DYNAMICS OF MODERN HOPFIELD NETWORKS

Sofiya Bogakovskaya
Laboratory of Computational Intelligence
s.bogakovskaya@gmail.com

Vladimir Fanaskov
AXXX

Ivan Oseledets
AXXX

ABSTRACT

We empirically analyze the dynamical characteristics of modern Hopfield networks (MHNs) in classification tasks. For bounded activation functions, trajectories are nearly linear, converge to class-specific memory vectors, and exhibit mean-shift dynamics. Conversely, rectified activation functions yield unstable dynamics where memories are instead encoded in the direction of the state vector. Using adversarial examples generated via projected gradient ascent, we show that while certain MHN architectures outperform MLPs in robustness, this trend is not universal. Critically, even robust MHNs possess global basin structures significantly more complex than standard MLP decision boundaries.

1 INTRODUCTION

The Hopfield network introduced in (Hopfield, 1982), (Hopfield, 1984) is a classical connectionist model of associative memory. A wealth of results is available, including the estimation of capacity (McEliece et al., 2003), the study of spurious states (Athithan & Dasgupta, 1997), the speed of convergence (Komlós & Paturi, 1988), relations to other pattern recognition methods (Lippmann et al., 1987), and so on. The model is typically trained with a biologically plausible Hebbian learning rule (Hopfield, 1982) and studied from the perspective of statistical physics or biology.

Contemporary deep learning prioritizes efficiency, scalability, and expressivity at the expense of interpretability and biologically plausible mechanisms. Given that, to a degree, modern deep learning practices are in tension with classical Hopfield networks. However, a group of authors recently showed that associative memory models can embrace the main tenets of contemporary deep learning (Krotov & Hopfield, 2020), (Ramsauer et al., 2020). These models, known as modern Hopfield networks (Krotov et al., 2025), have many viable properties: (i) they unify all previous memory models under a common formalism based on Lagrange functions (Krotov & Hopfield, 2020); (ii) it is possible to define hierarchical memory models analogous to classical deep neural networks (Krotov, 2021); (iii) modern Hopfield networks can be trained end-to-end with standard backpropagation through time (Hoover et al., 2022); (iv) they are related to classical energy-based models (LeCun et al., 2006), neural ordinary differential equations (Chen et al., 2018), and transformers (Ramsauer et al., 2020), (Hoover et al., 2024); (v) there is a link between modern Hopfield networks and diffusion models (Hoover et al., 2023).

Intriguing relations with leading architectures and generative deep learning approaches make modern Hopfield networks an interesting object of study. From a mathematical perspective, a modern Hopfield memory is a dynamical system. What are the properties of this system, especially after training is done? This is the main problem we investigate in this article. It is intrinsically hard to address this question from a theoretical perspective, so here we provide an initial empirical analysis. More specifically, we train modern Hopfield networks to classify input patterns following the overall setup of (Krotov & Hopfield, 2018) and (Hoover et al., 2022). The classification task was selected for two reasons: (i) the overall simplicity of the problem; (ii) the steady states are not completely prescribed, so there is room for learning abstract representations.

Our main contributions are:

1. We perform an extensive evaluation of Hopfield networks with distinct activation functions and integration times during training.
2. We experimentally evaluate the empirical capacity of Hopfield networks and find that during integration, class-specific samples compress toward distinct zones of attraction and

the estimated intrinsic dimension decreases, indicating local manifold compression. Additionally, individual classes often split into multiple distinct subsets, so class memory is distributed across several attractors.

3. We demonstrate that the typical trajectory of a state-space vector is very close to a straight line with almost zero local and global curvature.
4. We show that for Hopfield networks with activation functions from the rectifier family, trajectories converge to the point at infinity, and memories are encoded as directions.
5. We study the structure of the basins, including robustness to adversarial perturbations and their global arrangement. For adversarial robustness, we observe mixed results: Hopfield networks with ReLU activations are more robust than MLPs, while those with $\tanh$ activations are less robust. The global structure of Hopfield network basins of attraction is more complicated than the global structure of MLP decision regions, even in cases where the Hopfield network has better adversarial robustness.

2 Model Description and Problem Setting

The modern Hopfield network was proposed by Krotov & Hopfield (2021). The dynamics of this model can be compactly written as the following system of ordinary differential equations:

$$\dot{x}(t) = W g(x(t)) - x(t) + b, \quad x(0) = x_0, \tag{1}$$

where $x \in \mathbb{R}^N$ is a state-space vector and $g(x)$ is the activation function.

Under certain conditions, this system of ODEs admits an energy or Lyapunov function which decreases along the trajectories of the dynamical system, ensuring the existence of steady states interpreted as memories stored in the neural network. A detailed description is available in Appendix A and follows the arguments of Krotov & Hopfield (2020).

To empirically evaluate the modern Hopfield network (1), we consider the MNIST classification problem. We train a classifier parametrized by a Hopfield network, largely following Krotov & Hopfield (2018): (i) each class $i = 0, \ldots, 9$ is parametrized by a vector v where $v_k = +1$ if $k = i$ and $v_k = -1$ if $k \neq i$; (ii) the system of ODEs (1) is integrated for $t \in [0, T]$, where $T = t_{\text{train}}$ is the time of integration during training; (iii) the prediction for the class $\widetilde{v}$ is constructed from the last 10 components of the state vector $x(T)$; (iv) the loss used is $\|\widetilde{v} - v\|_2^2$. To evaluate gradients, we use backpropagation through time implemented in the Diffrax library (Kidger, 2021).

