Deep connections between learning from limited labels & physical parameter estimation - inspiration for regularization

Bas Peters
Computational Geosciences Inc.
1623 West 2nd Ave, Vancouver, BC, Canada
bas@compgeoinc.com

Abstract

Recently established equivalences between differential equations and the structure of neural networks enabled some interpretation of training of a neural network as partial-differential-equation (PDE) constrained optimization. We add to the previously established connections, explicit regularization that is particularly beneficial in the case of single large-scale examples with partial annotation. We show that explicit regularization of model parameters in PDE constrained optimization translates to regularization of the network output. Examination of the structure of the corresponding Lagrangian and backpropagation algorithm do not reveal additional computational challenges. A hyperspectral imaging example shows that minimum prior information together with cross-validation for optimal regularization parameters boosts the segmentation accuracy.

1. Introduction

Although many well-known computer vision benchmarks come with hundreds or more fully annotated images, many real-world applications come with few labels, and it is impossible/very costly to collect more labels/ground-truth.

Working with limited labels/annotation is also the default for many inverse problems. For instance, partial-differential-equation (PDE) constrained optimization for parameter estimation aims to estimate physical model parameters that predict the observations (labels). Examples include acoustic velocity estimation from observed seismic waves or conductivity from electromagnetic fields. Most inverse problems cannot be 'solved' using just the partially observed fields and a physical model that connects input and output. Prior knowledge about the underlying mathematical structure of the quantity to be estimated is typically necessary, i.e., regularization.

In this work, we focus on PDE-constrained optimization problems and regularization, describe their deep connection to learning from limited labels using neural networks, and show that there are subtle differences between the two tasks. Despite these differences, we show how to transfer regularization ideas from PDE-constrained parameter estimation to help training neural networks in the case of limited labels.

Recently, researchers established a connection between time-stepping methods for solving differential equations and deep neural networks [8, 21, 3]. This includes the recognition of the standard ResNet [10] as a time-dependent heat equation, deep neural networks based on the reaction-diffusion (advection) equation [7, 25], as well as second-order equations like the nonlinear telegraph equation [2]. These contributions concern various discretizations of the forward problem in the context of inverse problems.

Regarding optimization, [13] showed that the backpropagation algorithm is equivalent to an implementation of the method of Lagrangian multipliers for equality constrained nonlinear optimization. Stating the training of a network as constrained optimization opens the door to methods other than backpropagation: for instance, [22, 6] increase parallelism and [5, 15, 20] exploit the connection to PDE-constrained optimization to reduce memory. In this work, we

- show that regularization on the model parameters in PDE-constrained optimization translates to regularization of the output of a neural network;
- show there are no additional computational complications related to network-output regularization via a derivation of the Lagrangian and backpropagation algorithm.
- illustrate that network-output regularization based on minimal prior knowledge can boost the segmentation accuracy when training on few labels.

First, we establish the correspondence between the terminology in deep learning and PDE-constrained optimization. Next, we state the problems and derive an optimization

algorithm for both regularized PDE-constrained optimization and training deep neural networks. An example shows that basic assumptions about the output of a network for a given application lead to a simple, yet effective method to improve semantic segmentation results in the label-sparse training regime.

2. Terminology differences

In PDE-constrained optimization, the *model* is a physical differential equation, such as the heat, or Maxwell equations. The *model parameters* are the coefficients of the equation, i.e., the spatial distribution of acoustic velocity or electrical conductivity. The input to the model are the initial conditions, or source functions. The output of the physical system is compared to the observed *data*. When training a neural network, we commonly refer to the input of a network as the *data*. The output (prediction) of the network is compared to the *labels*. The *model* is the structure of the neural network; examples of *model parameters* are biasses and convolutional kernels. The goal of image segmentation is to estimate model parameters that transform the input data (images) into the labels (segmentation).

