Supplementary Material

1 Definitions of the system, training

1.1 Notation

All vectors are denoted using the bra-ket notation. For vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$, we write

$$\boldsymbol{v} \to |v\rangle\,, \tag{S.1}$$

$$\boldsymbol{u}^T \to \langle u|\,. \tag{S.2}$$

Dot products in bra-ket notation are written as $\langle u|v\rangle$, for two vectors in the same vector space.

In the context of this work, this means the network's visible and hidden detectors become memories $|M_\mu\rangle$ and labels $|L_\mu\rangle$, respectively. The memories $|M_\mu\rangle$ live in the *input space*, and the labels $|L_\mu\rangle$ live in the *output space*. To manipulate them, we define the following memory and label orthonormal bases,

$$\{|\hat{m}_i\rangle\}, \quad i \in [0, 784) \tag{S.3}$$

$$\left\{ \left|\hat{l}_d\right\rangle \right\}, \quad d \in [0, 9]. \tag{S.4}$$

Intuitively, the memory basis fetches individual pixels of a 28x28 (MNIST) image. While, the label basis can be understood as isolating a probability of being (or not being) of category/class d.

1.2 Generalized Hopfield Network classifier

In this work, the input, $|\sigma\rangle$, used is a flattened 28×28 image of a handwritten (MNIST) digit. The pixels of the image range between -1 and 1. Generally, when illustrated a continous blue-white-red ('bwr') colormap is used, where blue is -1, white is 0 and red is $+1$.

The network's task is to classify the input image $|\sigma\rangle$ into its digit class (0, 1, 2, ..., 9). The output is a vector, where each element can be understood as the probability of being (or **not** being) of digit class d. The elements of the output range continuously from -1 (is **not** of digit d) to $+1$ (is of digit class d). The output vector is computed as follows,

$$|o(\sigma)\rangle = \tanh\left(\sum_\mu |L_\mu\rangle\, f_n\Big(\frac{\langle M_\mu|\sigma\rangle}{T}\Big) \right), \tag{S.5}$$

Notice each memory $|M_\mu\rangle$ is associated to a label vector $|L_\mu\rangle$, and tanh is applied element-wise. Also, a non-linear function f is applied,

$$f_n(x) = \Big(ReLU(x)\Big)^n = \begin{cases} x^n, & x \geq 0 \\ 0, & x < 0 \end{cases} . \tag{S.6}$$

Due to the nature of the inputs, $\langle M_\mu|\sigma\rangle$ is almost always positive. Hence, we can focus on the case where

$$\boxed{|o(\sigma)\rangle = \tanh\left(\sum_\mu |L_\mu\rangle \left(\frac{\langle M_\mu|\sigma\rangle}{T}\right)^n\right).} \tag{S.7}$$

The hyperparameters of the system are then n and T, and control the non-linearity of the activation function.

1.2.1 The rescaled temperature

The temperature T can be seen as a normalization quantity on the memory dot products $\langle M_\mu|\sigma\rangle$. To add some intuition to this parameter, we express it as a fraction of the theoretical maximum dot product. This is referred to as the rescaled temperature (T_r).

As we'll see in further sections, elements (or pixels) of $|M_\mu\rangle$ and $|\sigma\rangle$ are restricted to be between -1 and $+1$, hence the theoretical maximum dot product is equal to the number of pixels/the dimensionality of the training data (D). In the MNIST case, this leads to the following definition:

$$T_r = \frac{T}{784} \iff T = 784T_r. \tag{S.8}$$

1.3 Training

The memories and labels are obtained through training. To that end, we define a training set $\mathcal{T}$ composed of training (input) samples $|\sigma\rangle$ with a known expected output $\left|t_{|\sigma\rangle}\right\rangle$. To give an example of the structure of $\left|t_{|\sigma\rangle}\right\rangle$, if $|\sigma\rangle$ is a 5, then the expected output will be,

$$\left\langle \hat{l}_5 \middle| t_{|\sigma\rangle} \right\rangle = +1, \tag{S.9}$$

$$\left\langle \hat{l}_{d\neq 5} \middle| t_{|\sigma\rangle} \right\rangle = -1. \tag{S.10}$$

In another form, $\left|t_{|\sigma\rangle}\right\rangle = (-1,-1,-1,-1,-1,+1,-1,-1,-1,-1)$. Each training step then consists in gradient descent followed by normalizations/clipping.

1.3.1 Cost function

Gradient descent implies an energy/cost function that the system tries to optimize. In this classification task, the system will try to minimize the difference between the expected output

and the network's computed output. That is,

$$C = \sum_{|\sigma\rangle \in \mathcal{T}} \sum_{d} \left(\left\langle \hat{l}_d \middle| t_{|\sigma\rangle} \right\rangle - \left\langle \hat{l}_d \middle| o(\sigma) \right\rangle \right)^{2n} \tag{S.11}$$

is minimized. [1]

1.3.2 Training dynamics

The dynamics are:

$$|M_\mu(t+\delta t)\rangle = N_\mu(t) \left(|M_\mu(t)\rangle - \delta t \Big(\max_j \Big| \frac{\partial C}{\partial \langle \hat{m}_j | M_\mu(t) \rangle} \Big| \Big)^{-1} \frac{\partial C}{\partial \langle M_\mu(t)|} \right), \tag{S.12}$$

$$|L_\mu(t+\delta t)\rangle = \mathcal{C} \left(|L_\mu(t)\rangle - \delta t \left(\max_d \left| \frac{\partial C}{\partial \left\langle \hat{l}_d \middle| L_\mu(t) \right\rangle} \right| \right)^{-1} \frac{\partial C}{\partial \langle L_\mu(t)|} \right). \tag{S.13}$$

For a given memory, the gradient is calculated - a vector - and is normalized so that its largest element has amplitude 1. The normalized gradient is subtracted from the memory with a factor δt.
The result is normalized by its largest absolute pixel greater than 1 (if none are greater than 1, 1 is used). This normalization operation is encompassed in $N_\mu(t)$. A similar process is followed for the labels, but the last normalization step is replaced by clipping, $\mathcal{C}(\cdot)$.

$$N_\mu(t) = \max \Big(1, \max_i \Big| \langle \hat{m}_i | M_\mu(t) \rangle - \delta t \Big(\max_j \Big| \frac{\partial C}{\partial \langle \hat{m}_j | M_\mu(t) \rangle} \Big| \Big)^{-1} \frac{\partial C}{\partial \langle \hat{m}_i | M_\mu(t) \rangle} \Big| \Big), \tag{S.14}$$

$$\mathcal{C}(x) = \begin{cases} +1, & x > +1 \\ -1, & x < -1 \\ x, & else \end{cases} . \tag{S.15}$$

[1] The exponent is generally written as $2m$, but in all simulations $m = n$ is used.

The gradient descent quantities, can be expanded into

$$\frac{\partial C}{\partial \langle M_\mu(t)|} = -\sum_{|\sigma\rangle\in\mathcal{T}}\sum_d\left(\left\langle \hat{l}_d\middle| t_{|\sigma\rangle}\right\rangle - \left\langle \hat{l}_d\middle| o(\sigma)\right\rangle\right)^{2n-1} \times\left(1-\left\langle \hat{l}_d\middle| o(\sigma)\right\rangle^2\right)\left\langle \hat{l}_d\middle| L_\mu\right\rangle\left(\frac{\langle M_\mu|\sigma\rangle}{T}\right)^{n-1}|\sigma\rangle \tag{S.16}$$

$$\frac{\partial C}{\partial \langle L_\mu(t)|} = -\sum_{|\sigma\rangle\in\mathcal{T}}\sum_d\left(\left\langle \hat{l}_d\middle| t_{|\sigma\rangle}\right\rangle - \left\langle \hat{l}_d\middle| o(\sigma)\right\rangle\right)^{2n-1} \times\left(1-\left\langle \hat{l}_d\middle| o(\sigma)\right\rangle^2\right)\left(\frac{\langle M_\mu|\sigma\rangle}{T}\right)^{n}\left|\hat{l}_d\right\rangle \tag{S.17}$$

Due to normalization, (absolute) constant overall factors do not matter in the gradient descent quantities.

