Structure preserving neural networks based on ODEs

Anonymous Author(s)
Affiliation
Address
email

Neural networks have gained popularity due to their ability to provide accurate solutions in various applications. Generally, the problems in which they are effective can be phrased as approximation tasks in high-dimensional spaces. This success is often based on experimental evidence, and more theoretical insight is needed. Recently, many authors have considered the design of deep learning architectures that satisfy certain mathematical properties such as stability, symplecticity, volume preservation, or constraints on the Lipschitz constant, [16], [13], [8], [27], [7], [9], [29], [14], [31], [33]. The networks presented in these articles are often quite specific since they have been derived ad hoc, typically ensuring only one particular property.

There are multiple situations where one could be interested in networks with some prescribed structure. Referring to $F : \mathcal{X} \to \mathcal{Y}$ as the function approximated by the network, we report here three cases:

1. when F has some known property that is important for the application of interest,
2. when the geometry of $\mathcal{X}$ and $\mathcal{Y}$ prescribes a particular structure for F,
3. when we can approximate F sufficiently accurately with functions in $\mathcal{G}$, a family of functions with a property that makes them well suited to design the network layers.

This manuscript describes a general and systematic way to impose desired mathematical structures on neural networks. We remark that the interest in preserving the specific structure of a function can be found in the field of geometric numerical integration, [12], [17], [21], [15]. For this reason, the approach we introduce is based on defining the network layers as a (structure-preserving) discretisation of the solutions of a specific differential equation. To illustrate the methodology, we present the derivation of three structured neural networks. Each of them represents one of the three situations reported above.

There have been multiple attempts to formulate unifying principles for the design of neural networks. We mention in particular the continuous-in-time interpretation of residual neural networks (see, e.g., [32], [19], [5], [26], [2]) on which our work builds upon. Residual neural networks (ResNets) are compositions of parametric maps that are either linear or of the form

$$\mathbb{R}^{n_i} \ni x \mapsto f_{\theta_i}(x) = x + h\Lambda(\theta_i, x) \in \mathbb{R}^{n_{i+1}}. \tag{1}$$

A typical expression for Λ is $\Lambda(\theta_i, x) = \Sigma(A_i x + b_i)$ with $\Sigma(z) = [\sigma(z_1), ..., \sigma(z_n)]$ and $\sigma : \mathbb{R} \to \mathbb{R}$. The continuous-in-time interpretation of ResNets arises noticing that if $n_i = n_{i+1}$, f_{θ_i} coincides with one $h-$step of the explicit Euler method applied to $\dot{x}(t) = \Lambda(\theta(t), x(t))$. In this work, we consider time-switching systems of the form

$$\dot{x}(t) = \Lambda(\theta(t), x(t)) = f_{s(t)}(x(t)),\ s : [0, T] \to \{1, ..., N\},\ f_i \in \mathcal{F}, \tag{2}$$

with s being piecewise constant (see, e.g., [25], [18]), and $\mathcal{F}$ a family of parametric vector fields. In other words, we consider vector fields with weight functions that are piecewise constant in time.

This interpretation of ResNets gives the foundation of the approach we present. Indeed, we propose to modify two steps in the continuous-in-time interpretation of ResNets. We focus on properties of

Submitted to the DLDE Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

functions preserved under composition, such as being $1-$Lipschitz or volume-preserving. Let us call $\mathcal{P}$ the property of interest. Then our strategy to define a neural network satisfying $\mathcal{P}$ consists in

1. finding a family $\mathcal{F}$ of parametric vector fields having flow maps that satisfy $\mathcal{P}$,
2. replacing the explicit Euler method in (1) with suitable numerical methods that reproduce $\mathcal{P}$ also at the discrete level.