3. Output-regularized neural network training as PDE-constrained optimization

Rather than standard tensor notation, we employ matrix-vector product descriptions to stay close to PDE-constrained optimization literature. We block-vectorize states and Lagrangian multipliers, while weight/parameter tensors are flattened into block-matrices, see [21, 18, 4]. To keep notation compact, we focus the ResNet [10] with time-step h. The network state $\mathbf{y}_j$ at layer j is given by

$$\mathbf{y}_j = \mathbf{y}_{j-1} - hf(\mathbf{K}_j\mathbf{y}_{j-1}). \tag{3.1}$$

The block-matrix $\mathbf{K}$ denotes the network parameters where the number of block-rows is equal to the number of output channels, the number of block-columns is equal to the number of input channels. The nonlinear pointwise f is the activation function. Given data $\mathbf{d}$ and labels $\mathbf{c}$, we propose to minimize a loss function $l(\mathbf{d}, \{\mathbf{K}\}, \mathbf{c})$, subject to one equality constraint per time-step, i.e.,

$$\begin{aligned}\min_{\{\mathbf{K}\}}\ & l(\mathbf{Q}\mathbf{y}_n, \mathbf{c}) + \alpha R(\mathbf{y}_n) \ \text{ s.t.} \\ & \mathbf{y}_n = \mathbf{y}_{n-1} - f(\mathbf{K}_n\mathbf{y}_{n-1}) \\ & \vdots \\ & \mathbf{y}_j = \mathbf{y}_{j-1} - f(\mathbf{K}_j\mathbf{y}_{j-1}) \\ & \vdots \\ & \mathbf{y}_1 = \mathbf{d}.\end{aligned} \tag{3.2}$$

There are n time-steps (network layers). The matrix $\mathbf{Q}$ selects from the prediction, the pixel indices where we have labels. This is analogous to restricting physical fields to sensor locations [17, 19]. Compared to earlier regularized PDE-constrained optimization formulations [8], we propose to apply $\alpha R(\mathbf{y}_n)$ to the network output (and not on the parameters $\mathbf{K}$ as in weight-decay, or parameter smoothness in time [8]), such that we can explicitly control the properties of the predicted probability maps. This is similar in spirit to [1], see also [12] for context regarding other types of implicit/explicit regularization. While we could employ automatic differentiation to the above problem, we need to look at the Lagrangian to see what are the effects of output regularization on the subsequent optimization procedures:

$$\begin{aligned}L(\{\mathbf{y}\},\{\mathbf{p}\},\{\mathbf{K}\}) &= l(\mathbf{Q}\mathbf{y}_n, \mathbf{d}) + \alpha R(\mathbf{y}_n) \\ &- \sum_{j=2}^{n} \mathbf{p}_j^\top(\mathbf{y}_j - \mathbf{y}_{j-1} + f(\mathbf{K}_j\mathbf{y}_{j-1})) - \mathbf{p}_1^\top(\mathbf{y}_1 - \mathbf{d}),\end{aligned} \tag{3.3}$$

where $\mathbf{p}_j$ are the Lagrangian multipliers. To construct an algorithm for problem (3.2), we need the partial derivatives of L w.r.t. the variables,

$$\nabla_{\mathbf{y}_n} L = \nabla_{\mathbf{y}_n} l(\mathbf{Q}\mathbf{y}_n, \mathbf{d}) + \alpha\nabla_{\mathbf{y}_n} R(\mathbf{y}_n) - \mathbf{p}_n \tag{3.4}$$

$$\text{for } j = n-1, \cdots, 2:$$

$$\nabla_{\mathbf{y}_{j-1}} L = \mathbf{p}_j - \mathbf{K}_{j-1}^\top \operatorname{diag}(f'(\mathbf{K}_{j-1}\mathbf{y}_{j-1}))\mathbf{p}_j - \mathbf{p}_{j-1} \tag{3.5}$$

$$\nabla_{\mathbf{p}_j} L = -\mathbf{y}_j + \mathbf{y}_{j-1} - f(\mathbf{K}_j\mathbf{y}_{j-1}) \tag{3.6}$$