3 Results

3.1 Geometric properties of trajectories and attractors

We start by examining the evolution of states for a trained model with a $\tanh$ activation function and an integration time of $t_{\text{train}} = T = 15$ during training. To visualize the high-dimensional state vectors, we applied UMAP (McInnes et al., 2018), a dimension reduction method that constructs a low-dimensional embedding by approximating the data's topological structure and preserving local

(a) $t = 0$ (b) $t = 15$ (c) $t = 40$

Figure 1: Visualization of 500 trajectories, projected on low-dimensional space at $t = 0, 15, 40$.

neighborhoods. UMAP seeks a projection in which points that are nearby in the original space remain nearby in the embedding while also tending to preserve larger-scale relationships. The results presented in Figure 1 (with classes encoded by distinct colors) indicate that sample sets become progressively compressed and converge toward certain zones of attraction. At $t = 15.0$, the clusters are approximately equidistant from one another but oriented in different directions. Some sets disintegrate into smaller subsets.

When clusters are examined after $t_{\text{train}} = T = 15$, as seen in Figure 1c, they thin and adopt thread-like shapes with dense endpoints that correspond to basins of attraction. This evolution is analogous to mean-shift dynamics, where data points are iteratively transported toward local modes of the density (Milanfar, 2025).

This evolution is analogous to mean-shift dynamics, where data points are iteratively transported toward local modes of the density (Milanfar, 2025).

Table 1: Intrinsic Dimension (ID) and memory for selected MNIST digits.

class	1	2	3	4
ID at $x(0)$	10.03	12.93	15.37	12.82
ID at $x(T)$	7.03	8.32	8.47	7.63
clusters	2	2	2	2

Intrinsic dimension (ID) (Facco et al., 2017) measures the minimum number of degrees of freedom needed to describe local data variability. We estimate it with the 2 nearest neighbours estimator, which recovers this by modeling the distribution of ratios between the first and second nearest-neighbor distances, producing a robust local estimate of dimensionality. The intrinsic dimension at $x(T)$ is smaller than at $x(0)$; see Table 1 and Table 3 in the Appendix. A reduction in the estimated intrinsic dimension indicates that the data manifold becomes locally lower-dimensional.

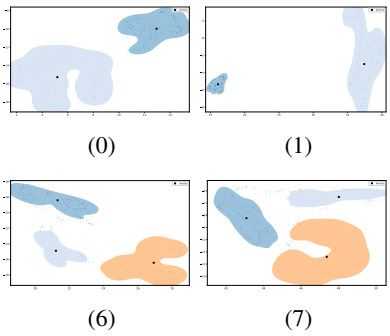


Figure 2: Interclass clustering.

Our problem setting does not specify memory vectors explicitly, so it is interesting to examine how many distinct memory vectors are allotted for each subclass.[1] In Figure 2 and Figure 6 in the Appendix, inter-class clustering is shown for each class, performed with the HDBSCAN method (McInnes et al., 2017), a hierarchical density-based algorithm that finds clusters of varying density and labels sparse points as noise. One can see that, for some classes, the subclusters are proportionate and clearly separated. The number of clusters, estimated by HDBSCAN, is presented in Table 1 and Table 3 in the Appendix. With rare exceptions, we observe that the number of distinct memory vectors for each MNIST digit is small, bounded roughly by 2 or 3.

It is also interesting to look at the geometric characteristics of trajectories. Two basic quantities that we measure are the trajectory length relative to the length of a straight line, $s = \int_{t_0}^{T} \|\dot{x}(t)\|_2 \, \mathrm{d}t / \|x(T) - x(t_0)\|_2$, and the curvature $\kappa = \|\mathbf{a} - \frac{\mathbf{a}\cdot\mathbf{v}}{\|\mathbf{v}\|_2^2}\mathbf{v}\|_2 / \|\mathbf{v}\|_2^2$.

In Figure 3a, we provide the distribution of s over 100 samples from every class. One can see that it is in the vicinity of 1, so trajectories are very close to straight lines. In Figure 3b, one can see that the curvature κ remains 0 at the beginning of the trajectory and then increases to small values of the order $\kappa \approx 0.15$. Thus, the local radius of the trajectory is about $R = \frac{1}{\kappa} \approx 6.7$, which is still large on the scale of the system. Consequently, the trajectories may be regarded as relatively straight.

We also performed a set of experiments to determine how integration time and the choice of activation influence the accuracy of the learned classifier. Detailed results and discussion can be found in Appendix C.

[1] Formally, the dynamical systems that we consider are invertible, but in practice, steady states are close to each other and can be replaced by a single vector.

3.2 HOPFIELD NETWORK WITH RELU ACTIVATIONS

For Hopfield networks with ReLU activations, we observe that the dynamical variables fail to converge to stationary states. Instead, all trajectories diverge such that $\|x(t)\|_2 \to \infty$ as t increases.