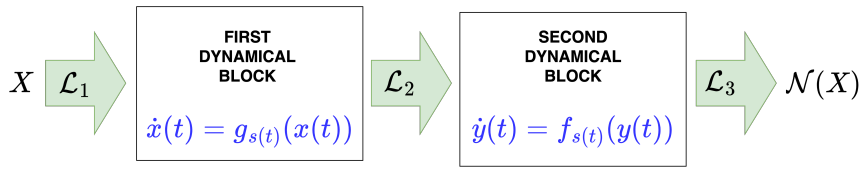

Figure 1: ResNet with two dynamical blocks and three linear layers. $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3$ are linear maps.

As with classical ResNets, we also introduce linear layers in the network, see Figure 1. For this reason, we introduce the name "dynamical blocks" to refer only to the groups of layers obtained from an ODE. We remark that one can still get networks with the desired structure by constraining the linear layers to preserve the desired property $\mathcal{P}$.

The key reason that motivates our approach is that it is often easier to impose a property on a vector field than on its flow map. The additional advantage of this strategy is that the research areas of dynamical systems and numerical analysis are well developed. One can thus exploit this knowledge to design neural networks that reproduce the desired structure. Moreover, this approach enables us to derive new structured networks systematically and collocate other existing architectures into a more general setting, making their analysis easier.

1 Volume-preserving neural networks

We start with the first case of interest. We focus, in particular, on volume-preserving functions. For this reason, we do not allow for dimensionality changes throughout the network. In this case, defining the approximating network based on switching systems and splitting methods can be a flexible strategy. Consider the switching system defined by $\dot{z}(t) = f_{s(t)}(z(t)) \in \mathbb{R}^n$ where for every value $s(t) = i$, the vector field f_i has a specific partitioning that makes it divergence-free. For example, if we have $n = 2m$,

$$f_{s(t)}(z) = \begin{bmatrix} u_{s(t)}(z[m:]) \\ v_{s(t)}(z[:m]) \end{bmatrix}, \quad u_i, v_j : \mathbb{R}^m \to \mathbb{R}^m$$

satisfies such a condition, and its flow map will be volume-preserving. We can numerically integrate such a vector field while preserving this property. Indeed, we can compose the exact flow maps of the two volume-preserving vector fields

$$f^1_{s(t)}(z) = \begin{bmatrix} u_{s(t)}(z[m:]) \\ 0 \end{bmatrix}, \quad f^2_{s(t)}(z) = \begin{bmatrix} 0 \\ v_{s(t)}(z[:m]) \end{bmatrix}, \quad f_{s(t)} = f^1_{s(t)} + f^2_{s(t)},$$

i.e. we can apply a splitting method to approximate the solutions (see [21]). This approach gives architectures that are close to the ones of RevNets (see, e.g., [9]). A particular class of these blocks can be obtained with second-order ODEs as $\ddot{x}(t) = f_{s(t)}(x(t))$ and, in particular, with second-order conservative ODEs like $\ddot{x}(t) = \nabla V_{s(t)}(x(t))$.

2 Mass-preserving neural networks

Some function approximation problems deal with data having a particular structure. As an example, we now focus on problems where the target function $F : \mathbb{R}^n \to \mathbb{R}^m$ is known to satisfy $T_n x := \sum_{i=1}^n x_i = T_m F(x) := \sum_{i=1}^m F(x)_i$. We refer to functions of this type as mass preserving functions (see [3]). The neural network approximating F should also be mass preserving to generate interpretable outputs.

We interpret the network layers as discrete flow maps of specific vector fields. A vector field $X \in \mathfrak{X}(\mathbb{R}^n)$ whose flow map preserves the sum of the components of the state vector is one having a linear first integral $g(x) = \mathbf{1}^T x = \sum_{i=1}^n x_i, \quad \mathbf{1} = [1, ..., 1]^T \in \mathbb{R}^n$. Any vector field X having g as a conserved quantity can be written in the form $X(x) = S(x)\mathbf{1}$ for a skew-symmetric matrix-valued function $S : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ (see [22]). Thus, we can model the network as a suitable time discretisation of an ODE of the form

$$\dot{y}(t) = (\hat{A}(y) - \hat{A}(y)^T)\mathbf{1}, \;\; \hat{A} : \mathbb{R}^n \to \mathbb{R}^{n\times n}. \tag{3}$$

