Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 54-9 (2021) 620–623

Deep learning as optimal control problems [⋆]

Martin Benning [*] Elena Celledoni [**] Matthias J. Ehrhardt [***] Brynjulf Owren [****] Carola-Bibiane Schönlieb [†]

[*] *School of Mathematical Sciences, Queen Mary University of London. London E1 4NS, UK (m.benning@qmul.ac.uk)*
[**] *Department of Mathematical Sciences, NTNU, 7491 Trondheim, Norway (e-mail: elena.celledoni@ntnu.no)*
[***] *Institute for Mathematical Innovation, University of Bath, Bath BA2 7JU, UK (e-mail: me549@bath.ac.uk)*
[****] *Department of Mathematical Sciences, NTNU, 7491 Trondheim, Norway (e-mail: brynjulf.owren@ntnu.no)*
[†] *Department of Applied Mathematics and Theoretical Physics, University of Cambridge, Cambridge CB3 0WA, UK(e-mail:cbs31@cam.ac.uk)*

Abstract: We briefly review recent work where deep learning neural networks have been interpreted as discretisations of an optimal control problem subject to an ordinary differential equation constraint. We report here new preliminary experiments with implicit symplectic Runge-Kutta methods. In this paper, we discuss ongoing and future research in this area.

Keywords: Deep Neural Networks, ResNet, Optimal Control.

1. INTRODUCTION

This work is inspired by the interpretation of deep learning algorithms as discrete optimal control problems subject to an ordinary differential equation (ODE) constraint, see E (2017), Haber et al. (2018). A number of new deep learning architectures can be generated adopting this point of view, Li et al. (2018), Chang et al. (2018), Chen et al. (2018), Ruthotto et al. (2018), Gholami et al. (2019), Zhang et al. (2019). In Benning et al. (2019), we have clarified the connection of general classical Runge-Kutta discretizations of the optimal control ODEs with the deep learning architectures of ResNet type. Discretizing the ODEs with symplectic, partitioned Runge-Kutta methods leads to explicit formulae for the evaluation of the gradients in the gradient descent iterations. The differential equation setting lends itself to learning additional parameters such as the time discretisation. We compare these deep learning algorithms numerically in terms of induced flow and generalisation ability.

⋆ MB acknowledges support from the Leverhulme Trust Early Career Fellowship ECF-2016-611 'Learning from mistakes: a supervised feedback-loop for imaging applications'. CBS acknowledges support from the Leverhulme Trust project on Breaking the non-convexity barrier, the Philip Leverhulme Prize, the EPSRC grant No. EP/M00483X/1, the EPSRC Centre No. EP/N014588/1, the European Union Horizon 2020 research and innovation programmes under the Marie Skodowska-Curie grant agreement No. 777826 NoMADS and No. 691070 CHiPS, the Cantab Capital Institute for the Mathematics of Information and the Alan Turing Institute. We gratefully acknowledge the support of NVIDIA Corporation with the donation of a Quadro P6000 and a Titan Xp GPU used for this research. EC and BO thank the SPIRIT project (No. 231632) under the Research Council of Norway FRIPRO funding scheme. This work was supported by EPSRC grant No. EP/K032208/1.

An artificial neural network is a network of processing units (neurons) each taking inputs and producing outputs. Let $\mathbf{y}^{[in]} \in \mathcal{D}$ denote one sample of the input data, then the corresponding output $\mathbf{y}^{[out]}$ is obtained by applying a composition of maps ϕ_ℓ each depending on parameters $U^{[\ell]}$:

$$\mathbf{y}^{[out]} = \phi_N \circ \cdots \circ \phi_\ell \circ \cdots \circ \phi_1 \left(\mathbf{y}^{[in]}\right), \quad \phi_\ell(\mathbf{y}^{[\ell]}) = \phi(\mathbf{y}^{[\ell]}, U^{[\ell]}).$$

Each map ϕ_ℓ corresponds to a layer of the neural network. Learning is the process consisting on establishing an appropriate cost function on the network, e.g.