Integration of the original system of equations becomes problematic when $\|x(t)\|_2 \gg 1$. To stabilize the model, we introduce new variables $z(t) = 1/\|x(t)\|_2$ and $y(t) = x(t)z(t)$. In Appendix D, we show that:

$$\begin{aligned} \dot{y}(t) &= (I - y(t)y(t)^\top)(W\,\mathrm{ReLU}(y(t)) - y(t) + bz(t)), \\ \dot{z}(t) &= -y(t)^\top\,(W\,\mathrm{ReLU}(y(t)) - y(t) + bz(t))\,z(t). \end{aligned} \tag{2}$$

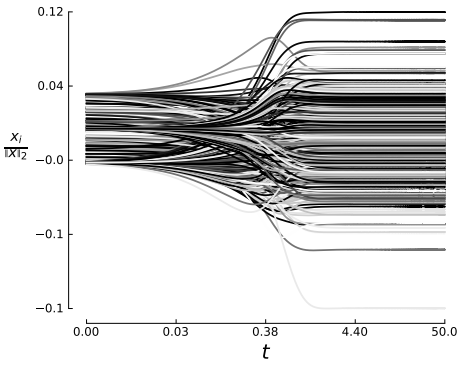


Figure 4: Evolution of $x(t)/\|x(t)\|_2$ for a Hopfield network with ReLU activations, $t_{\text{train}} \in [0, 0.1]$.

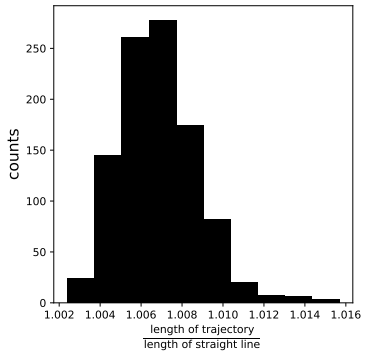
Typical trajectories of $y(t) = x(t)/\|x(t)\|_2$ are shown in Figure 4. It is evident that the components of $y(t)$ rapidly reach equilibrium values. In Appendix E, we also provide a plot of the inverse L_2 norm of the state vector, which shows that it converges to 0.

The stabilization of $y(t)$ and $z(t)$ suggests that it should be possible to find a well-defined Lyapunov function that demonstrates the stability of the trajectory at infinity, i.e., at $z(t) = 0$. As shown in Appendix D, the original energy function is not defined for $z(t) = 0$, so this problem remains open.

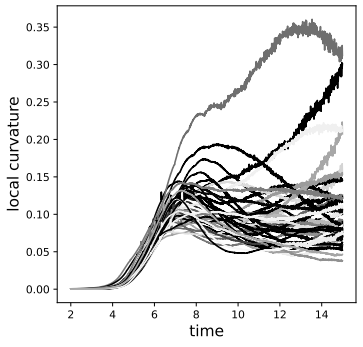
In Appendix E, we demonstrate similar behavior for the softmax activation function. It is likely that a trained Hopfield network with any activation function from the rectifier family: (i) leads to divergent trajectories $\|x(t)\|_2 \to \infty$ as $t \to \infty$; (ii) has convergent normalized state vectors $x(t)/\|x(t)\|_2 \to c(x_0)$. If this conjecture is correct, it implies that Hopfield networks with activations from the rectifier family can store memories in a novel, unintended way.

3.3 ADVERSARIAL PERTURBATIONS AND ATTRACTOR STRUCTURE

For the selected problem, Hopfield networks are completely specified by their attractors and basins. Describing the basin is a hard problem even for simple low-dimensional iterative systems; for example, the Newton fractal can be described as a set of basins of attraction of the Newton rootfinder



(a) Histogram of relative length of trajectories.



(b) Curvature over time for 50 samples.

Figure 3: Global and local curvatures of trajectories.

(a) MLP

(b) ReLU, $t_{\text{train}} \in [0, 1]$

Figure 5: Global arrangement of basins of attraction obtained with an $\epsilon = 0.1$ adversarial perturbation in the l_∞ norm. For the specified networks, we search for adversarial examples starting from a given test set sample and record the classification results after perturbation. Such information indirectly indicates which basins of attraction are located near each other. The results are demonstrated with a directed graph: if it is possible to change the classification result from digit M to digit N, a directed edge from M to N is present. It is clear that the Hopfield network leads to a more intricate global structure of basins, despite the fact that it is more robust to adversarial inputs.

(Hubbard et al., 2001). For high-dimensional systems, an additional complication is the curse of dimensionality, which likely prevents recovering fine details of basin geometry even in the case of MNIST with a $\simeq 10^3$-dimensional space. As a compromise, we estimate how close the basins of attraction are to each other.