$$\nabla_{\mathbf{K}_j} L = \left[\frac{\partial\big(\mathbf{K}_j\mathbf{y}_{j-1}\big)}{\partial\mathbf{K}_j}\right]^\top \operatorname{diag}(f'(\mathbf{K}_j\mathbf{y}_{j-1}))\mathbf{p}_j \tag{3.7}$$

where we left out the derivatives related to the first layer as they are the initial condition that we set equal to the input data while training. f' denotes the derivative of f. The above shows that the derivatives of the regularization term appear in the partial derivative of the Lagrangrian with respect to the last state ($\mathbf{y}_n$) only. To compute a gradient, we use the 'standard' tools: adjoint-state/backpropagation, see [13, 5] for details about their equivalence. Note that to satisfy the first-order optimality conditions for (3.2), each of the partial derivatives in (3.4) needs to vanish: 1) Forward propagate through the network to satisfy all constraints in (3.2); $\nabla_{\mathbf{p}_j} L$ vanishes. 2) Propagate backward to obtain all Lagrangian multipliers $\mathbf{p}_j$. 3) compute gradients w.r.t. the network parameters $\mathbf{K}_j$ for every layer. In Algorithm 1 we show a slightly different version that computes the gradient w.r.t. parameters on the fly. This procedure avoids storing more Lagrangian multipliers than the length of the recursion of the differential equation discretization, instead of storing the multipliers for all layers. Note that it is possible to avoid storing all states $\mathbf{y}_j$ via reversible networks that re-compute the states on the fly during backpropagation [15].

Algorithm 1: adjoint-state/backpropagation to compute the gradient of a network

$\mathbf{y}_1 = \mathbf{d}$ //Initialization ;
for $i = 2, \cdots, n$ **do**
  $\mathbf{y}_j = \mathbf{y}_{j-1} - f(\mathbf{K}_j \mathbf{y}_{j-1})$ // Forward;
end
Compute final Lagrangian multiplier;
$\mathbf{p}_n = \nabla_{\mathbf{y}_n} l(\mathbf{Q}\mathbf{y}_n, \mathbf{d}) + \alpha \nabla_{\mathbf{y}_n} R(\mathbf{y}_n)$;
//Propagate backward;
for $i = n, n-1, \cdots, 2$ **do**
  $\nabla_{\mathbf{K}_j} L = \left[\frac{\partial\left(\mathbf{K}_j \mathbf{y}_{j-1}\right)}{\partial \mathbf{K}_j}\right]^\top \operatorname{diag}(f'(\mathbf{K}_j \mathbf{y}_{j-1}))\mathbf{p}_j$;
  $\mathbf{p}_{j-1} = \mathbf{p}_j - \mathbf{K}_{j-1}^\top \operatorname{diag}(f'(\mathbf{K}_{j-1}\mathbf{y}_{j-1}))\mathbf{p}_j$;
end

4. Examples of simple explicit regularizers

So far we showed that PDE-constrained optimization and training neural-networks are similar processes. However, the goals are different. PDE-constrained optimization often estimates material properties (parameters) with the help of an additive scaled regularization function $R(\mathbf{K})$. In contrast, when training networks, we primarily care about the prediction (network output). Despite these different objectives, a successful technique to prevent overfitting is regularization of the network parameters, just as in the PDE-constrained optimization case. This is a form of implicit regularization, as it is not trivial to see how the regularization relates to the visual appearance of the output for applications like semantic segmentation and other image-to-image applications. In the PDE-constrained optimization setting, regularizing the parameters is a form of explicit regularization because the regularization directly acts on the quantity of interest.

The second contribution of this work is to recognize that we can carry over the explicit regularization nature from PDE-constrained optimization to training neural networks by adding a regularization (penalty) term on the network output (final state $\mathbf{y}_n$).