$$\sum_{\mathbf{y}^{[in]} \in \mathcal{D}} J(\mathbf{y}^{[out]}),$$

and minimimising this cost function with respect to the parameters $U^{[1]}, \ldots, U^{[N]}$. This is done in practice by a gradient descent iteration method. Deep neural networks are those with a large number of layers. Increasing the number of layers leads to a larger number of parameters and should correspond to increased predictive power of the neural network. However it is well known that adding layers can lead to networks that are harder to train. The training and validation errors do not always decrease as the number of layers N increases. Deep learning algorithms can suffer from vanishing or exploding gradients in the stochastic gradient descent algorithms which are used to solve the optimisation problem. It is reasonable to think that this problematic behaviour of the algorithms has to do with questions of stability and convergence as N goes to infinity. It was observed in He et al. (2016) that the ResNet architecture is more suitable to deep learning compared to other architectures. In the same paper experiments with the ResNet architecture are reported with a large number of layers showing that indeed ResNet can be trained and gives improved training and validation errors

10.1016/j.ifacol.2021.06.124

as the number of layers increases. Starting from the ResNet architecture, and letting N go to infinity one obtains the optimal control continuous problem. The ultimate goals of the present and ongoing research are: good algorithms (stable and converging); more efficient training of deep neural networks; algorithms that generalise well.

2. RESNET AS OPTIMAL CONTROL

A simple version of the ResNet architecture for supervised learning is summarised in what follows.

Setting and definitions:

- $\mathbf{y}^{[in]} \in \mathcal{D}$ data, $\mathbf{y}^{[in]} \in \mathbf{R}^n$;
- training datum $\mathbf{y}^{[in]}$ with training label $c^{[in]} \in \mathbf{R}^d$;
- N the number of layers;
- $U^{[\ell]} := (K^{[\ell]}, \mathbf{b}^{[\ell]})$ parameters to be learned: $K^{[\ell]} \in \mathbf{R}^{n\times n}$, $\mathbf{b}^{[\ell]} \in \mathbf{R}^n$, $\ell = 0, \ldots, N$;
- $\mathcal{J}(\mathbf{y}^{[out]}) := \frac{1}{2} \sum_{\mathbf{y}^{[in]}\in\mathcal{D}} \|\mathcal{C}(K^{[out]}\mathbf{y}^{[out]} + \mathbf{b}^{[out]}) - c^{[in]}\|_2^2$ cost function, $\mathcal{C} : \mathbf{R} \to \mathbf{R}$ classifier applied component-wise on vectors [1];
- $f(\mathbf{y}^{[\ell]}, U^{[\ell]}) := \sigma(K^{[\ell]}\mathbf{y}^{[\ell]} + \mathbf{b}^{[\ell]})$, $\sigma : \mathbf{R} \to \mathbf{R}$ is the activation function acting component-wise on vectors [2];
- $\mathbf{y}^{[0]} := \mathbf{y}^{[in]}$, and $\mathbf{y}^{[out]} := \mathbf{y}^{[N]}$.

Deep learning problem (ResNet):

Consider the optimisation problem

$$\min_{U^{[\ell]},\, \ell=0,\ldots,N} \mathcal{J}(\mathbf{y}^{[N]}), \tag{1}$$

subject to

$$\mathbf{y}^{[\ell+1]} = \varphi_{h_\ell}(\mathbf{y}^{[\ell]}, U^{\ell}), \quad \mathbf{y}^{[0]} := \mathbf{y}^{[in]},\, \mathbf{y}^{[out]} := \mathbf{y}^{[N]}, \tag{2}$$

where

$$\varphi_{h_\ell}(\mathbf{y}^{[\ell]}) := \mathbf{y}^{[\ell]} + h_\ell\, f(\mathbf{y}^{[\ell]}, U^{[\ell]}). \tag{3}$$

Observing that the constraint equation (2) and (3) of this optimisation problem is the forward Euler discretization of the ODE $\dot{\mathbf{y}}(t) = f(\mathbf{y}(t), U(t))$, the deep learning problem (1), (2) and (3) can be seen as the discretization of a continuous optimal control problem, see Haber et al. (2018) and Li et al. (2018):

Optimal control problem:

$$\min_{U(t)} \mathcal{J}(\mathbf{y}(T)), \quad t \in [0, T] \tag{4}$$

subject to

$$\dot{\mathbf{y}} = f(\mathbf{y}, U), \quad \mathbf{y}(0) = \mathbf{y}^{[in]}. \tag{5}$$

The optimal control point of view is useful for several reasons, in fact

- it can be used to create new architectures;
- experience shows that continuous models are useful simplifications of the reality and they are easier to study;
- it offers a starting point for analysis.