From a machine learning perspective, a natural way to find the closest decision boundary is to construct adversarial examples. Similarly, in the case of Hopfield networks, adversarial examples show how close distinct basins of attraction are to one another. To find adversarial examples for Hopfield networks, we apply projected gradient ascent (Madry et al., 2017):

$$\delta^n = \frac{\epsilon \left(\delta^{n-1} + \alpha \nabla L(\mathcal{M}\left(x + \delta^{n-1}\right), t)\right)}{\max\left(\epsilon, \left\|\delta^{n-1} + \alpha \nabla L(\mathcal{M}\left(x + \delta^{n-1}\right), t)\right\|_\infty\right)} \Rightarrow \underset{N \gg 1}{\delta^N} \simeq \arg \max_{\|\delta\|_\infty \leq \epsilon} L(\mathcal{M}\left(x + \delta\right), t), \tag{3}$$

where $L(\mathcal{M}(x), t)$ is the loss function for a given feature x, trained model $\mathcal{M}$, and target t. Note that for a Hopfield network, the computation of $\mathcal{M}(x)$ requires the integration of an ODE.

The success of the perturbations for different ϵ, activation functions, and training integration times is given in Table 2 and Table 9 in the Appendix. The results are mixed: Hopfield networks with activations from the rectifier family are generally more robust than vanilla MLPs; the same is true for a sigmoid in the regime of small perturbations. However, Hopfield networks with $\tanh$ activations are the least robust, including MLPs. This last result is at odds with the well-known observation that neural ODEs are generally more robust than standard feed-forward models Yan et al. (2019).

Table 2: The success of adversarial perturbation (given in %) for several activation functions, $\|\delta\|_\infty \leq \epsilon$. The accuracy of the unperturbed model is available in the last line of the table.

ϵ	MLP	ReLU		σ	
t_{train}		0.1	1	0.1	1
0.06	8%	6%	8%	10%	6%
0.1	28%	13%	10%	22%	14%
0.5	71%	45%	60%	85%	76%
0.9	69%	49%	64%	92%	84%
1.3	70%	64%	71%	96%	90%
acc.	98%	98%	96%	94%	98%

Information about neighboring basins is provided in Figure 5. Clearly, the global structure of the basins is more entangled for Hopfield networks with ReLU activations compared to MLPs, even when the integration time is small. This result is somewhat paradoxical, given that associative memory with ReLU activation is more robust to adversarial perturbations than MLPs. These results suggest that while most samples are not located on the boundary of the basins, the boundaries themselves have complicated shapes, allowing many distinct nearby basins to be found.

REFERENCES

Gopalasamy Athithan and Chandan Dasgupta. On the problem of spurious patterns in neural associative memory models. *IEEE Transactions on Neural Networks*, 8(6):1483–1491, 1997.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7, 09 2017. doi: 10.1038/s41598-017-11873-y.

Benjamin Hoover, Duen Horng Chau, Hendrik Strobelt, and Dmitry Krotov. A universal abstraction for hierarchical hopfield networks. In *The Symbiosis of Deep Learning and Differential Equations II*, 2022.

Benjamin Hoover, Hendrik Strobelt, Dmitry Krotov, Judy Hoffman, Zsolt Kira, and Duen Horng Chau. Memory in plain sight: A survey of the uncanny resemblances between diffusion models and associative memories. In *Associative Memory {\&} Hopfield Networks in 2023*, 2023.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. *Advances in Neural Information Processing Systems*, 36, 2024.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

John J Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the national academy of sciences*, 81(10):3088–3092, 1984.

John Hubbard, Dierk Schleicher, and Scott Sutherland. How to find all roots of complex polynomials by newton's method. *Inventiones mathematicae*, 146(1):1–33, 2001.

Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

János Komlós and Ramamohan Paturi. Convergence results in an associative memory model. *Neural Networks*, 1(3):239–250, 1988.

Dmitry Krotov. Hierarchical associative memory. *arXiv preprint arXiv:2107.06446*, 2021.

Dmitry Krotov and John Hopfield. Dense associative memory is robust to adversarial inputs. *Neural computation*, 30(12):3151–3167, 2018.

Dmitry Krotov and John Hopfield. Large associative memory problem in neurobiology and machine learning. *arXiv preprint arXiv:2008.06996*, 2020.

Dmitry Krotov and John Hopfield. Large associative memory problem in neurobiology and machine learning, 2021. URL https://arxiv.org/abs/2008.06996.

Dmitry Krotov, Benjamin Hoover, Parikshit Ram, and Bao Pham. Modern methods in associative memory. *arXiv preprint arXiv:2507.06211*, 2025.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Richard Paul Lippmann, Bernard Gold, and Marylin L Malpass. A comparison of hamming and hopfield neural nets for pattern classification. Technical report, 1987.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Robert McEliece, Edward Posner, Eugene Rodemich, and Santosh Venkatesh. The capacity of the hopfield associative memory. *IEEE transactions on Information Theory*, 33(4):461–482, 2003.

Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017. doi: 10.21105/joss.00205. URL https://doi.org/10.21105/joss.00205.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018. doi: 10.21105/joss.00861. URL https://doi.org/10.21105/joss.00861.

Peyman (@docmilanfar) Milanfar. Mean-shift iteratively moves points towards regions of higher density, 2025. URL https://x.com/docmilanfar/status/1922138004346306758. Post on X (formerly Twitter).

Hubert Ramsauer, Bernhard Schäfl, Johannes Lehner, Philipp Seidl, Michael Widrich, Thomas Adler, Lukas Gruber, Markus Holzleitner, Milena Pavlović, Geir Kjetil Sandve, et al. Hopfield networks is all you need. *arXiv preprint arXiv:2008.02217*, 2020.