Consider semantic segmentation in the case of limited data and even more limited supervision. Specifically, consider applications with a single (possibly large-scale) example with partial labeling or point annotations [17] where it is difficult or impossible to collect additional examples. In the next section, we present results for such an application: time-lapse hyperspectral imaging using partial labeling. Insufficient annotation leads to high-frequency/oscillatory artifacts in the final network state, or, probability map, $\mathbf{y}_n$. Using the prior knowledge that the true thresholded prediction is piecewise-constant and there are not many isolated pixels or line segments with a class different from their surroundings, a reasonable choice for regularization is a quadratic smoother

$$R(\mathbf{y}_n) = \frac{1}{2}\|\nabla_1 \mathbf{y}_n\|_2^2 + \frac{1}{2}\|\nabla_2 \mathbf{y}_n\|_2^2 \tag{4.8}$$

with gradient

$$\nabla_{\mathbf{y}_n} R(\mathbf{y}_n) = \nabla_1^\top \nabla_1 \mathbf{y}_n + \nabla_2^\top \nabla_2 \mathbf{y}_n. \tag{4.9}$$

The discrete gradients ∇_1 and ∇_2 are the derivatives of an image in the first or second coordinate, respectively. This regularizer adds to the final Lagrangian multiplier. Subsequently, this information backpropagates and influences the gradient w.r.t. the network parameters $\nabla_{\mathbf{K}} L$. In the above example, the quadratic smoothing makes sure the final output state transitions across class boundaries smoothly.

5. Choice of α and numerical example

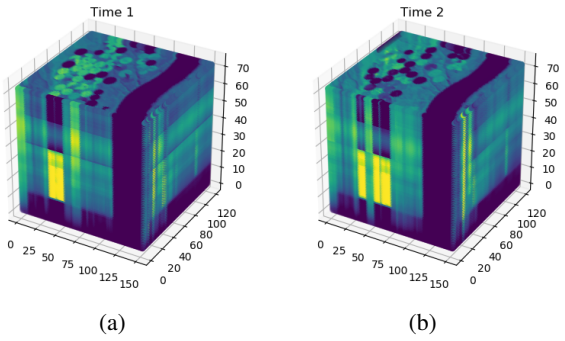
This example shows that 1) while the overall optimization problem is still non-convex, cross-validation to select the regularization parameter α results in expected behavior; 2) the proposed explicit regularization can improve predictions. The task is land-use change detection from time-lapse (4D) hyperspectral data. The input data [9] are two 3D hyperspectral datasets collected over the same location, with two spatial and one frequency axis, see Figure 1.

The goal is to output a 2D map of the earth's surface that shows land-use change (Figure 1). For training, there are 200 randomly selected annotated pixels; and 50 pixels for validation. The available labels were annotated by an expert. The default mode of operation for many hyperspectral tasks is interpolation/extrapolation of the labels to the full map, see, e.g., [14, 11, 24, 16, 23]. Generalization to new locations is not a concern. The difficulty of annotation causes limited availability of annotated hyperspectral data; the person that creates the labels needs to know the application and nature of hyperspectral data. Alternatively, the labels come from costly ground truth observations.

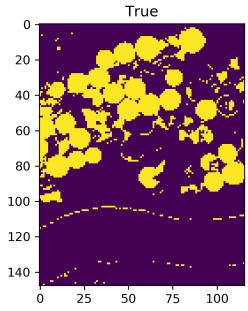
We train a fully reversible network [15] with 10 layers with up to 32 channels for 250 stochastic gradient descent iterations with a decaying learning rate. This is sufficiently many iterations, such that the validation loss is no longer decreasing. Training includes basic data-augmentation via random flips and rotations. We repeat training for a range of α values and assess the results using the mean of the intersection over union per class (mIoU, Figure 2). Figure 3 shows results and errors using no regularization, optimal regularization parameter, and a very large α. While the best result is not perfect, the simple regularization on the network output yields a significant increase in mIoU.

6. Conclusions

We extended the deep connections between partial-differential-equation (PDE) constrained optimization and



(a) (b)



(c)

Figure 1: (a) and (b): Two time-instances of hyperspectral data collected over the same location. (c) true land-use change (a 2D map of the surface of the Earth).