The matrix function $\hat{A}$ is upper triangular, and to model it we reshape it into the parametric vector-function $A(x) = W^T \Sigma(Vx + v) \in \mathbb{R}^{n(n-1)/2}$. Furthermore, as presented in [11, Chapter 4], every Runge–Kutta method preserves linear first integrals. Thus, one can model mass preserving neural networks by composing layers of the following types:

1. Lifting layers: $L : \mathbb{R}^k \to \mathbb{R}^{k+s}$, $L(x_1, ..., x_k) = (x_1, ..., x_k, 0, ..., 0)$,
2. Projection layers: $P : \mathbb{R}^{k+s} \to \mathbb{R}^k$, $P(x_1, ..., x_k, x_{k+1}, ..., x_{k+s}) = (x_1 + o, ..., x_k + o)$ with $o = \sum_{i=1}^s x_{k+i}/s$,
3. Dynamical blocks: one-step explicit Euler discretisations of (3).

To test the neural network architecture, we approximate the flow map of the ODE

$$\dot{y} = \begin{bmatrix} 0 & -y_3 y_1^2 & y_2 y_3 \\ y_3 y_1^2 & 0 & -\sin y_1 \\ -y_2 y_3 & \sin y_1 & 0 \end{bmatrix} \mathbf{1} = X(y). \tag{4}$$

We approximate the time$-\Delta t$ flow map of such a system, with $\Delta t = 1/40$. The network is trained in a recurrent manner (see, e.g., [7], [6]). As training data we use tuples of the form $\{(x_i = y_i^0, y_i^1, ..., y_i^M)\}_{i=1,...,N}$, where $y_i^{j+1} = \Psi^{\Delta t}(y_i^j)$ is an accurate approximation of the exact time-Δt flow of X. Figure 2 compares the results obtained with an accurate approximation of the true solution of (4) to those coming from the mass-preserving neural network.

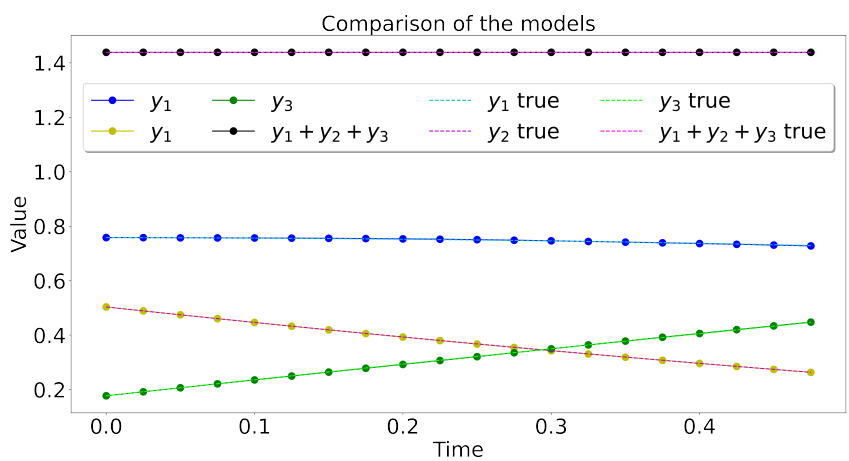

Figure 2: One test trajectory up to time $T = 0.5$. We report the sum of the components, which is also preserved by the network. The components of the vector (y_1, y_2, y_3) are the network predictions.

3 Lipschitz constrained neural networks

We now focus on the problem of classifying points of a set $\mathcal{X} \subset \mathbb{R}^n$ into its C classes. This problem consists hence in approximating the function $\ell : \mathcal{X} \to \{1, ..., C\}$ that assigns all the points of $\mathcal{X}$ to their correct classes. Neural networks can provide accurate solutions to this problem, but they can be very sensitive to suitable input perturbations called adversarial attacks (see [28], [20], [4], [10]). In