1.3.3 Minibatchs and Stochastic Gradient Descent

Alternately, the training set $\mathcal{T}$ is sometimes broken down into minibatchs, $\mathcal{T}_i$. Practically, this changes our training sample sums,

$$\sum_{|\sigma\rangle\in\mathcal{T}} \to \sum_{|\sigma\rangle\in\mathcal{T}_i\in\{\mathcal{T}_i\}}, \tag{S.18}$$

at each training epoch a minibatch is picked within the set of all minibatchs $\{\mathcal{T}_i\}$. The picking method can be deterministic (e.g. simple iteration) or stochastic (e.g. i is picked from a uniform random distribution). The latter is referred to as Stochastic Gradient Descent (SGD), while the former is standard minibatch training. Typically, minibatchs are used to speed up training, however, for simplicity we use a single $\mathcal{T}$, and no minibatchs unless otherwise specified (e.g. Fig. S.5 - S.8).

1.3.4 Other forms of stochasticiy

Note that (S.16) and (S.17) are purely deterministic. However, when examining the robustness of the dynamics in Figs. S.13 - S.16 small gaussian noise is added to the gradient descent quantities at each training epoch.

2 Memory Coefficients

Gradient descent quantities are effectively a weighed sum of the training samples. The normalizations are simply additional factors, hence any memory (with small initial conditions) can be decomposed into

$$|M_\mu\rangle = \sum \alpha_{\mu,|\sigma\rangle}\,|\sigma\rangle\,. \tag{S.19}$$

Likewise, the training dynamics can be written in terms of the α-coefficients.

2.1 The Moore-Penrose inverse

One way to obtain the α's, is to use the Moore-Penrose inverse. Consider the following matrix,

$$\boldsymbol{T} = \begin{pmatrix} \langle \sigma_1 | \\ \langle \sigma_2 | \\ \langle \sigma_3 | \\ \vdots \\ \langle \sigma_{N_T} | \end{pmatrix}, \tag{S.20}$$

containing all N_T training samples. Then, there always exists a Singular Value Decomposition (SVD) such that,

$$\boldsymbol{T} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T, \tag{S.21}$$

where $\boldsymbol{S}$ is a diagonal matrix (generally not square) with entries (all positive) sorted in descending order, and $\boldsymbol{U}, \boldsymbol{V}$ are unitary; $\boldsymbol{U}^T\boldsymbol{U} = I$. We define then, for this diagonal matrix $\boldsymbol{S}$, a pseudo inverse

$$(\boldsymbol{S}^\dagger)_{ij} = \begin{cases} \frac{1}{\boldsymbol{S}_{ij}}, & \boldsymbol{S}_{ij} > \epsilon \\ 0, & else \end{cases} . \tag{S.22}$$

That is we take the reciprocal of all diagonal entries above a tolerance ϵ. Then, we can define the pseudo-matrix for all matrices, and in particular our training set matrix:

$$\boldsymbol{T}^\dagger = \boldsymbol{V}\boldsymbol{S}^\dagger\boldsymbol{U}^{\boldsymbol{T}}. \tag{S.23}$$

An important property of the pseudo-inverse is[2],

$$\boldsymbol{T}\boldsymbol{T}^\dagger\boldsymbol{T} = \boldsymbol{T}. \tag{S.24}$$

Consider a memory $|M\rangle$ with an unknown composition, and we obtain some set of α's by using the pseudo-inverse,

$$\boldsymbol{\alpha}^T_{inv} = \langle M | \boldsymbol{T}^\dagger = \boldsymbol{\alpha}^T_{unknown}\boldsymbol{T}\boldsymbol{T}^\dagger, \tag{S.25}$$

which may or may not be the same as $\boldsymbol{\alpha}^T_{unknown}$, we are guaranteed by the property highlighted above that the reconstruction is the same,

$$\boldsymbol{\alpha}^T_{inv}\boldsymbol{T} = \boldsymbol{\alpha}^T_{unknown}\boldsymbol{T}. \tag{S.26}$$

Uniqueness of our α representation, depends on the rank of $\boldsymbol{T}$, but existence and equivalence are always guaranteed. For examples, see Fig. S.1.
Practically, we use the Numpy library to compute the Moore-Penrose inverse, with a tolerance value equal to $10^{-15}\times$ the largest singular value. Generally, the largest deviation we observe between a reconstructed memory and the original is in the order of 10^{-6}.

[2] $\boldsymbol{T}\boldsymbol{T}^\dagger\boldsymbol{T} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T\boldsymbol{V}\boldsymbol{S}^\dagger\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{U}\boldsymbol{S}\boldsymbol{S}^\dagger\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T = \boldsymbol{T}$. Which is exact if $\epsilon = 0$, but numerically we often choose $\epsilon \sim 10^{-15}$, and singular values less than that are approximated to be zero/negligible.

2.1.1 Coarse-graining α

The α-representation is a limited visualization tool when the training set contains multiple examples per digit class. In that case, to properly visualize the dynamics we define,

$$\bar{\alpha}_d = \sum_{|\sigma\rangle \in \mathcal{T}_d} \alpha_{|\sigma\rangle} \tag{S.27}$$

where $\mathcal{T}_d$ is a subset of $\mathcal{T}$ which contains only digit of class d. The intuition here is as follows, consider a memory $|M\rangle$, one can show it can be decomposed into

$$|M\rangle = \sum_d \bar{\alpha}_d \, |\bar{\sigma}_d\rangle + \sum_d \sum_{|\sigma\rangle \in \mathcal{T}_d} \alpha_{|\sigma\rangle} \, |\delta\sigma\rangle \, . \tag{S.28}$$

Marginalization then recovers the proportions of class-averaged digits $|\bar{\sigma}_d\rangle$. Ignoring digit specific variations $|\delta\sigma\rangle$. E.g. $\bar{\alpha}_1$ represents the proportion of the average 1 in a memory.

3 Study of the first saddle / The 1-memory system

Due to the small initial conditions, we can approximate the memories of a system to be initially identical. For a system of N_k memories, this means

$$|o(\sigma)\rangle = \tanh\left(\sum_\mu^{N_k} |L_\mu\rangle \left(\frac{\langle M_\mu | \sigma \rangle}{T} \right)^n \right) \approx \tanh\left(N_k \, |L\rangle \left(\frac{\langle M | \sigma \rangle}{T} \right)^n \right), \tag{S.29}$$

$$|o(\sigma)\rangle \approx \tanh\left(|L\rangle \left(\frac{\langle M | \sigma \rangle}{\tilde{T}} \right)^n \right), \tag{S.30}$$

Hence, any system initially behaves as a 1-memory system (see Fig. S.21 and Fig. S.22) with an effective temperature,

$$\tilde{T} = \frac{T}{\sqrt[n]{N_k}}. \tag{S.31}$$

It is then important to understand the behavior of the fixed points of the 1-memory system, as they become the saddles of higher-dimensional systems.

3.1 The analytical case : 2 digits

Consider the case of two training samples, $|A\rangle$, $|B\rangle$, of (distinct) classes A and B respectively. We define the following shorthand for our basis, $\left|\hat{l}_A\right\rangle$ for class A, $\left|\hat{l}_B\right\rangle$ for class B and $\left|\hat{l}_\gamma\right\rangle$ for any class other than A, B. This means,

$$\left\langle \hat{l}_A \middle| t_{|A\rangle} \right\rangle = 1 = \left\langle \hat{l}_B \middle| t_{|B\rangle} \right\rangle, \tag{S.32}$$

$$\left\langle \hat{l}_B \middle| t_{|A\rangle} \right\rangle = \left\langle \hat{l}_A \middle| t_{|B\rangle} \right\rangle = -1 = \left\langle \hat{l}_\gamma \middle| t_{|A\rangle} \right\rangle = \left\langle \hat{l}_\gamma \middle| t_{|B\rangle} \right\rangle. \tag{S.33}$$

Next for $|A\rangle \neq |B\rangle$, we can always construct[3] $|A'\rangle, |B'\rangle$ such that,

$$\langle A'|A\rangle = \langle B'|B\rangle = 1, \tag{S.34}$$

$$\langle A'|B\rangle = \langle B'|A\rangle = 0. \tag{S.35}$$

For the memory we write,

$$|M\rangle = \alpha_{|A\rangle}|A\rangle + \alpha_{|B\rangle}|B\rangle. \tag{S.36}$$