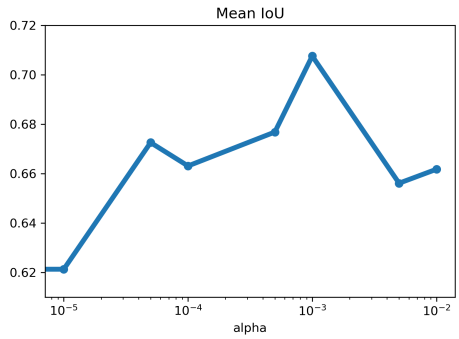


Figure 2: Mean intersection over union of the prediction as a function of the explicit output regularization strength.

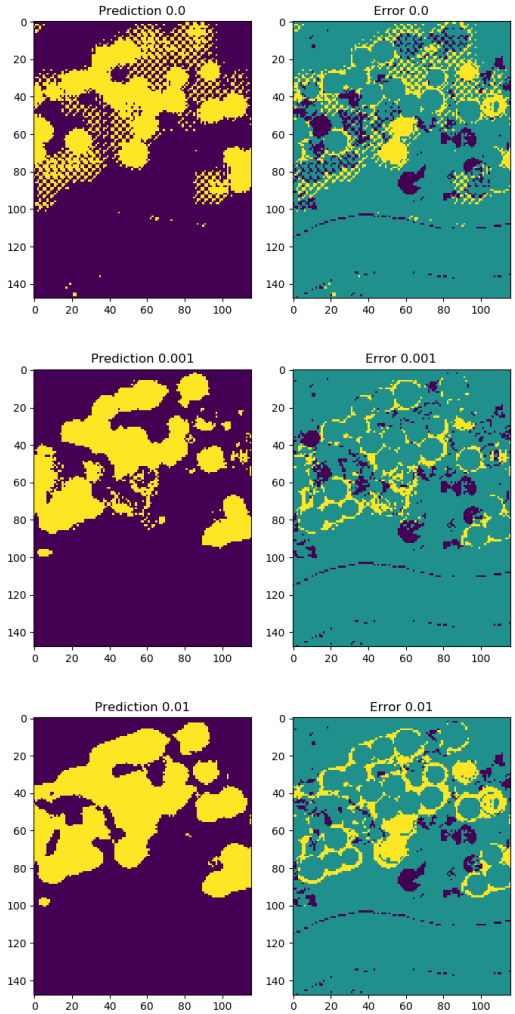


Figure 3: Top: prediction without regularization leads to oscillatory artifacts in the prediction. Middle: results for optimal regularization parameter α in terms of cross-validation for mIoU (Figure 2). Bottom: results for very large α

training neural networks on few data with partial annotation. An insufficient number of annotated pixels leads to oscillatory artifacts in image segmentations. In PDE-constrained optimization, regularization on the model parameters mitigates problems related to insufficient labeling (observations). This carries over directly to the neural network setting as, for example, weight decay. While weight decay regularizes the parameters directly, the final prediction of interest relates indirectly. We showed that explicit regularization on the output of a network serves the same purpose as parameter regularization in PDE-constrained optimization. Brief derivation of the optimization problem and algorithm shows this does not cause computational complications. A hyperspectral example illustrates that simple cross-validation for selecting the regularization strength can improve prediction quality while making minimal assumptions about prior knowledge. Future work should explore more sophisticated regularizers and methods to adapt the regularization strength while training.