[30], the authors characterise this sensitivity problem mathematically. Calling $\mathcal{N}$ the approximating network and $\mathcal{M}_{\mathcal{N}}(x) = \mathcal{N}(x)^T e_{\ell(x)} - \max_{j \neq \ell(x)} \mathcal{N}(x)^T e_j$ its margin at $x \in \mathcal{X}$, they show that $\mathcal{N}$ classifies as x the points in $B_\varepsilon(x) = \{y \in \mathbb{R}^n : \|y - x\|_2 < \varepsilon\}$ if $\mathcal{M}_{\mathcal{N}}(x) \geq \sqrt{2}\varepsilon \mathrm{Lip}(\mathcal{N})$[1]. Motivated by this result, we present a strategy to constrain the Lipschitz constant of ResNets to the value of 1. To show the flexibility of the proposed framework and obtain expressive networks, we impose such property without relying only on $1-$Lipschitz layers, differently from [5], [33] and [23].

Consider the two families of vector fields

$$\mathcal{F}_c = \{-A^T \Sigma(Ax + b) \in \mathbb{R}^n : A^T A = I_n\}, \quad \sigma(s) = \max\{x, x/2\}, \quad \text{and}$$
$$\mathcal{F} = \{X \in \mathfrak{X}(\mathbb{R}^n) : \mathrm{Lip}(X) \leq 1\}.$$

Here we denote with $I_n \in \mathbb{R}^{n \times n}$ the identity matrix. We remark that $-A^T \Sigma(Ax + b) = -\nabla_x V(x, A, b)$ for $V(x, A, b) = \mathbf{1}^T \Gamma(Ax + b)$ that is strongly-convex in the x variable. Here we denote with $\mathbf{1} = [1, ..., 1] \in \mathbb{R}^n$, and $\Gamma(z) = [\gamma(z_1), ..., \gamma(z_n)]$ with $\gamma' = \sigma$. The strong-convexity of V is fundamental in our derivation.

With suitable step restrictions, the composition of the Euler steps $\Psi_1^{h_1}(x) = x - A^T \Sigma(Ax + b)$ and $\Psi_2^{h_2}(x) = x + h_2 X(x)$, $X \in \mathcal{F}$, can be made 1-Lipschitz. Indeed, by direct computation, we get

$$\mathrm{Lip}(\Psi_1^{h_1} \circ \Psi_2^{h_2}) \leq \mathrm{Lip}(\Psi_1^{h_1}) \cdot \mathrm{Lip}(\Psi_2^{h_2}) \leq \sqrt{1 - 2h_1 + h_1^2}(1 + h_2).$$

Thus, the composition $\Psi_2^{h_2} \circ \Psi_1^{h_1}$ is $1-$Lipschitz if the step sizes are properly chosen. Repeating a similar reasoning for $\Psi_2^{h_2/2} \circ \Psi_1^{h_1/2} \circ \Psi_2^{h_2/2} \circ \Psi_1^{h_1/2}$, one can obtain weaker restrictions on the step sizes. This is the solution we adopt for the following experiment.

We now apply this reasoning to design a neural network with increased robustness to L^2-PGD adversarial attacks on the CIFAR-10 dataset. Instead of considering the generic $\mathcal{F}$, we work with the subfamily $\mathcal{F}_e = \{\Sigma(Bx + c) : B \in \mathbb{R}^{n \times n}, \|B\|_2 \leq 1, c \in \mathbb{R}^n\}$. The step sizes are constrained to satisfy the $1-$Lipschitz constraint derived above. Weight orthogonality comes from the regularisers proposed in [31]. The L^2-PGD adversarial attacks are generated with Foolbox [24]. We compare the results with a baseline ResNet characterised by updates of the form $x \mapsto x + B\Sigma(Ax + b)$, and hence approximately double the number of weights. Table 1 highlights that for Lipschitz constrained ResNets, the accuracy obtained for highly perturbed images is considerably better than the one with the baseline ResNet. The lower accuracy for clean images is due to the constraints we impose on the network and the reduced number of weights, making it harder to train. On the other hand, other alternation strategies and constraining techniques might lead to better results on this side.

