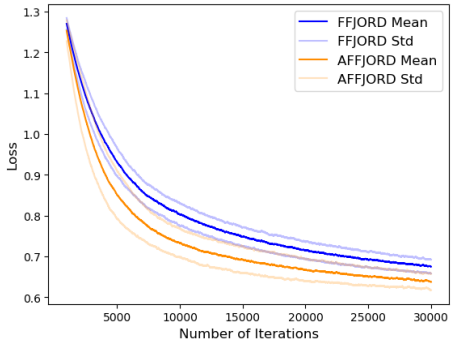

Figure 12: Performance of FFJORD and AFFJORD on the TY toy 2D dataset.

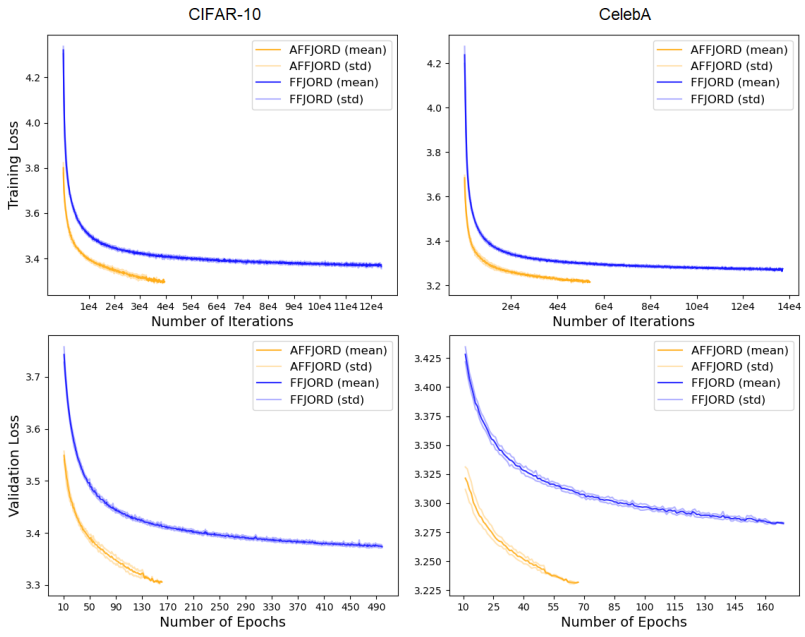

Figure 13: Graphs of training and evaluation losses of FFJORD and AFFJORD on CIFAR-10 as well as CelebA (32×32). We run the experiments 5 times. Lower is better.

J Generated Samples

Below in Figure 14, Figure 15 and Figure 16 , are presented additional examples of generated samples of MNIST, CIFAR-10 and CelebA(32×32) by AFFJORD, respectively.

Figure 14: Generated samples of MNIST by AFFJORD

Figure 15: Generated samples of CIFAR-10 by AFFJORD

Figure 16: Generated samples of CelebA(32×32) by AFFJORD

In addition in Figure 17, Figure 18 and Figure 19, are presented additional examples of generated samples of MNIST, CIFAR-10 and CelebA(32×32) by FFJORD, respectively.

Figure 17: Generated samples of MNIST by FFJORD

Figure 18: Generated samples of CIFAR-10 by FFJORD

Figure 19: Generated samples of CelebA(32×32) by FFJORD

K Limitations and Future Work

Number of Function Evaluations (NFE). As originally reported in (Grathwohl et al., 2019), 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 20. 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 with an increasing number of integration steps.

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.

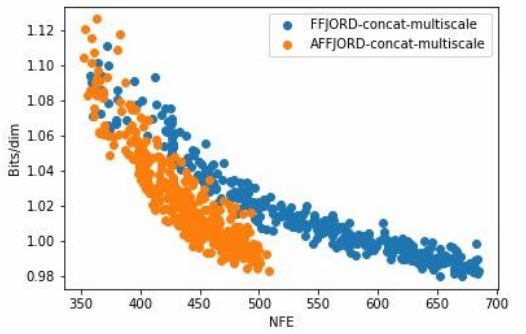

Figure 20: Number of function evaluations of FFJORD and AFFJORD when the 'concatenate' architecture is used.

Data Dependent Augmented Dimensions As described in Section 2, 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 (5) 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.