3. FIRST ORDER OPTIMALITY AND PARTITIONED SYMPLECTIC RUNGE-KUTTA METHODS

In this section, we discuss necessary conditions for optimality of the deep learning optimal control problem (4) (5) leading to a Hamiltonian boundary value problem (BVP), and the use of different Runge-Kutta discretizations of the underlying continuous deep learning problem. The usual numerical approach to optimal control consists in first discretizing the cost function $\mathcal{J}$ and then solving the discrete optimisation problem, and leads to a gradient descent method for determining the control parameters U. If the discretization method for the ODE is the forward Euler method, this way of proceeding reproduces the ResNet optimization problem formulated in (1), (2) and (3). An alternative (indirect) approach is to first optimise then discretize. These two approaches are equivalent provided the method used for discretizing the Hamiltonian boundary value problem is a symplectic partitioned Runge-Kutta method, see Hager (2000) and Sanz-Serna (2015). In the deep learning context, the second approach, using symplectic, partitioned Runge-Kutta methods, advantageously provides explicit formulae for the gradients, these could be useful for the implementation but are perhaps even more useful for analysis.

First order necessary conditions for optimality of the optimal control problem are obtained via the Pontryagin maximum principle Pontryagin et al. (1986). We denote by $\mathbf{y}$ one sample of data propagated through the neural network.

Theorem 1. The boundary value system

$$\dot{\mathbf{y}} = f(\mathbf{y}, U), \tag{6}$$

$$\dot{\mathbf{p}} = -\left(\partial_{\mathbf{y}} f(\mathbf{y}(t), U(t))\right)^T \mathbf{p} \tag{7}$$

$$0 = \left(\partial_U f(\mathbf{y}(t), U(t))\right)^T \mathbf{p} \tag{8}$$

with boundary conditions $\mathbf{y}(0) = \mathbf{x}$, $\mathbf{p}(T) = \nabla_{\mathbf{y}}\mathcal{J}|_{\mathbf{y}(T)}$, expresses the first order necessary conditions for optimality of the optimal control problem (4), (5).

The system (6), (7), (8) is a constrained Hamiltonian system with Hamiltonian

$$H(\mathbf{y}, \mathbf{p}, U) = \mathbf{p}^T f(\mathbf{y}, U).$$

The constraint $0 = \left(\partial_U f(\mathbf{y}(t), U(t))\right)^T \mathbf{p}$ implies that this is an index one differential algebraic equation.

Consider a Runge-Kutta method with s stages and Butcher tableau (A, b, c), where $A = \{a_{i,j}\}_{i,j=1}^s$, $b = \{b_i\}_{i=1}^s$, $c = \{c_i\}_{i=1}^s$ are the coefficients of the method. Recall that two Runge-Kutta methods (A, b, c) and $(\tilde{A}, \tilde{b}, \tilde{c})$ with coefficients satisfying

$$\tilde{b}_i = b_i,\ \tilde{c}_i = c_i, \quad b_i\tilde{a}_{i,j} + \tilde{b}_j a_{i,j} - b_i\tilde{b}_j = 0, \quad i, j = 1, \ldots, s,$$

form a symplectic partitioned Runge-Kutta pair, Hairer et al (2006).

[1] In our experiments the data are propagated through the network keeping the same dimension throughout the network until the last layer where $K^{[out]}$ and $\mathbf{b}^{[out]}$ are used to project the data to a space of the same dimension as the label space. This does not need to be the case in general. We assume that $K^{[out]}$ and $\mathbf{b}^{[out]}$ are learned parameters. A possible choice for the classifier is $\mathcal{C}(\xi) := \frac{\exp(\xi)}{1+\exp(\xi)}$.