Hanshu Yan, Jiawei Du, Vincent YF Tan, and Jiashi Feng. On robustness of neural ordinary differential equations. *arXiv preprint arXiv:1910.05513*, 2019.

A Details on the modern Hopfield networks

Modern Hopfield networks considered in the main text have a form

$$\dot{x}(t) = Wg(x(t)) - x(t) + b, \quad x(0) = x_0, \tag{4a}$$

$$g(x) = \frac{\partial L(x)}{\partial x}, \quad W = W^\top, \tag{4b}$$

$$\Lambda_{ij}(x) = \frac{\partial^2 L(x)}{\partial x_i \partial x_j}, \ \Lambda(x) \geq 0, \tag{4c}$$

$$E(x) = (x-b)^\top g(x) - L(x) - \tfrac{1}{2}\left(g(x)\right)^\top W g(x). \tag{4d}$$

We primarily consider activations that strictly satisfy (4c) (for example $\tanh$ or sigmoid), with Hessian having zero eigenvalues (e.g., ReLU), and also activations that do not satisfy (4c) (e.g., GELU). Note that for ReLU activations the energy is unbounded from below.

Equation 4a defines the evolution of network state $x(t)$ over time t. Here W is a symmetric weight matrix, b is a bias vector and $g(x)$ is the activation function. Condition 4b states that $g(x)$ is the gradient of the Lagrangian $L(x)$ and ensures that the energy function is defined. Condition 4c ensures the energy decreases over time and 4d defines the energy function.

Proof:

We choose energy function:

$$E(x) = (x-b)^\top g(x) - L(x) - \frac{1}{2}(g(x))^\top W g(x), \tag{5}$$

$$dE(x) = dx^\top g(x) + (x-b)^\top \frac{\partial g(x)}{\partial x} dx - \frac{\partial L}{\partial x} dx - \nabla^2 L \frac{W + W^\top}{2} g(x) dx \tag{6}$$

Considering $g(x) = \frac{\partial L}{\partial x}$ and $W = W^\top$:

$$dE(x) = \nabla^2 L(x - b - Wg(x))dx \tag{7}$$

So

$$\frac{\partial E}{\partial x} = \nabla^2 L(x)[x - Wg(x) - b] \tag{8}$$

is true and we can choose the system with dynamics:

$$\dot{x} \equiv Wg(x) - x + b. \tag{9}$$

Then existing of stationary states would mean derivative of energy over t aims to 0:

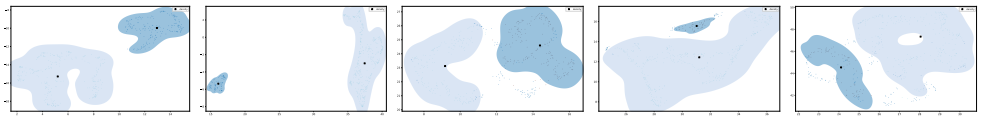
$$\frac{dE}{dt} = \frac{\partial E}{\partial x}\dot{x} = [x - Wg(x) - b]\nabla^2 L(x)\dot{x} = -\dot{x}^\top \nabla^2 L(x)\dot{x} \leq 0 \tag{10}$$

$\nabla^2 L > 0$ guarantees that energy is strictly decreasing and it's derivative is 0 when stationary states are achieved.

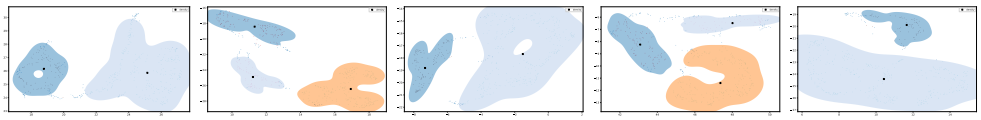
B ADDITIONAL RESULTS ON THE STRUCTURE OF ATTRACTORS

Table 3: How much memory is allocated for each number

class	0	1	2	3	4	5	6	7	8	9
ID at $x(0)$	11.94	10.03	12.93	15.37	12.82	13.51	11.94	12.19	14.04	13.05
ID at $x(t_1)$	8.35	7.03	8.32	8.47	7.63	6.98	5.81	8.32	8.43	8.95
clusters	2	2	2	2	2	2	3	2	3	2



(a) 0 (b) 1 (c) 2 (d) 3 (e) 4



(f) 5 (g) 6 (h) 7 (i) 8 (j) 9

Figure 6: Clusterization done for every class

C HOW MODEL PERFORMANCE DEPENDS ON THE INTEGRATION TIME

In this section we will try to analyze, how neural network performance depends on different activation functions and time-related hyperparameters like time of integration t_1 and step of integration dt.

Models with ReLU learn very fast with high accuracy. Tables in Appendix C show that models with ReLU activation function consistently produce very good results. We will return to this in the following Section 3.2.

Models with hyperbolic tangent also produces excellent results at all time scales. Reducing learning rate significantly improved accuracy at long time scales.

Models with sigmoid and softplus give high accuracy at short time scales but break down at long time scales. Further reduction of the learning rate would make training excessively long and may not yield the desired improvement.

There may be several reasons why models fail to learn at long time scales: sensitivity to initial conditions, error accumulation, and changes in the position of attractors or attraction zones.