3.1.1 The ℓ invariant manifold

For the label, plugging $\left\langle \hat{l}_A \middle| L \right\rangle = -\left\langle \hat{l}_B \middle| L \right\rangle$ into (S.13, S.17) leads directly to $\partial_t \left\langle \hat{l}_A \middle| L \right\rangle = -\partial_t \left\langle \hat{l}_B \middle| L \right\rangle$. This means we have an invariant manifold. As for stability, one can define

$$\left\langle \hat{l}_A \middle| L \right\rangle = \ell - \delta\ell, \tag{S.37}$$

$$\left\langle \hat{l}_B \middle| L \right\rangle = -\Big(\ell + \delta\ell\Big), \tag{S.38}$$

and obtain a first order approximation of $\partial_t(\delta\ell)$, this is long so we omit it, but it leads to wide regions of stability. Also, by examining the dynamics one can see that $\left\langle \hat{l}_\gamma \middle| L \right\rangle \to -1$ monotonically.[4]

The 1-memory system enters this ℓ-manifold quickly and most of the interesting dynamics occur within it (see Fig. S.21 and Fig. S.22). Hence, we study the dynamics when

$$\left\langle \hat{l}_A \middle| L \right\rangle = \ell = -\left\langle \hat{l}_B \middle| L \right\rangle, \tag{S.39}$$

$$\left\langle \hat{l}_\gamma \middle| L \right\rangle = -1. \tag{S.40}$$

Notice, this simplifies the system drastically, it is now 3 dimensional; $\alpha_{|A\rangle}, \alpha_{|B\rangle}$ and ℓ. The output is then,

$$\left\langle \hat{l}_A \middle| o(\sigma) \right\rangle = \tanh\left(\ell\Big(\frac{\langle M|\sigma\rangle}{\tilde{T}}\Big)^n\right) = -\left\langle \hat{l}_B \middle| o(\sigma) \right\rangle, \tag{S.41}$$

$$\left\langle \hat{l}_\gamma \middle| o(\sigma) \right\rangle = -\tanh\left(\Big(\frac{\langle M|\sigma\rangle}{\tilde{T}}\Big)^n\right), \tag{S.42}$$

where $\langle M|\sigma\rangle$ can be written directly in terms of the α's,

$$\langle M|\sigma\rangle = \alpha_{|A\rangle}\langle A|\sigma\rangle + \alpha_{|B\rangle}\langle B|\sigma\rangle. \tag{S.43}$$

This leads to the following cost function,

$$C = \sum_{|\sigma\rangle\in\{|A\rangle,|B\rangle\}} \left(\left\langle \hat{l}_A \middle| t_{|\sigma\rangle} \right\rangle - \left\langle \hat{l}_A \middle| o(\sigma) \right\rangle\right)^{2n} + 4 \sum_{|\sigma\rangle\in\{|A\rangle,|B\rangle\}} \left(1 + \left\langle \hat{l}_\gamma \middle| o(\sigma) \right\rangle\right)^{2n}, \tag{S.44}$$

ignoring overall factors.

[3] These are the columns of our pseudo-inverse.

[4] Up to numerical artefacts for very small temperatures.

3.1.2 Memory normalization

Due to the nature of the MNIST dataset, any (non-identical) $|A\rangle$ and $|B\rangle$ will have two overlapping pixels with value -1, as well as two overlapping pixels of opposite signs with amplitude near 1. This means, for any (real) k_A, k_B we write,

$$\max_j \Big| k_A \langle \hat{m}_j | A \rangle + k_B \langle \hat{m}_j | B \rangle \Big| = |k_A| + |k_B|. \tag{S.45}$$

Simply put, for any linear combination of two (non-identical) training samples, the largest absolute pixel is approximately equal to the absolute sum of the linear coefficients of $|A\rangle$ and $|B\rangle$.

3.1.3 Gradient descent quantities

We define now the gradient descent quantities,

$$\nabla_{|A\rangle} = -\langle A' | \frac{\partial C}{\partial \langle M |}, \tag{S.46}$$

$$\nabla_{|B\rangle} = -\langle B' | \frac{\partial C}{\partial \langle M |}, \tag{S.47}$$

$$\nabla_{\ell} = -\Big\langle \hat{l}_A \Big| \frac{\partial C}{\partial \langle L |}, \tag{S.48}$$

$$\nabla_{\gamma} = -\Big\langle \hat{l}_\gamma \Big| \frac{\partial C}{\partial \langle L |}, \tag{S.49}$$

these are **scalar** quantities, see (S.16, S.17).[5]

3.1.4 α and ℓ dynamics

Using the above, with (S.39, S.40) and (S.45), we get

$$\alpha_{|A\rangle}(t+\delta t) = N(t)\Big(\alpha_{|A\rangle}(t) + \delta t \frac{\nabla_{|A\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}\Big), \tag{S.50}$$

$$\alpha_{|B\rangle}(t+\delta t) = N(t)\Big(\alpha_{|B\rangle}(t) + \delta t \frac{\nabla_{|B\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}\Big), \tag{S.51}$$

$$\ell(t+\delta t) = \mathcal{C}\Big(\ell(t) + \delta t \frac{\nabla_{\ell}}{\max(|\nabla_{\ell}|, |\nabla_{\gamma}|)}\Big), \tag{S.52}$$

where $N(t)$ can also be simplified with (S.45),

$$N^{-1}(t) = \max(1, \Big|\alpha_{|A\rangle}(t) + \delta t \frac{\nabla_{|A\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}\Big| + \Big|\alpha_{|B\rangle}(t) + \delta t \frac{\nabla_{|B\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}\Big|). \tag{S.53}$$

[5] The minus sign makes things more intuitive...

3.1.5 Gradient descent quantities (expanded)

Using (S.16, S.17) and (S.39, S.40) we can write,

$$\begin{aligned}\nabla_{|A\rangle} = &\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle^2\right)\left(\frac{\langle M|A\rangle}{\tilde{T}}\right)^{n-1}\ell \\ &+4\left(1+\left\langle \hat{l}_\gamma \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_\gamma \middle| o(A)\right\rangle^2\right)\left(\frac{\langle M|A\rangle}{\tilde{T}}\right)^{n-1},\end{aligned} \tag{S.54}$$

$$\begin{aligned}\nabla_{|B\rangle} = &-\left(1+\left\langle \hat{l}_A \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(B)\right\rangle^2\right)\left(\frac{\langle M|B\rangle}{\tilde{T}}\right)^{n-1}\ell \\ &+4\left(1+\left\langle \hat{l}_\gamma \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_\gamma \middle| o(B)\right\rangle^2\right)\left(\frac{\langle M|B\rangle}{\tilde{T}}\right)^{n-1},\end{aligned} \tag{S.55}$$

$$\begin{aligned}\nabla_{\ell} = &\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle^2\right)\left(\frac{\langle M|A\rangle}{\tilde{T}}\right)^{n} \\ &-\left(1+\left\langle \hat{l}_A \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(B)\right\rangle^2\right)\left(\frac{\langle M|B\rangle}{\tilde{T}}\right)^{n},\end{aligned} \tag{S.56}$$

$$\begin{aligned}\nabla_{\gamma} = &-\left(1+\left\langle \hat{l}_\gamma \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_\gamma \middle| o(A)\right\rangle^2\right)\left(\frac{\langle M|A\rangle}{\tilde{T}}\right)^{n} \\ &-\left(1+\left\langle \hat{l}_\gamma \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_\gamma \middle| o(B)\right\rangle^2\right)\left(\frac{\langle M|B\rangle}{\tilde{T}}\right)^{n}\end{aligned} \tag{S.57}$$

where time-dependence for α's, ℓ and $|M\rangle$ is implicit.

3.1.6 General behavior

Notice first that for positive ℓ, $\nabla_{|A\rangle}$ is the sum of two positive terms (1st and 2nd line of S.54). Hence to obtain $\nabla_{|A\rangle} = 0$ we require $\ell < 0$. Conversely $\nabla_{|B\rangle} = 0$ requires $\ell > 0$. Hence, $\nabla_{|A\rangle}, \nabla_{|B\rangle}$ cannot both be zero. Thus, normalization is necessary to get a fixed point.