[2] Various choices of σ are available in the literature, we have used $\sigma(x) = \tanh(x)$ in our experiments, another very popular choice is the so called ReLu function.

$$\begin{tikzcd}
 & \begin{array}{c} \min\limits_{U} J(\mathbf{y}(T)) \\ \text{subject to} \\ \dot{\mathbf{y}} = f(\mathbf{y}) \end{array} \arrow[dl, "\text{discretize}"'] \arrow[dr, "\text{optimize}"] & \\
\begin{array}{c} \min\limits_{U} J(\mathbf{y}^{[N]}) \\ \text{subject to} \\ \mathbf{y}^{[\ell+1]} = \varphi(\mathbf{y}^{[\ell]}) \end{array} \arrow[dr, "\text{optimize}"'] & & \begin{array}{l} \dot{\mathbf{y}} = f(\mathbf{y}),\ \mathbf{y}(0) = \mathbf{y}_0, \\ \dot{\mathbf{p}} = -\left(\partial_{\mathbf{y}} f(\mathbf{y}(t), U(t))\right)^T \mathbf{p} \\ \mathbf{p}(T) = \nabla_{\mathbf{y}} J(\mathbf{y}(T)) \\ 0 = (\partial_U f(\mathbf{y}(t), U(t)))^T \mathbf{p} \end{array} \arrow[dl, "\text{discretize}"] \\
 & \begin{array}{l} \mathbf{y}^{[\ell+1]} = \varphi(\mathbf{y}^{[\ell]}), \\ \mathbf{p}^{[\ell+1]} = \tilde{\varphi}(\mathbf{p}^{[\ell]}), \\ \nabla_{U^{[\ell]}} J^{[\ell]} = 0 \end{array} &
\end{tikzcd}$$

Fig. 1. Here φ and $\tilde{\varphi}$ denote a pair of partitioned, symplectic Runge-Kutta methods, e.g. φ forward Euler, $\tilde{\varphi}$ backward Euler.

We assume to discretize the BVP (6), (7), (8) with the symplectic, partitioned Runge-Kutta method with method (A, b, c) for $\dot{\mathbf{y}} = f(\mathbf{y}, U)$, and $(\tilde{A}, \tilde{b}, \tilde{c})$ for $\dot{\mathbf{p}} = -\left(\partial_{\mathbf{y}} f(\mathbf{y}(t), U(t))\right)^T \mathbf{p}$. Symplectic partitioned Runge-Kutta methods are suited to discretize this system, because they respect the variational nature of the problem, as explained in Theorem 2 and summarised in Figure 1.

Theorem 2. The discrete BVP system

$$\begin{aligned}
\mathbf{y}^{[\ell+1]} &= \mathbf{y}^{[\ell]} + hf(\mathbf{y}^{[\ell]} U^{[\ell]}), \qquad \ell = 1, \dots, N \\
\mathbf{p}^{[\ell+1]} &= \mathbf{p}^{[\ell]} - h\left(\partial_{\mathbf{y}} f(\mathbf{y}^{[\ell]}, U^{[\ell]})\right)^T \mathbf{p}_{[\ell+1]}, \quad \ell = 0, \dots, N-1 \\
0 &= \left(\partial_U f(\mathbf{y}^{[\ell]}, U^{[\ell]})\right)^T \mathbf{p}^{[\ell+1]}, \qquad \ell = 0, \dots, N-1.
\end{aligned}$$

expresses the first order necessary conditions for optimality of the following discrete optimal control problem

$$\min_{(\mathbf{y}^{[\ell]}, U^{[\ell]}),\, \ell=0,\dots,N-1} \mathcal{J}(\mathbf{y}^{[N]}),$$

subject to

$$\mathbf{y}^{[\ell+1]} = \mathbf{y}^{[\ell]} + hf(\mathbf{y}^{[\ell]}, U^{[\ell]}), \quad \mathbf{y}(0) = \mathbf{y}^{[in]}.$$

Remark 1. Theorem 2 here formulated for the symplectic Euler method (combination of the forward Euler for the first equation and backward Euler for the second equation), holds for any symplectic partitioned Runge-Kutta method, see Sanz-Serna (2015).

3.1 Explicit formulae for the gradients

We next assume $U(t)$ to be constant over each time step, so that there is only one set of parameters $U^{[\ell]}$ per time step.