Tables below show the accuracies of models with varying activation functions (relu, gelu, tanh, sigmoid, softplus), step $dt \in \{0.01, 0.05, 0.1\}$ and integration time $T = \{1.0, 3.0, 5.0, 10.0, 15.0\}$. Other hyperparameter listed in the table description.

We consider a model well-trained if its accuracy on the test set exceeds 95%. If accuracy exceeds 90% and the loss and accuracy dynamics are "correct" (loss strictly decreasing and accuracy strictly increasing), we will also consider it "good" – meaning it requires some additional training. Models

Table 4: Performance with ReLU activation (epochs: 20, batch_size = 128, lr = 1e-3)

dt\T	**1.0**	**3.0**	**5.0**	**10.0**	**15.0**
0.01	**97.7%**	**97.5%**	**97.7%**	**97.3%**	**96.7%**
0.05	**97.7%**	**97.9%**	**96.7%**	**97.2%**	**96.6%**
0.10	**97.3%**	**97.5%**	**97.4%**	**97.0%**	**96.3%**

Table 5: Performance with GELU activation (epochs: 20, batch_size = 128, lr = 1e-3)

dt\T	**1.0**	**3.0**	**5.0**	**10.0**	**15.0**
0.01	**97.8%**	**97.2%**	**97.1%**	**95.7%**	**91.2%**
0.05	**97.5%**	**97.5%**	**96.9%**	**94.3%**	73.3%
0.10	**97.6%**	**97.2%**	**97.0%**	**96.1%**	85.1%

with low accuracy ($< 70\%$) will be considered "bad". "Good" models are highlighted with bold font.

D STABILISED RELU MODEL

Here we derive equations (2). Recall, that original dynamical system reads

$$\dot{x}(t) = W\,\mathrm{ReLU}(x(t)) - x(t) + b. \tag{11}$$

Observe that

$$\dot{x}(t) = \frac{d}{dt}\left(\frac{x(t)}{\|x(t)\|_2}\,\|x(t)\|_2\right) = \frac{d}{dt}\left(y(t)\,\|x(t)\|_2\right) = \dot{y}(t)\,\|x(t)\|_2 + y(t)\,(y(t))^\top\,\dot{x}(t), \tag{12}$$

or

$$\dot{y}(t) = (I - y(t)\,(y(t))^\top)\dot{x}(t)/\,\|x(t)\|_2\,. \tag{13}$$

Since ReLU is homogeneous function we also have

$$\dot{x}(t) = W\,\mathrm{ReLU}(x(t)) - x(t) + b = \|x(t)\|_2\,(W\,\mathrm{ReLU}(y(t)) - y(t) + bz(t))\,. \tag{14}$$

The last two expressions confirm ODE for $y(t)$ from the main text.

ODE for $z(t)$ can be confirmed as follows

$$\dot{z}(t) = \frac{d}{dt}\left(\frac{1}{\|x(t)\|_2}\right) = -\frac{(y(t))^\top\,\dot{x}(t)}{\|x(t)\|_2^2} = -y(t)^\top\,(W\,\mathrm{ReLU}(y(t)) - y(t) + bz(t))\,z(t). \tag{15}$$

E SUPPLEMENTARY RESULTS ON ACTIVATIONS FROM RECTIFIER FAMILY

As an example of an activation function from the rectifier family we consider the softplus function. For this case activation and Lagrange functions are

$$g(x) = \log\left(1 + \exp(x)\right),\; L(x) = -\mathrm{Li}_2(-\exp(x)),\; \mathrm{Li}_2(x) = -\int_0^x \frac{\log\left(1-u\right)}{u}du. \tag{16}$$

Mathematically equivalent, but more numerically stable, expression that we use in the implementation reads

$$\begin{aligned} g(x) &= \begin{cases} x + \log(1+\exp(-x)),\; x > 0; \\ \log(1+\exp(-x)),\; x \leq 0. \end{cases} \\ L(x) &= \begin{cases} \frac{x^2}{2} + \frac{1}{2}\mathrm{Li}_2(\exp(-2x)) - \mathrm{Li}_2(\exp(-x)),\; x > 0; \\ \mathrm{Li}_2(\exp(x)) - \frac{1}{2}\mathrm{Li}_2(\exp(2x)),\; x \leq 0. \end{cases} \end{aligned} \tag{17}$$

As one can see in Figure 8 the dynamic is qualitatively similar to the dynamics of Hopfield ReLU network, with stabilisation of direction toward point at infinity.

Evolution of the inverse norms of state vectors for both ReLU and softplus activations are available in Figure 9. One can clearly see that trajectories diverge.