Again from the gradient descent quantities, ℓ will tend to be positive when $\alpha_{|A\rangle} > \alpha_{|B\rangle}$ (conversely true), which pushes the system towards $|\alpha_{|A\rangle}| + |\alpha_{|B\rangle}| = 1$. (see Fig. S.21 and Fig. S.22)

3.1.7 Normalization condition

To reiterate, normalization of the memory only happens when a pixel/element of the memory has amplitude greater than 1, otherwise we say there is no normalization or trivial normalization (which means $N(t) = 1$). If we look closer at $N(t)$, the condition for (non-trivial)

normalization to occur is,

$$\left|\alpha_{|A\rangle}(t) + \delta t \frac{\nabla_{|A\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}\right| + \left|\alpha_{|B\rangle}(t) + \delta t \frac{\nabla_{|B\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}\right| > 1. \tag{S.58}$$

For very small δt we can write,

$$\left|\alpha_{|A\rangle}(t)\right| + \left|\alpha_{|B\rangle}(t)\right| + \frac{\left|\alpha_{|A\rangle}(t)\right|}{\alpha_{|A\rangle}(t)} \delta t \frac{\nabla_{|A\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|} + \frac{\left|\alpha_{|B\rangle}(t)\right|}{\alpha_{|B\rangle}(t)} \delta t \frac{\nabla_{|B\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|} > 1, \tag{S.59}$$

this is essentially a first order expansion of the absolute value, and works for small enough δt and non-zero α's. The special case where one of the α's is exactly zero is simple enough, omitted here for conciseness.
In the limit of $\delta t \to 0$, (S.59) is true when:

$$\left|\alpha_{|A\rangle}(t)\right| + \left|\alpha_{|B\rangle}(t)\right| = 1, \tag{S.60}$$

$$\frac{\left|\alpha_{|A\rangle}(t)\right|}{\alpha_{|A\rangle}(t)} \nabla_{|A\rangle} + \frac{\left|\alpha_{|B\rangle}(t)\right|}{\alpha_{|B\rangle}(t)} \nabla_{|B\rangle} > 0. \tag{S.61}$$

That is, for normalization to occur the memory must be at the (S.60) boundary and the dynamics must try push it 'outside' of the the boundary (S.61).

3.1.8 Continuous memory dynamics (no normalization)

If either of (S.60, S.61) are not held, then taking the $\lim_{\delta t \to 0}$ leads to

$$\frac{\partial \alpha_{|A\rangle}}{\partial t} = \frac{\nabla_{|A\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}, \tag{S.62}$$

$$\frac{\partial \alpha_{|B\rangle}}{\partial t} = \frac{\nabla_{|B\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|} \tag{S.63}$$

As outlined in 3.1.6, this offers no memory nullclines[6] as $\nabla_{|A\rangle}$ and $\nabla_{|B\rangle}$ cannot both be zero.

3.1.9 Continuous memory dynamics (with normalization)

If both (S.60, S.61) are held, we can write

$$N(t) \approx 1 - \delta t \frac{|\alpha_{|A\rangle}(t)|}{\alpha_{|A\rangle}(t)} \frac{\nabla_{|A\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|} - \delta t \frac{|\alpha_{|B\rangle}(t)|}{\alpha_{|A\rangle}(t)} \frac{\nabla_{|B\rangle}}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|} \tag{S.64}$$

[6] A memory nullcline means $\partial_t |M\rangle = 0$, which requires $\frac{\partial \alpha_{|A\rangle}}{\partial t} = \frac{\partial \alpha_{|B\rangle}}{\partial t} = 0$.

Applying this to the dynamics, keeping only first order δt and taking the limit we get,

$$\frac{\partial \alpha_{|A\rangle}}{\partial t} = \frac{\nabla_{|A\rangle} - \alpha_{|A\rangle}(t)\Big(\frac{|\alpha_{|A\rangle}(t)|}{\alpha_{|A\rangle}(t)}\nabla_{|A\rangle} + \frac{|\alpha_{|B\rangle}(t)|}{\alpha_{|B\rangle}(t)}\nabla_{|B\rangle}\Big)}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}, \tag{S.65}$$

$$\frac{\partial \alpha_{|B\rangle}}{\partial t} = \frac{\nabla_{|B\rangle} - \alpha_{|B\rangle}(t)\Big(\frac{|\alpha_{|A\rangle}(t)|}{\alpha_{|A\rangle}(t)}\nabla_{|A\rangle} + \frac{|\alpha_{|B\rangle}(t)|}{\alpha_{|B\rangle}(t)}\nabla_{|B\rangle}\Big)}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}. \tag{S.66}$$

This leads to points/curves where $\frac{\partial \alpha_{|A\rangle}}{\partial t} = \frac{\partial \alpha_{|B\rangle}}{\partial t} = 0$, i.e. memory nullclines.

Also while the normalization conditions are held, our memories are effectively one-dimensional and can be described by

$$\alpha_{|A\rangle} = \alpha, \tag{S.67}$$

$$\alpha_{|B\rangle} = \pm\Big(1 - |\alpha|\Big), \tag{S.68}$$

where $-1 \leq \alpha \leq 1$. Essentially, we have two one-dimensional branches for the memory, one where $\alpha_{|B\rangle}$ is positive and the other for negative $\alpha_{|B\rangle}$. Generally we need to examine these two branches independently for fixed points, but for intermediate and high n all fixed points are held in the positive branch. The above should also give intuition as to why $\frac{\partial \alpha_{|A\rangle}}{\partial t} = 0 \iff \frac{\partial \alpha_{|B\rangle}}{\partial t} = 0$.

3.1.10 Continuous label dynamics

The same can be done for labels, for $\mathcal{C}(\cdot)$ to be non-trivial we require

$$|\ell| = 1, \tag{S.69}$$

$$\text{sign}(\nabla_\ell) = \text{sign}(\ell). \tag{S.70}$$

which leads to,

$$\frac{\partial \ell}{\partial t} = 0, \tag{S.71}$$

a label nullcline, when the label is at its boundary (S.69) and the dynamics are trying to push it outside (S.70). More generally, if either of (S.69 or S.70) is not satisfied (i.e. no clipping required) then,

$$\frac{\partial \ell}{\partial t} = \frac{\nabla_\ell}{\max(|\nabla_\ell|, |\nabla_\gamma|)}, \tag{S.72}$$

which leads to nullclines as well.

3.1.11 Memory nullclines

When (S.60, S.61) are held, $\partial_t \alpha_{|A\rangle} = \partial_t \alpha_{|B\rangle} = 0$ means

$$0 = \frac{\nabla_{|A\rangle} - \alpha_{|A\rangle}(t)\left(\frac{|\alpha_{|A\rangle}(t)|}{\alpha_{|A\rangle}(t)}\nabla_{|A\rangle} + \frac{|\alpha_{|B\rangle}(t)|}{\alpha_{|B\rangle}(t)}\nabla_{|B\rangle}\right)}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}, \tag{S.73}$$

$$0 = \frac{\nabla_{|B\rangle} - \alpha_{|B\rangle}(t)\left(\frac{|\alpha_{|A\rangle}(t)|}{\alpha_{|A\rangle}(t)}\nabla_{|A\rangle} + \frac{|\alpha_{|B\rangle}(t)|}{\alpha_{|B\rangle}(t)}\nabla_{|B\rangle}\right)}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}. \tag{S.74}$$

which are redundant - if one of the above is zero, the other must be zero. Either equation can be rearranged into,

$$0 = \frac{\nabla_{|A\rangle}}{\alpha_{|A\rangle}} - \frac{\nabla_{|B\rangle}}{\alpha_{|B\rangle}}, \tag{S.75}$$

or,

$$\boxed{\frac{\nabla_{|A\rangle}}{\alpha_{|A\rangle}} = \frac{\nabla_{|B\rangle}}{\alpha_{|B\rangle}}.} \tag{S.76}$$

3.1.12 Label nullclines

Label nullclines come either from (S.69, S.70) being both satisfied, or from

$$\boxed{\nabla_\ell = 0.} \tag{S.77}$$

3.1.13 The intersection

The boundary condition (S.60) makes the space essentially 2D (one α dimension and one ℓ). The memory and label nullclines each define one dimensional curves and hence intersect at at one or many points; these are fixed points of the dynamics. This is a numerically solvable system of 2 equations.

Generally, the $|\alpha_{|A\rangle}| + |\alpha_{|B\rangle}| = 1$ boundary means we have to look at two branches, one where $\alpha_{|B\rangle} = 1 - |\alpha_{|A\rangle}|$ and another where $\alpha_{|B\rangle} = -1 + |\alpha_{|A\rangle}|$. However for intermediate and high-n, only the positive branch matters, moreover $\alpha_{|A\rangle} > 0$. We illustrate this simpler case below.