Type	$\varepsilon = 0.0$	$\varepsilon = 0.06$	$\varepsilon = 0.14$	$\varepsilon = 0.3$	$\varepsilon = 0.5$	$\varepsilon = 1.0$
Lipschitz, Margin = 0.07	0.808	0.790	0.775	0.739	0.678	0.516
Lipschitz, Margin = 0.15	0.796	0.782	0.761	0.718	0.657	0.505
Baseline, Margin = 0.07	0.896	0.870	0.816	0.670	0.461	0.114
Baseline, Margin = 0.15	0.903	0.856	0.790	0.642	0.424	0.073

Table 1: We denote with ε the magnitude of the adversarial perturbations. With 'Lipschitz' we refer to Lipschitz constrained networks. All the networks have been trained with hinge loss function [1] for different margin values.

4 Conclusion

In this work, we have introduced a framework to combine the design of ODE models with the choice of proper numerical methods to obtain neural networks with prescribed properties. We illustrate the procedure deriving three families of structured networks. Some numerical experiments also support their practicality.

[1] Here we denote with $\|\cdot\|_2$ the Euclidean norm of $\mathbb{R}^n$, with $e_i = [0, ..., 0, 1, 0, ..., 0] \in \mathbb{R}^n$ the $i - th$ canonical basis vector and $\mathrm{Lip}(\mathcal{N})$ is the Lipschitz constant of $\mathcal{N}$.

References

[1] *Hinge Loss Pytorch*. `https://pytorch.org/docs/stable/generated/torch.nn.MultiMarginLoss.html`.

[2] A. AGRACHEV AND A. SARYCHEV, *Control on the manifolds of mappings with a view to the deep learning*, Journal of Dynamical and Control Systems, (2021), pp. 1–20.

[3] S. BLANES, A. ISERLES, AND S. MACNAMARA, *Positivity-preserving methods for population models*, arXiv preprint arXiv:2102.08242, (2021).

[4] N. CARLINI AND D. WAGNER, *Towards evaluating the robustness of neural networks*, in 2017 IEEE Symposium on Security and Privacy, IEEE, 2017, pp. 39–57.

[5] E. CELLEDONI, M. J. EHRHARDT, C. ETMANN, R. I. MCLACHLAN, B. OWREN, C.-B. SCHÖNLIEB, AND F. SHERRY, *Structure-preserving deep learning*, European Journal of Applied Mathematics, 32 (2021), pp. 888–936.

[6] E. CELLEDONI, A. LEONE, D. MURARI, AND B. OWREN, *Learning Hamiltonians of constrained mechanical systems*, Journal of Computational and Applied Mathematics, 417 (2023), p. 114608.

[7] Z. CHEN, J. ZHANG, M. ARJOVSKY, AND L. BOTTOU, *Symplectic Recurrent Neural Networks*, in International Conference on Learning Representations, 2020.

[8] C. L. GALIMBERTI, L. FURIERI, L. XU, AND G. FERRARI-TRECATE, *Hamiltonian deep neural networks guaranteeing non-vanishing gradients by design*, arXiv preprint arXiv:2105.13205, (2021).

[9] A. N. GOMEZ, M. REN, R. URTASUN, AND R. B. GROSSE, *The reversible residual network: Backpropagation without storing activations*, Advances in Neural Information Processing Systems, 30 (2017).

[10] I. J. GOODFELLOW, J. SHLENS, AND C. SZEGEDY, *Explaining and harnessing adversarial examples*, in 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, Y. Bengio and Y. LeCun, eds., 2015.

[11] E. HAIER, C. LUBICH, AND G. WANNER, *Geometric Numerical integration: structure-preserving algorithms for ordinary differential equations*, Springer, 2006.

[12] E. HAIRER, M. HOCHBRUCK, A. ISERLES, AND C. LUBICH, *Geometric numerical integration*, Oberwolfach Reports, 3 (2006), pp. 805–882.

[13] J. HERTRICH, S. NEUMAYER, AND G. STEIDL, *Convolutional proximal neural networks and plug-and-play algorithms*, Linear Algebra and its Applications, 631 (2021), pp. 203–234.