Table 6: Performance with tanh activation (epochs: 20, batch_size = 128)

Learning rate = 1e-3					
dt\T	**1.0**	**3.0**	**5.0**	**10.0**	**15.0**
0.01	**97.1%**	**96.5%**	**95.5%**	29.8%	16.1%
0.05	**97.0%**	**96.4%**	**94.6%**	**88.8%**	17.1%
0.10	**96.9%**	**96.1%**	**95.3%**	9.9%	19.3%
Learning rate = 1e-4					
dt\t1	**1.0**	**3.0**	**5.0**	**10.0**	**15.0**
0.01	-	-	**95.9%**	**95.4%**	**95.4%**
0.05	-	-	**96.1%**	**96.2%**	**95.8%**
0.10	-	-	**95.8%**	**95.7%**	**96.0%**

Table 7: Performance with sigmoid activation (epochs: 20, batch_size = 128)

Learning rate = 1e-3					
dt\T	**1.0**	**3.0**	**5.0**	**10.0**	**15.0**
0.01	**92.4%**	**91.0%**	**89.0%**	10.4%	9.9%
0.05	**92.5%**	**92.2%**	**90.7%**	10.4%	11.2%
0.10	**91.5%**	**91.9%**	**89.4%**	11.2%	11.2%
Learning rate = 1e-4					
dt\T	**1.0**	**3.0**	**5.0**	**10.0**	**15.0**
0.01	-	-	74.6%	11.2%	9.9%
0.05	-	-	74.8%	11.2%	10.4%
0.10	-	-	76.8%	9.9%	11.2%

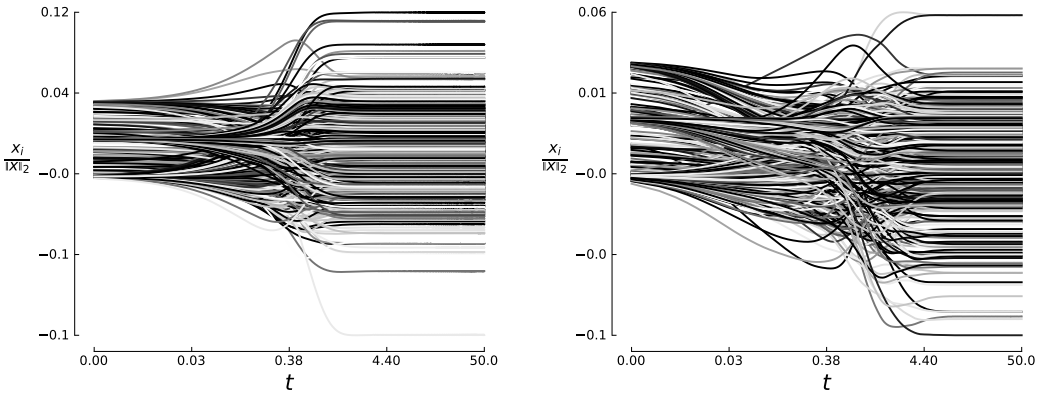
F SUPPLEMENTARY RESULTS ON THE STRUCTURE OF BASINS AND ADVERSARIAL ROBUSTNESS

Global structure of basins in available in Figure 10.

Additional results on adversarial robustness are in Table 9.

Table 8: Performance with softplus activation (epochs: 20, batch_size = 128)

Learning rate = 1e-3					
dt\T	**1.0**	**3.0**	**5.0**	**10.0**	**15.0**
0.01	**95.0%**	**93.5%**	**92.7%**	11.2%	11.2%
0.05	**93.8%**	**94.4%**	**93.2%**	11.2%	10.2%
0.10	**94.8%**	**94.5%**	**90.9%**	9.9%	9.9%
Learning rate = 1e-4					
dt\t1	**1.0**	**3.0**	**5.0**	**10.0**	**15.0**
0.01	-	-	**91.3%**	76.2%	11.2%
0.05	-	-	**89.9%**	74.9%	11.2%
0.10	-	-	**89.2%**	76.4%	10.4%

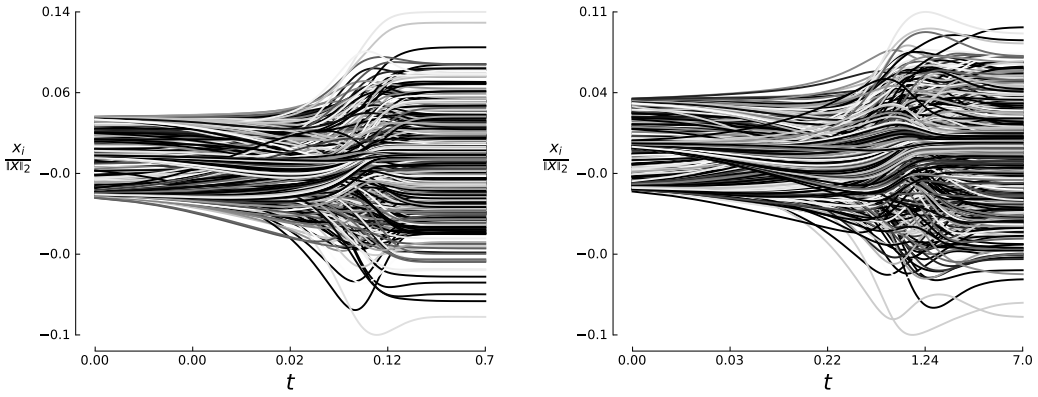


(a) $t_{\text{train}} \in [0, 0.1]$ (b) $t_{\text{train}} \in [0, 1]$

Figure 7: Evolution of $x(t) / \|x(t)\|_2$ for Hopfield network with ReLU activations. Despite state vector $x(t)$ being divergent, $\|x(t)\|_2 \to \infty$ for $t \to \infty$, the *direction* to the point at infinity stabilizes over time. When a smaller time interval is used during training, the trajectories are more regular and demonstrate faster convergence to stationary values.