3.1.14 A simple case

In this case,

$$\alpha_{|A\rangle} = \alpha, \tag{S.78}$$

$$\alpha_{|B\rangle} = 1 - \alpha, \tag{S.79}$$

for $0 \leq \alpha \leq 1$. We can use (S.65) to obtain

$$\frac{\partial \alpha}{\partial t} = \frac{\nabla_{|A\rangle} - \alpha\Big(\nabla_{|A\rangle} + \nabla_{|B\rangle}\Big)}{|\nabla_{|A\rangle}| + |\nabla_{|B\rangle}|}, \tag{S.80}$$

while the label equation keeps the form of (S.72),

$$\frac{\partial \ell}{\partial t} = \frac{\nabla_\ell}{\max(|\nabla_\ell|, |\nabla_\gamma|)}. \tag{S.81}$$

The normalization condition on the memory (S.61) becomes much simpler,

$$\nabla_{|A\rangle} + \nabla_{|B\rangle} > 0$$

which is always held on the memory nullcline ($\partial_t \alpha = 0$), which reduces to

$$\boxed{\frac{\nabla_{|A\rangle}}{\alpha} = \frac{\nabla_{|B\rangle}}{1-\alpha}.} \tag{S.82}$$

The label nullcline keeps the same form,

$$\boxed{\nabla_\ell = 0.} \tag{S.83}$$

These equations both depend (implicitly) on α and ℓ. For examples of memory and label nullclines, see Fig. S.27.

3.2 The purely numerical case : N_k digits

A similar theoretical framework is more difficult to achieve for larger systems, nonetheless the first saddle(s) can still be obtained in a computationally efficient way, by simulating a 1-memory system, with the same training set and with an effective temperature

$$\tilde{T} = \frac{T}{\sqrt[n]{N_k}}. \tag{S.84}$$

This can be done for Gaussian initial conditions, where the primary saddle will be found, or with various initial condition (using α coefficients) where other saddles will be found.

4 Splitting / The 2-memory system

To describe the splitting dynamics, we write for the labels $|L_+\rangle, |L_-\rangle$,

$$\langle l_A | L_+ \rangle = \ell + \delta\ell = -\langle l_B | L_+ \rangle, \tag{S.85}$$
$$\langle l_A | L_- \rangle = \ell - \delta\ell = -\langle l_B | L_- \rangle, \tag{S.86}$$
$$\langle l_\gamma | L_\pm \rangle = -1 \tag{S.87}$$

the only assumption here is that we are near the ℓ-manifold described in the 1-memory system (section 3). We can do a similar expansion for the memories, which generally are defined as

$$|M_\pm\rangle = \alpha_{|A\rangle,\pm}\,|A\rangle + \alpha_{|B\rangle,\pm}\,|B\rangle\,.$$

For simplicity assume $\alpha_{|A\rangle,\pm}, \alpha_{|B\rangle,\pm} > 0$, and $\alpha_{|A\rangle,\pm} + \alpha_{|B\rangle,\pm} = 1$ - this holds if we are near a (positive) 1-memory fixed point. Then, we can write

$$|M_\pm\rangle = (\alpha \pm \delta\alpha)\,|A\rangle + (1 - \alpha \mp \delta\alpha)\,|B\rangle\,. \tag{S.88}$$

Up to first order in $\delta\alpha$ and $\delta\ell$, the output is equal to the 1-memory case,

$$|o(\sigma)\rangle \approx \tanh\left(|L\rangle\left(\frac{\langle M|\sigma\rangle}{\tilde{T}}\right)^n\right), \tag{S.89}$$

with

$$\tilde{T} = \frac{T}{\sqrt[n]{2}}, \tag{S.90}$$

and

$$|M\rangle = |M_\pm\rangle\Big|_{\delta\alpha=0}, \tag{S.91}$$

$$|L\rangle = |L_\pm\rangle\Big|_{\delta\ell=0}. \tag{S.92}$$

For the δ-quantities we first write,

$$\langle M_\pm|\sigma\rangle^n \approx \langle M|\sigma\rangle^n \pm n\,\langle M|\sigma\rangle^{n-1}\,\frac{\partial\,\langle M|\sigma\rangle}{\partial\alpha}\delta\alpha, \tag{S.93}$$

where

$$\frac{\partial\,\langle M|A\rangle}{\partial\alpha} = \langle A|A\rangle - \langle A|B\rangle\,, \tag{S.94}$$

$$\frac{\partial\,\langle M|B\rangle}{\partial\alpha} = \langle A|B\rangle - \langle B|B\rangle\,. \tag{S.95}$$

This leads to the following gradient descent quantities,

$$\begin{aligned}
\nabla_{|A\rangle,\pm} \approx & \left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle^2\right)\langle M|A\rangle^{n-1}\,\ell \\
& + 4\left(1+\left\langle \hat{l}_\gamma \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_\gamma \middle| o(A)\right\rangle^2\right)\langle M|A\rangle^{n-1} \\
& \pm (n-1)\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle^2\right)\langle M|A\rangle^{n-2}\,\frac{\partial\langle M|A\rangle}{\partial\alpha}\ell\delta\alpha \\
& \pm 4(n-1)\left(1+\left\langle \hat{l}_\gamma \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_\gamma \middle| o(A)\right\rangle^2\right)\langle M|A\rangle^{n-2}\,\frac{\partial\langle M|A\rangle}{\partial\alpha}\delta\alpha \\
& \pm \left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle^2\right)\langle M|A\rangle^{n-1}\,\delta\ell
\end{aligned} \tag{S.96}$$

$$\begin{aligned}
\nabla_{|B\rangle,\pm} \approx & -\left(1+\left\langle \hat{l}_A \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(B)\right\rangle^2\right)\langle M|B\rangle^{n-1}\,\ell \\
& + 4\left(1+\left\langle \hat{l}_\gamma \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_\gamma \middle| o(B)\right\rangle^2\right)\langle M|B\rangle^{n-1} \\
& \mp (n-1)\left(1+\left\langle \hat{l}_A \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(B)\right\rangle^2\right)\langle M|B\rangle^{n-2}\,\frac{\partial\langle M|B\rangle}{\partial\alpha}\ell\delta\alpha \\
& \pm 4(n-1)\left(1+\left\langle \hat{l}_\gamma \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_\gamma \middle| o(B)\right\rangle^2\right)\langle M|B\rangle^{n-2}\,\frac{\partial\langle M|B\rangle}{\partial\alpha}\delta\alpha \\
& \mp \left(1+\left\langle \hat{l}_A \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(B)\right\rangle^2\right)\langle M|B\rangle^{n-1}\,\delta\ell
\end{aligned} \tag{S.97}$$

$$\begin{aligned}
\nabla_{\ell,\pm} \approx & \left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle^2\right)\langle M|A\rangle^{n} \\
& -\left(1+\left\langle \hat{l}_A \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(B)\right\rangle^2\right)\langle M|B\rangle^{n} \\
& \pm n\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(A)\right\rangle^2\right)\langle M|A\rangle^{n-1}\,\frac{\partial\langle M|A\rangle}{\partial\alpha}\delta\alpha \\
& \mp n\left(1+\left\langle \hat{l}_A \middle| o(B)\right\rangle\right)^{2n-1}\left(1-\left\langle \hat{l}_A \middle| o(B)\right\rangle^2\right)\langle M|B\rangle^{n-1}\,\frac{\partial\langle M|B\rangle}{\partial\alpha}\delta\alpha
\end{aligned} \tag{S.98}$$

keeping only the first order in δ. The normalization condition becomes

$$\nabla_{|A\rangle,\pm} + \nabla_{|B\rangle,\pm} > 0, \tag{S.99}$$

for our initial assumptions to hold, this inequality must be held. Then,

$$\frac{\partial(\alpha \pm \delta\alpha)}{\partial t} \approx \frac{\nabla_{|A\rangle,\pm} - \Big(\alpha \pm \delta\alpha\Big)\Big(\nabla_{|A\rangle,\pm} + \nabla_{|B\rangle,\pm}\Big)}{|\nabla_{|A\rangle,\pm}| + |\nabla_{|B\rangle,\pm}|}, \tag{S.100}$$

again keeping only first order terms,

$$\frac{\partial(\alpha)}{\partial t} \pm \frac{\partial(\delta\alpha)}{\partial t} \approx \frac{\nabla_{|A\rangle,\pm} - \alpha\Big(\nabla_{|A\rangle,\pm} + \nabla_{|B\rangle,\pm}\Big) \mp \delta\alpha\Big(\nabla_{|A\rangle} + \nabla_{|B\rangle}\Big)}{|\nabla_{|A\rangle,\pm}| + |\nabla_{|B\rangle,\pm}|}. \tag{S.101}$$

Where $\nabla_{|A\rangle}, \nabla_{|B\rangle}$ represent the 1-memory quantities; equivalent to setting $\delta\alpha = \delta\ell = 0$ in $\nabla_{|\sigma\rangle,\pm}$. From the above one can recover $\partial_t\alpha$ and $\partial_t\delta\alpha$. As for ℓ, we assume $\ell < 1$ and obtain,

$$\frac{\partial(\ell)}{\partial t} \pm \frac{\partial(\delta\ell)}{\partial t} \approx \frac{\nabla_{\ell,\pm}}{\max(|\nabla_{\ell,\pm}|, |\nabla_{\gamma}|)}. \tag{S.102}$$

Further approximations are available (notably for the denominators). This first-order approximation works particularly well near a 1-memory fixed point, as shown in Fig. S.29, and Fig. S.30.