[14] P. JIN, Z. ZHANG, A. ZHU, Y. TANG, AND G. E. KARNIADAKIS, *SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems*, Neural Networks, 132 (2020), pp. 166–179.

[15] F. KANG AND S. ZAI-JIU, *Volume-preserving algorithms for source-free dynamical systems*, Numerische Mathematik, 71 (1995), pp. 451–463.

[16] Y. LECUN, B. BOSER, J. S. DENKER, D. HENDERSON, R. E. HOWARD, W. HUBBARD, AND L. D. JACKEL, *Backpropagation applied to handwritten zip code recognition*, Neural computation, 1 (1989), pp. 541–551.

[17] B. LEIMKUHLER AND S. REICH, *Simulating Hamiltonian dynamics*, no. 14, Cambridge University Press, 2004.

[18] D. LIBERZON, *Switching in systems and control*, vol. 190, Springer, 2003.

[19] Y. LU, A. ZHONG, Q. LI, AND B. DONG, *Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations*, in International Conference on Machine Learning, PMLR, 2018, pp. 3276–3285.

[20] A. MADRY, A. MAKELOV, L. SCHMIDT, D. TSIPRAS, AND A. VLADU, *Towards Deep Learning Models Resistant to Adversarial Attacks*, in International Conference on Learning Representations, 2018.

[21] R. I. MCLACHLAN AND G. R. W. QUISPEL, *Splitting methods*, Acta Numerica, 11 (2002), pp. 341–434.

[22] R. I. MCLACHLAN, G. R. W. QUISPEL, AND N. ROBIDOUX, *Geometric integration using discrete gradients*, Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 357 (1999), pp. 1021–1045.

[23] L. MEUNIER, B. DELATTRE, A. ARAUJO, AND A. ALLAUZEN, *Scalable Lipschitz Residual Networks with Convex Potential Flows*, arXiv preprint arXiv:2110.12690, (2021).

[24] J. RAUBER, W. BRENDEL, AND M. BETHGE, *Foolbox: A Python toolbox to benchmark the robustness of machine learning models*, arXiv preprint arXiv:1707.04131, (2017).

[25] D. RUIZ-BALET AND E. ZUAZUA, *Neural ODE control for classification, approximation and transport*, arXiv preprint arXiv:2104.05278, (2021).

[26] L. RUTHOTTO AND E. HABER, *Deep neural networks motivated by partial differential equations*, Journal of Mathematical Imaging and Vision, 62 (2020), pp. 352–364.

[27] B. SMETS, J. PORTEGIES, E. J. BEKKERS, AND R. DUITS, *PDE-based group equivariant convolutional neural networks*, Journal of Mathematical Imaging and Vision, (2022), pp. 1–31.

[28] C. SZEGEDY, W. ZAREMBA, I. SUTSKEVER, J. BRUNA, D. ERHAN, I. J. GOODFELLOW, AND R. FERGUS, *Intriguing properties of neural networks*, CoRR, abs/1312.6199 (2014).

[29] A. TROCKMAN AND J. Z. KOLTER, *Orthogonalizing Convolutional Layers with the Cayley Transform*, in International Conference on Learning Representations, 2021.

[30] Y. TSUZUKU, I. SATO, AND M. SUGIYAMA, *Lipschitz-margin training: Scalable certification of perturbation invariance for deep neural network*, Advances in Neural Information Processing Systems, 31 (2018).

[31] J. WANG, Y. CHEN, R. CHAKRABORTY, AND S. X. YU, *Orthogonal convolutional neural networks*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11505–11515.

[32] E. WEINAN, *A proposal on machine learning via dynamical systems*, Communications in Mathematics and Statistics, 1 (2017), pp. 1–11.

[33] M. ZAKWAN, L. XU, AND G. FERRARI-TRECATE, *On Robust Classification using Contractive Hamiltonian Neural ODEs*, arXiv preprint arXiv:2203.11805, (2022).