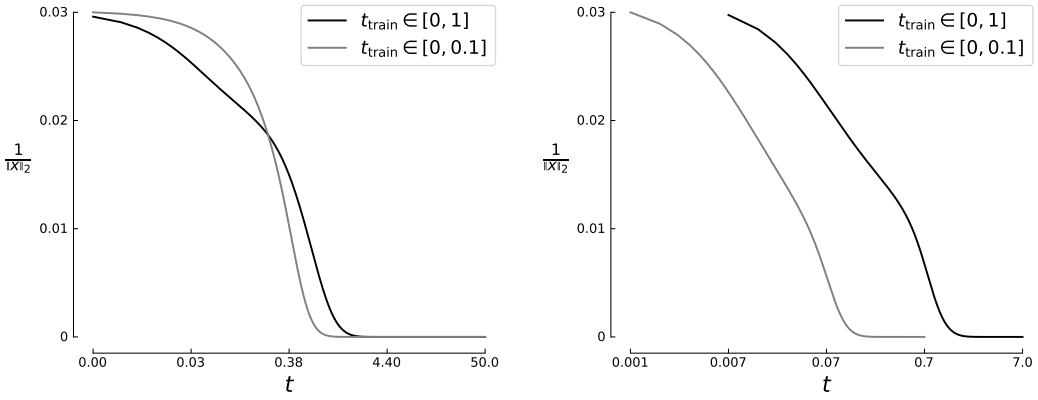
Table 9: Table with the success of adversarial perturbation (given in %) for several activation functions, $\|\delta\|_\infty \leq \epsilon$. The accuracy of the unperturbed model is available in the last line of the table. Accuracy is reported for a randomly selected subset of test set.

ϵ	MLP	ReLU		softplus		σ		tanh	
t_{train}		0.1	1	0.1	1	0.1	1	0.1	1
0.06	8%	6%	11%	5%	8%	10%	6%	15%	15%
0.1	28%	13%	25%	11%	10%	22%	14%	30%	30%
0.5	71%	45%	50%	59%	60%	85%	76%	93%	99%
0.9	69%	49%	59%	67%	64%	92%	84%	100%	97%
1.3	70%	64%	69%	73%	71%	96%	90%	100%	98%
acc.	98%	98%	99%	96%	96%	94%	98%	100%	98%



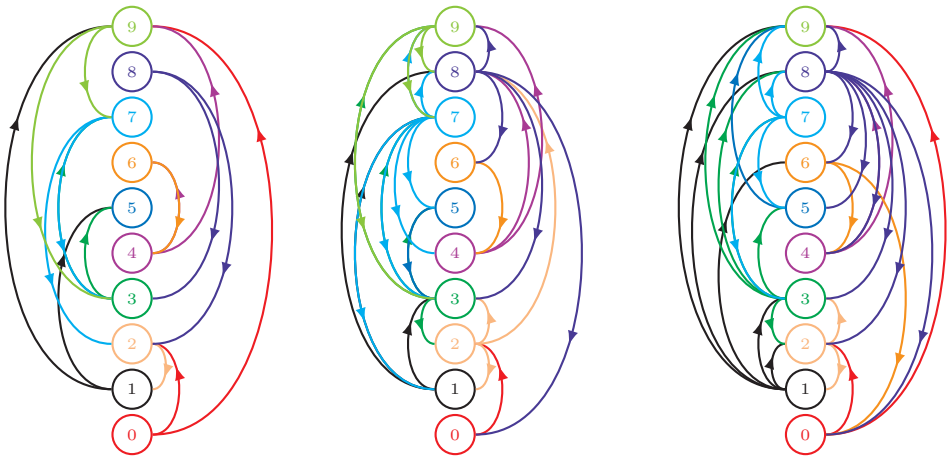
(a) $t_{\text{train}} \in [0, 0.1]$ (b) $t_{\text{train}} \in [0, 1]$

Figure 8: Evolution of $x(t) / \|x(t)\|_2$ for Hopfield network with softplus activations. We observe the same stabilisation as for Hopfield network with ReLU activations.



(a) ReLU (b) softplus

Figure 9: Evolution of $1 / \|x(t)\|_2$ for Hopfield network with ReLU and softplus activations. For both cases state vectors clearly converge to the point at infinity, i.e., $\|x(t)\|_2 \to \infty$ as $t \to \infty$.



(a) MLP (b) ReLU, $t_{\text{train}} \in [0, 0.1]$ (c) ReLU, $t_{\text{train}} \in [0, 1]$

Figure 10: Global arrangement of basins of attraction obtained with an $\epsilon = 0.1$ adversarial perturbation in the l_∞ norm. For the specified networks, we search for adversarial examples starting from a given test set sample and record the classification results after perturbation. Such information indirectly indicates which basins of attraction are located near each other. The results are demonstrated with a directed graph: if it is possible to change the classification result from digit M to digit N, a directed edge from M to N is present. It is clear that the Hopfield network leads to a more intricate global structure of basins, despite the fact that it is more robust to adversarial inputs.