References

[1] Aïcha BenTaieb and Ghassan Hamarneh. Topology aware fully convolutional networks for histology gland segmentation. In Sebastien Ourselin, Leo Joskowicz, Mert R. Sabuncu, Gozde Unal, and William Wells, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 460–468, Cham, 2016. Springer International Publishing. 2

[2] Bo Chang, Lili Meng, Eldad Haber, Lars Ruthotto, David Begert, and Elliot Holtham. Reversible architectures for arbitrarily deep residual neural networks. In *AAAI Conference on AI*, 2018. 1

[3] Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018. 1

[4] J. Ephrath, M. Eliasof, L. Ruthotto, E. Haber, and E. Treister. Leanconvnets: Low-cost yet effective convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing*, pages 1–1, 2020. 2

[5] Amir Gholaminejad, Kurt Keutzer, and George Biros. Anode: Unconditionally accurate memory-efficient gradients for neural odes. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 730–736. International Joint Conferences on Artificial Intelligence Organization, 7 2019. 1, 2

[6] Stefanie Günther, Lars Ruthotto, Jacob B. Schroder, Eric C. Cyr, and Nicolas R. Gauger. Layer-parallel training of deep residual neural networks. *SIAM Journal on Mathematics of Data Science*, 2(1):1–23, 2020. 1

[7] Eldad Haber, Keegan Lensink, Eran Treister, and Lars Ruthotto. IMEXnet a forward stable deep neural network. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2525–2534, Long Beach, California, USA, 09–15 Jun 2019. PMLR. 1

[8] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, dec 2017. 1, 2

[9] Mahdi Hasanlou and Seyd Teymoor Seydi. Hyperspectral change detection: an experimental comparative study. *International Journal of Remote Sensing*, 39(20):7029–7083, 2018. 3

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 2

[11] M. He, B. Li, and H. Chen. Multi-scale 3d deep convolutional neural network for hyperspectral image classification. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3904–3908, Sep. 2017. 3

[12] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017. 2

[13] Yann Lecun. A theoretical framework for back-propagation. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA*, pages 21–28. Morgan Kaufmann, 1988. 1, 2

[14] H. Lee and H. Kwon. Contextual deep cnn based hyperspectral classification. In *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3322–3325, July 2016. 3

[15] Keegan Lensink, Eldad Haber, and Bas Peters. Fully hyperbolic convolutional neural networks. *arXiv preprint arXiv:1905.10484*, 2019. 1, 2, 3

[16] Ying Li, Haokui Zhang, and Qiang Shen. Spectralspatial classification of hyperspectral imagery with 3d convolutional neural network. *Remote Sensing*, 9(1), 2017. 3

[17] Bas Peters, Justin Granek, and Eldad Haber. Multiresolution neural networks for tracking seismic horizons from few training images. *Interpretation*, 7(3):SE201–SE213, 2019. 2, 3

[18] Bas Peters, Eldad Haber, and Justin Granek. Does shallow geological knowledge help neural-networks to predict deep units? In *SEG Technical Program Expanded Abstracts 2019*, pages 2268–2272, 2019. 2

[19] Bas Peters, Eldad Haber, and Justin Granek. Neural networks for geophysicists and their application to seismic data interpretation. *The Leading Edge*, 38(7):534–540, 2019. 2

[20] Bas Peters, Eldad Haber, and Keegan Lensink. Symmetric block-low-rank layers for fully reversible multilevel neural networks. *arXiv preprint arXiv:1912.12137*, 2019. 1

[21] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, pages 1–13, 2018. 1, 2

[22] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. Training neural networks without gradients: A scalable admm approach. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2722–2731, New York, New York, USA, 20–22 Jun 2016. PMLR. 1

[23] Qin Xu, Yong Xiao, Dongyue Wang, and Bin Luo. Csa-mso3dcnn: Multiscale octave 3d cnn with channel and spatial attention for hyperspectral image classification. *Remote Sensing*, 12(1), 2020. 3

[24] Zhixiang Xue. A general generative adversarial capsule network for hyperspectral image spectral-spatial classification. *Remote Sensing Letters*, 11(1):19–28, 2020. 3

[25] Tianjun Zhang, Zhewei Yao, Amir Gholami, Kurt Keutzer, Joseph Gonzalez, George Biros, and Michael Mahoney. Anodev2: A coupled neural ode evolution framework. *arXiv preprint arXiv:1906.04596*, 2019. 1