Proposition 1. Let $\mathbf{y}^{[\ell]}$ and $\mathbf{p}^{[\ell]}$ be given by a partitioned Runge-Kutta method applied to the Hamiltonian BVP (6), (7), (8). Then the gradient of the cost function $\mathcal{J}$ with respect to the control parameters $U^{[\ell]}$, $\ell = 1, \dots, N$ is given by

$$\begin{aligned}
P_i^{[\ell]} &= \left(\mathbf{p}^{[\ell+1]} - h\sum_{k=1}^{s} \frac{a_{k,i} b_k}{b_i} \lambda_k^{[\ell]}\right) \\
\lambda_i^{[\ell]} &= -\partial_{\mathbf{y}} f(\mathbf{y}_i^{[\ell]}, U^{[\ell]})^T P_i^{[\ell]} \quad i = 1, \dots, s \\
\partial_{U^{[\ell]}} \mathcal{J}(y^{[N]}) &= h\sum_{i=1}^{s} b_i \partial_{U^{[\ell]}} f(\mathbf{y}_i^{[\ell]}, U^{[\ell]})^T P_i^{[\ell]}.
\end{aligned}$$

3.2 Numerical experiments

In Figure 2, we compare the evolution of the midpoint Runge-Kutta method (which is an implicit, symplectic Runge-Kutta method) and the ResNet. Implicit methods where not implemented in our earlier work. Our implementation of the implicit midpoint architecture is based on the formulae for the gradients given in Proposition 1, avoiding the automatic differentiation of the iteration method employed to solve numerically the nonlinear system of equations at each time step. We have here used $N = 5$ layers with constant step size $h = 0.1$, and performed 5000 iterations of gradient descent with backtracking.

4. CODES AND ONGOING WORK

Our code is available on the University of Cambridge repository under https://doi.org/10.17863/CAM.43231. In this code, we have implemented:

- Arbitrary partitioned symplectic Runge-Kutta formulae with constant step size (the forward method in this code is explicit).
- The ResNet model where the time step h_ℓ is also a learned parameter of the model.
- We have compared with a standard implementation of a `FeedForward` algorithm (not related to ODEs).

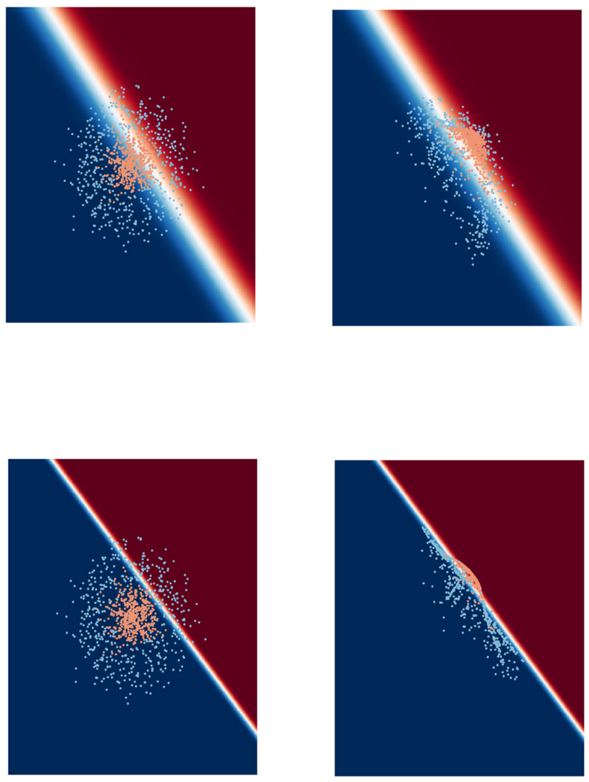

Fig. 2. Dynamics of neural networks based on an implicit, and an explicit Runge-Kutta scheme. Left plots time $t = 0$ right plots time $t = 0.5$; 5 layers, step-size $h = 0.1$.

A number of different experiments are reported in Benning et al. (2019). We found that different RK methods perform rather similarly. This may indicate that the underlying ODE is a common denominator for all the considered architectures and methods, and plays an important role, see also Chen et al. (2018) for similar observations. Leaving the step-size h as a parameter in the model improves the performance and leads to some interesting effects that could be investigated further.