5 Final states / The 2-memory system

The 2-memory system can also be used to understand the final states. Consider a system which has already split,

$$\Big\langle \hat{l}_A \Big| L_A \Big\rangle = 1 = \Big\langle \hat{l}_B \Big| L_B \Big\rangle, \tag{S.103}$$

$$\Big\langle \hat{l}_B \Big| L_A \Big\rangle = \Big\langle \hat{l}_A \Big| L_B \Big\rangle = -1 = \langle l_\gamma | L_A \rangle = \langle l_\gamma | L_B \rangle. \tag{S.104}$$

As for the memory,

$$|M_A\rangle = \Big(1 - |\delta_A|\Big)\,|A\rangle + \delta_A\,|B\rangle, \tag{S.105}$$

$$|M_B\rangle = \delta_B\,|A\rangle + \Big(1 - |\delta_B|\Big)\,|B\rangle. \tag{S.106}$$

The output reduces to,

$$\Big\langle \hat{l}_A \Big| o(\sigma) \Big\rangle = \tanh\left(\Big(\frac{\langle M_A|\sigma\rangle}{T}\Big)^n - \Big(\frac{\langle M_B|\sigma\rangle}{T}\Big)^n\right) = -\Big\langle \hat{l}_B \Big| o(\sigma) \Big\rangle, \tag{S.107}$$

$$\Big\langle \hat{l}_\gamma \Big| o(\sigma) \Big\rangle = \tanh\left(-\Big(\frac{\langle M_A|\sigma\rangle}{T}\Big)^n - \Big(\frac{\langle M_B|\sigma\rangle}{T}\Big)^n\right) \tag{S.108}$$

and the cost function becomes,

$$C = \left(1 - \left\langle \hat{l}_A \middle| o(A) \right\rangle\right)^{2n} + 4\left(1 + \left\langle \hat{l}_\gamma \middle| o(A) \right\rangle\right)^{2n} + \left(1 + \left\langle \hat{l}_A \middle| o(B) \right\rangle\right)^{2n} + 4\left(1 + \left\langle \hat{l}_\gamma \middle| o(B) \right\rangle\right)^{2n}. \tag{S.109}$$

As before, we define

$$\nabla_{|\sigma\rangle,\mu} = -\langle \sigma' | \frac{\partial C}{\partial \langle M_\mu |}, \tag{S.110}$$

$$\langle \sigma' | M_\mu \rangle = \alpha_{\sigma,\mu} \tag{S.111}$$

and we can write,

$$\partial_t \alpha_{\sigma,\mu} = \frac{\nabla_{|\sigma\rangle,\mu} - \alpha_{\sigma,\mu}\left(\frac{|\alpha_{A,\mu}|}{\alpha_{A,\mu}}\nabla_{|A\rangle,\mu} + \frac{|\alpha_{B,\mu}|}{\alpha_{B,\mu}}\nabla_{|B\rangle,\mu}\right)}{|\nabla_{|A\rangle,\mu}| + |\nabla_{|B\rangle,\mu}|}, \tag{S.112}$$

which leads to the following fixed point,

$$\frac{\alpha_{A,\mu}}{\nabla_{|A\rangle,\mu}} = \frac{\alpha_{B,\mu}}{\nabla_{|B\rangle,\mu}}. \tag{S.113}$$

If we combine this with our initial assumption, we get the following system of equation

$$\begin{cases} \frac{1-|\delta_A|}{\nabla_{|A\rangle,A}} = \frac{\delta_A}{\nabla_{|B\rangle,A}} \\ \frac{\delta_B}{\nabla_{|A\rangle,B}} = \frac{1-|\delta_B|}{\nabla_{|B\rangle,B}} \end{cases}. \tag{S.114}$$

Two variables, two equations; numerically solvable.

6 Other models

6.1 The intraclass feature-to-prototype transition

Consider a 1-memory system with two training samples $|A_1\rangle, |A_2\rangle$ of the same class A. The label can be written as

$$\left\langle \hat{l}_A \middle| L \right\rangle = 1 = -\left\langle \hat{l}_\gamma \middle| L \right\rangle, \tag{S.115}$$

for any $\gamma \neq A$. For the memory, we define two strategies the system may implement, the generalist and the specialist.

$$|M_g\rangle = \frac{|A_1\rangle + |A_2\rangle}{2}, \tag{S.116}$$

$$|M_s\rangle = |A_1\rangle \tag{S.117}$$

where the choice of either $|A_1\rangle$ or $|A_2\rangle$ in the specialist strategy depends strongly on the initial conditions. For simplicity, the generalist case is written for $\langle A_1|A_1\rangle = \langle A_2|A_2\rangle$.
The output of each strategy is,

$$\left\langle \hat{l}_A \middle| o(\sigma) \right\rangle_g = \tanh\left(\left(\frac{\langle A_1|\sigma\rangle + \langle A_2|\sigma\rangle}{2T} \right)^n \right) = -\left\langle \hat{l}_\gamma \middle| o(\sigma) \right\rangle_g , \tag{S.118}$$

$$\left\langle \hat{l}_A \middle| o(\sigma) \right\rangle_s = \tanh\left(\left(\frac{\langle A_1|\sigma\rangle}{T} \right)^n \right) = -\left\langle \hat{l}_\gamma \middle| o(\sigma) \right\rangle_s . \tag{S.119}$$

The cost function of a 1-memory system (with labels as defined previously) is,

$$C = 10\left(1 - \left\langle \hat{l}_A \middle| o(A_1) \right\rangle\right)^{2n} + 10\left(1 - \left\langle \hat{l}_A \middle| o(A_2) \right\rangle\right)^{2n}, \tag{S.120}$$

where the factors of 10, have been kept to hint at the symmetry involved. For each strategy we write,

$$C_g = 2\left(1 - \tanh\left(\left(\frac{\langle A_1|A_1\rangle + \langle A_1|A_2\rangle}{2T} \right)^n \right)\right)^{2n}, \tag{S.121}$$

$$C_s = \left(1 - \tanh\left(\left(\frac{\langle A_1|A_1\rangle}{T} \right)^n \right)\right)^{2n} + \left(1 - \tanh\left(\left(\frac{\langle A_1|A_2\rangle}{T} \right)^n \right)\right)^{2n}, \tag{S.122}$$

where the assumption $\langle A_1|A_1\rangle = \langle A_2|A_2\rangle$ was used to simplify C_g and overall (10) factors are removed. Now we approximate that in the region where C_s is valid, $C_s \approx 1$. [7] We say the curve where both strategies lead to the same cost/energy is the transition curve,

$$C_g = C_s \approx 1. \tag{S.123}$$

This leads to,

$$T \approx \frac{\langle A_1|A_1\rangle + \langle A_1|A_2\rangle}{2\sqrt[n]{\operatorname{arctanh}\left(1 - \frac{1}{2^n\sqrt{2}}\right)}}. \tag{S.124}$$

6.2 Populations and final states

Consider a system of N_k memories, with two training samples $|A\rangle, |B\rangle$ of classes A, B. Such systems tend to have only two unique memories, $|M_A\rangle$ and $|M_B\rangle$, with other memories being

[7] Explicitly, $\left(1 - \tanh\left(\left(\frac{\langle A_1|A_1\rangle}{T} \right)^n \right)\right)^{2n} \approx 0$ and, $\left(1 - \tanh\left(\left(\frac{\langle A_1|A_2\rangle}{T} \right)^n \right)\right)^{2n} \approx 1$; only one training sample is understood.