In the boundary value problem (6), (7), (8), the Hessian $\partial_{U,U}H$ is often nearly singular. A possible interesting avenue for future research could consider imposing additional (manifold) constraints to the optimisation problem and in turn to the Hamiltonian boundary value problem with the purpose of obtaining the invertibility of the Hessians. Partitioned, symplectic, Runge-Kutta methods among which the popular RATTLE method could be used in this case.

One aspect that we want to study is the impact that different RK discretisations have on the robustness of neural network predictions with respect to adversarial attacks (see for instance Goodfellow et al. (2014)). Adversarial attacks are small perturbations of the network input, designed to cause large errors in the network output. As such perturbations are only possible for unstable discretisations of potentially unstable dynamic systems, we plan to design and analyse neural networks based on RK discretisations for a stable underlying dynamic system and compare them to non-RK approaches.

While so far our focus has been to consider the case when the number of layers grows another interesting aspect is to investigate what happens when increasing the number of channels. This is of course been addressed before in the literature, for example in the celebrated universal approximation theorem Csaji (2001).

ACKNOWLEDGEMENTS

The authors would like to thank the Isaac Newton Institute for Mathematical Sciences, Cambridge, for support and hospitality during the programmes *Variational methods and effective algorithms for imaging and vision (2017)* and *Geometry, compatibility and structure preservation in computational differential equations (2019)* Grant number EP/R014604/1 where work on this paper was undertaken.

REFERENCES

H C Andersen Rattle: A velocity version of the shake algorithm for molecular dynamics calculations. *J. Comp. Phys.* 52: 24-34 (1983).

M Benning, E Celledoni, MJ Ehrhardt, B Owren, CB Schönlieb, Deep learning as optimal control problems: models and numerical methods, *JCD* 6, (2) 171-198, (2019). doi:10.3934/jcd.2019009

B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert and E. Holtham, Reversible Architectures for Arbitrarily Deep Residual Neural Networks, arXiv: 1709.03698v1, AAAI Conference on Artificial Intelligence, 2018.

Chen, T. Q., Rubanova, Y., Bettencourt, J., Duvenaud, D. (2018). Neural Ordinary Differential Equations. *NIPS Proceedings*, 2018.

B. C. Csaji Approximation with Artificial Neural Networks; Faculty of Sciences; Etvs Lornd University, Hungary

Weinan E, A Proposal on Machine Learning via Dynamical Systems, *Comm. in Math. and Stat.* 2017.

Gholami, A., Keutzer, K., Biros, G. . ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs. *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)*(2019, February 26)

E. Haber and L. Ruthotto, Stable Architectures for Deep Neural Networks, *Inverse Problems*, 34(1): 23 pp, arXiv: 1705.03341v2. doi:10.1088/1361-6420/aa9a90

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR)*, 2014.

W. Hager. Runge-Kutta methods in optimal control and the transformed adjoint system. *Num. Math.*, 87(2):247–282, 2000.

E. Hairer, C.Lubich and G. Wanner, Geometric Numerical Integration, Springer, second edition, 2006.

He, K., Zhang, X., Ren, S., Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*

Qianxiao Li, Long Chen, Cheng Tai, Weinan E, Maximum Principle Based Algorithms for Deep Learning, *Journal of Machine Learning Research* 18, 2018.

Y. LeCun. A theoretical framework for back-propagation. In Proceedings of the 1988 connectionist models summer school, 1: 21–28. CMU, Pittsburgh, Pa: Morgan Kaufmann, 1988.

L.S. Pontryagin, Pontryagin selected works in four volumes. The mathematical theory of optimal processes: vol. 4. *Classics of Soviet mathematics*, CRC Press (1986).

L. Ruthotto and E. Haber, Deep Neural Networks Motivated by Partial Differential Equations, *Journal of Mathematical Imaging and Vision*: 1–13, (2018), arXiv:1804.04272 doi:10.1007/s10851-019-00903-1

J. M. Sanz-Serna, Symplectic Runge-Kutta schemes for adjoint equations automatic differentiation, optimal control and more, *SIAM Rev.* 58: 3–33, (2015).

Zhang, T., Yao, Z., Gholami, A., Keutzer, K., Gonzalez, J., Biros, G., Mahoney, M. ANODEV2: A Coupled Neural ODE Evolution Framework. *NIPS Proceedings*, 2019, arXiv:1906.04596 (2019, June 9).