clones of the latter.[8] In which case, the output can be written as,

$$\left\langle \hat{l}_A \middle| o(\sigma) \right\rangle = \tanh\left(N_A \Big(\frac{\langle M_A|\sigma\rangle}{T} \Big)^n - N_B \Big(\frac{\langle M_B|\sigma\rangle}{T} \Big)^n \right) = -\left\langle \hat{l}_B \middle| o(\sigma) \right\rangle, \tag{S.125}$$

$$\left\langle \hat{l}_\gamma \middle| o(\sigma) \right\rangle = -\tanh\left(N_A \Big(\frac{\langle M_A|\sigma\rangle}{T} \Big)^n + N_B \Big(\frac{\langle M_B|\sigma\rangle}{T} \Big)^n \right). \tag{S.126}$$

with of course,

$$N_A + N_B = N_k. \tag{S.127}$$

The cost function of the system is,

$$C = \left(1 - \left\langle \hat{l}_A \middle| o(A) \right\rangle \right)^{2n} + \left(1 + \left\langle \hat{l}_A \middle| o(B) \right\rangle \right)^{2n} \tag{S.128}$$

where $\left\langle \hat{l}_\gamma \middle| o(\sigma) \right\rangle \approx -1$ for $\sigma \in \{A, B\}$, which eliminated a few terms. There is a competition between the two classes, and we make a symmetry argument that the system will converge towards,

$$\left(1 - \left\langle \hat{l}_A \middle| o(A) \right\rangle \right)^{2n} = \left(1 + \left\langle \hat{l}_A \middle| o(B) \right\rangle \right)^{2n}. \tag{S.129}$$

This is equivalent to writing,

$$N_A \langle M_A|A\rangle^n - N_B \langle M_B|A\rangle^n = N_B \langle M_B|B\rangle^n - N_A \langle M_A|B\rangle^n, \tag{S.130}$$

which directly links the final $|M_A\rangle, |M_B\rangle$ states to the population proportions;

$$N_A = (N_k) \frac{\langle M_B|B\rangle^n + \langle M_B|A\rangle^n}{\langle M_A|A\rangle^n + \langle M_A|B\rangle^n + \langle M_B|B\rangle^n + \langle M_B|A\rangle^n}. \tag{S.131}$$

Knowing the final states of $|M_A\rangle$ and $|M_B\rangle$ tells you the population proportions.

7 Methods for Figures

7.1 Figure 2A: Memory reconstruction

In Figure 2A (right), a plot is shown describing the number of training samples required to reconstruct a memory.

[8] This can be dealt with in a more nuanced way with average $\left|\bar{M}_A\right\rangle$ and neglecting deviations from the average.

For each memory, the α's are obtained using the Moore-Penrose inverse. Such that,

$$|M_\mu\rangle = \sum \alpha_{\mu,|\sigma\rangle} |\sigma\rangle$$

The sum is reorder by decreasing absolute α coefficients (i.e. largest absolute coefficients are first). The number of reconstruction samples is defined as the lowest k-value at which the partial (ordered) sum of the first k terms is within some tolerance value of the original memory.

$$|M_\mu\rangle_k = \sum_{r=0}^{k} \alpha_{\mu,|\sigma_r\rangle} |\sigma_r\rangle$$

$$\sum_j \Big| \langle \hat{p}_j | M_\mu \rangle_k - \langle \hat{p}_j | M_\mu \rangle \Big| \leq 0.05 \sum_j \Big| \langle \hat{p}_j | M_\mu \rangle \Big|$$

To avoid tolerances changing too much over individual memories of the same network, the RHS of the above was averaged over all μ's of a same network (i.e. all memories of a same network have the same tolerance).

7.2 Figures 2 and 3: UMAP Trajectories

The UMAP embedding is generated using all 60000 training images of the MNIST training dataset. The correlation metric is used, as well as the following hyperparameters: random state (4), number of neighbours (55), minimum distance (0.05). The resulting UMAP object is saved and reused for Figs. 2B and 3A, E. Note, we see similar dynamics with the default UMAP metric, and other hyperparameter values.

When projecting the memory dynamics onto the UMAP landscape, the previously defined object is used and memories of the same epoch are projected together.

In movies, it can be difficult to track the trajectories of individual memories/points, to counter this we add linear interpolation frames, to make the trajectories easier to follow. Note however, that this has no effect on the general dynamics and is mainly a cosmetic modification.

7.3 Figure 3: Transition matrix

First, using the UMAP of Figure 3A, E the nearest (euclideean) training sample of each memory for each epoch is calculated. Then, a transition matrix ($\mathbf{M}$) is initialized as an empty 10×10 matrix. Every time a memory's nearest training sample changes from class $j \to i$ (i.e. a memory moves from being near a training sample of class j to being near of one of class i), 1 is added to element M_{ij} and subtracted from element M_{ji}. Once all memories and epochs are counted, the resulting matrix is renormalized so that its (absolute) largest entry is 1. Note that each n-case is treated separately.

7.4 Figure 4: UMAP Trees

For both $n = 3, 30$, a 400 memory system was trained on a training set of 1000 digits (100 per class). The initial conditions were random gaussian, with mean -0.003 and standard deviation 10^{-5}. A learning rate of 0.005 was chosen, with no 'momentum'. The T_r parameter was fixed to 0.85.

The UMAP object was initialized with a number of neighbors equal to 400 for $n = 3$ and 1000 for $n = 30$, while other parameters were left at their default values. For each n case, fitting and transforming the data was done separately.

For $n = 3$, the data fed into UMAP consisted of the α coefficients (obtained using the Moore-Penrose inverse, see Section 2) for every memory between epoch 100 to 1000 of training, with a time resolution of 10 (i.e. every 10 epoch is kept).

For $n = 30$, the data fed into UMAP consisted of the α coefficients for every memory between epoch 400 and epoch 2300 of training, with a time resolution of 5.

Once fitted, the same data was transformed onto the 2D UMAP space resulting in the plots shown in Fig. 4. Each data point represents a specific memory μ at a time t, and is colored based on the largest label element,

$$\max_d \left\langle \hat{l}_d \middle| L_\mu(t) \right\rangle , \tag{S.132}$$

following the same color scheme as previous figures.

7.5 Figure 6B: Energy Landscape

The general energy landscape is given by Eq (S.11) and is applied to a two memory system with two training samples. The memories and labels are parameterized as described in section 4, meaning the system is effectively 4-dimensional $\alpha, \ell, \delta\alpha, \delta\ell$ with the last two quantities describing the split. This means we have $C(\alpha, \ell, \delta\alpha, \delta\ell)$.

Before the split, the dynamics of the two memories often lie on the 1d curve defined by the label nullclines (see 6B, left). This allows us to parameterize $\ell(\alpha)$ using the label nullcline curve from section 3.1.12. Post split, we notice that $\delta\ell \sim 2.6\delta\alpha$, we use a polynomial fit to parameterize the typical splitting trajectory $\delta\ell(\delta\alpha)$. This leaves us with effectively a 2d energy landscape: $C(\alpha, \delta\alpha)$.

Note, the energy landscape is not very sensitive to the parameterisation choice for $\delta\ell$, as a linear definition leads to similar results.

8 Supplementary Figures

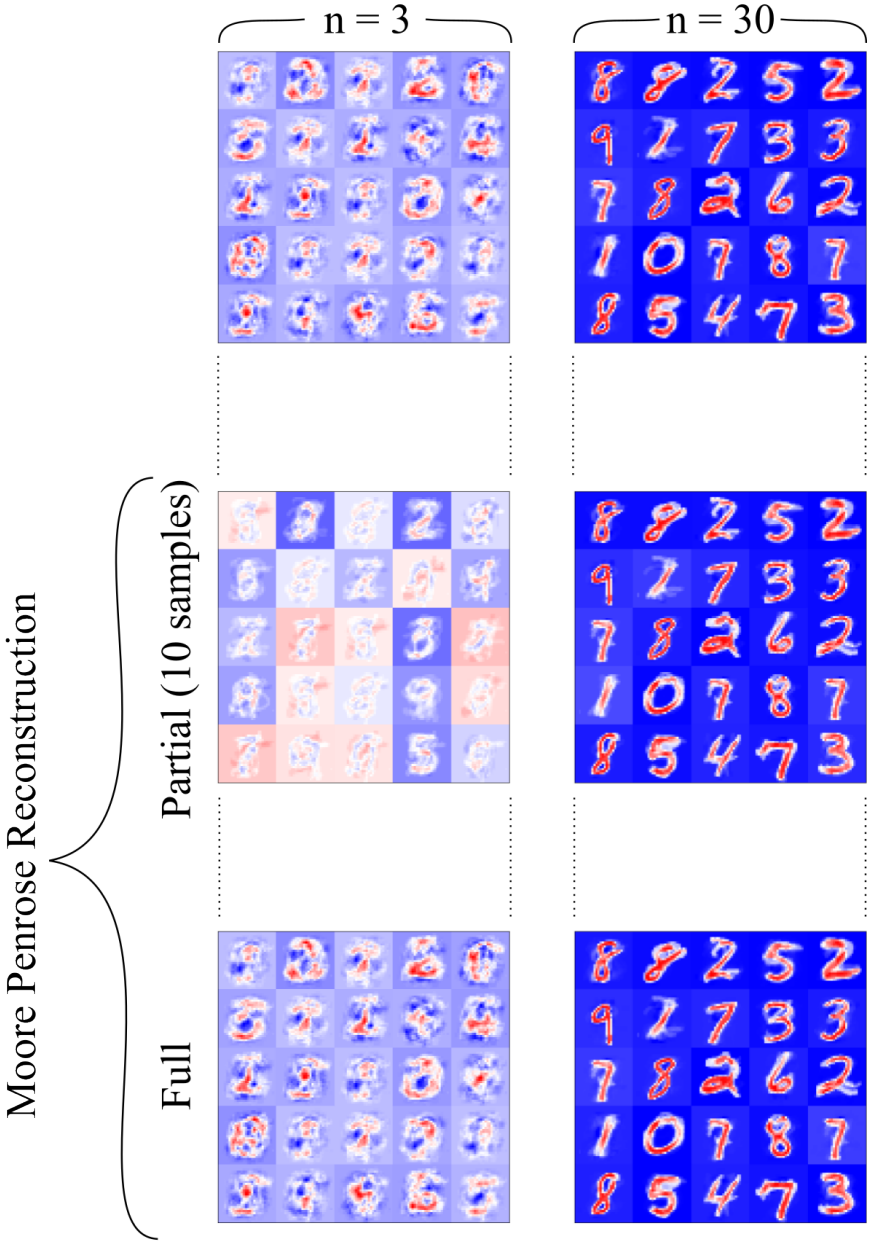

Figure S.1: In the first row, sample memories from a 100-memory system training on 200 digits (20 for each class) as in Fig. 3. The rescaled temperature is 0.85, and $n = 3$ or $n = 30$. The second and third rows are Moore-Penrose based memory reconstructions. The second row includes only the 10 most dominating coefficients, the third row includes all coefficients.

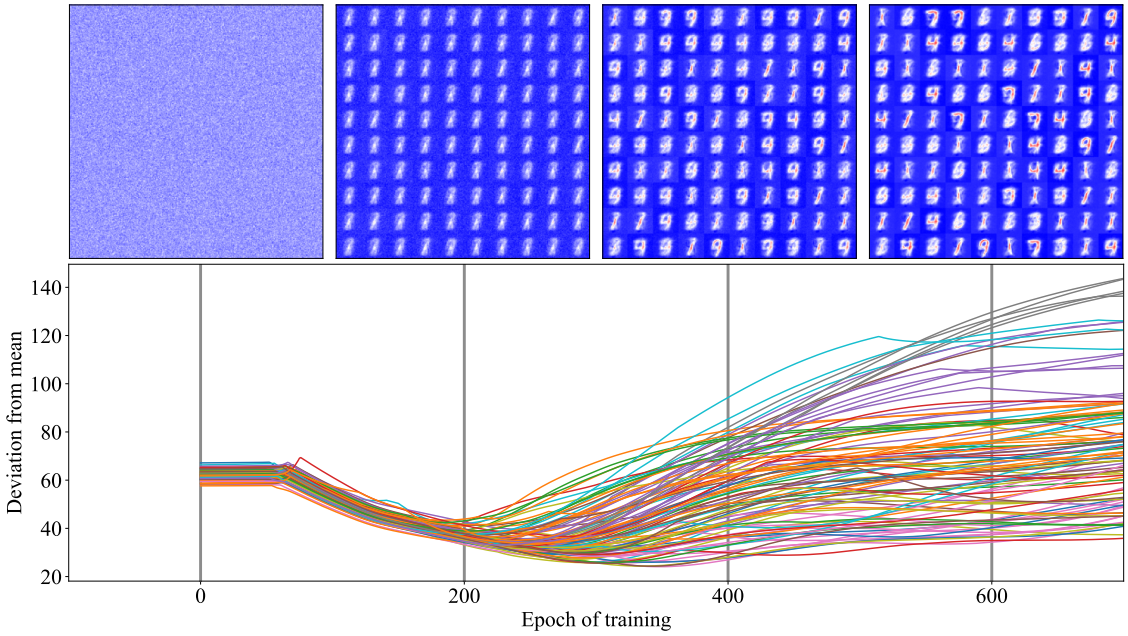

Figure S.2: A 100-memory system training on 200 digits (20 samples of each class) as in Fig. 3, the hyperparameters are $n = 30$, and $T_r = 0.85$. At each time point, the deviation between each memory and the average memory is shown, this highlights how memories converge towards a single 'saddle' before splitting.

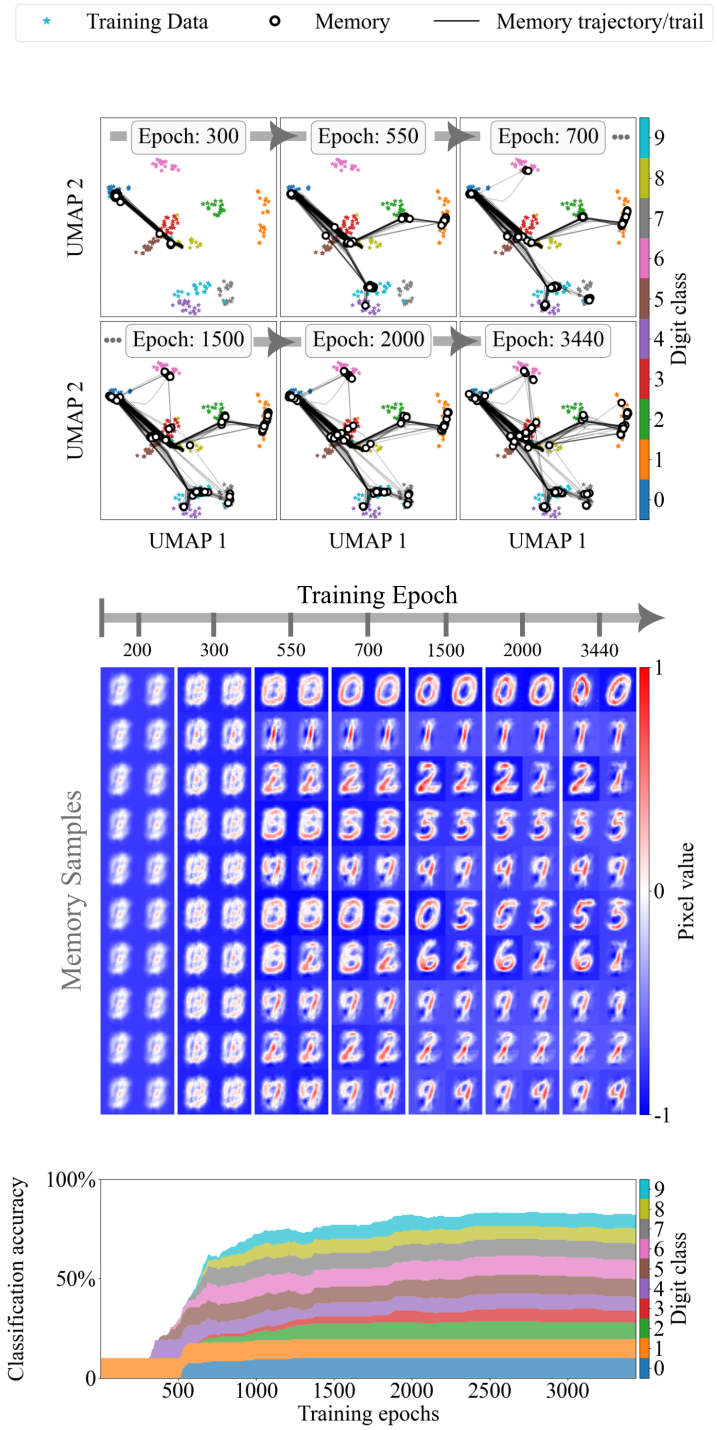

Figure S.3: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 15$, and $T_r = 0.85$.

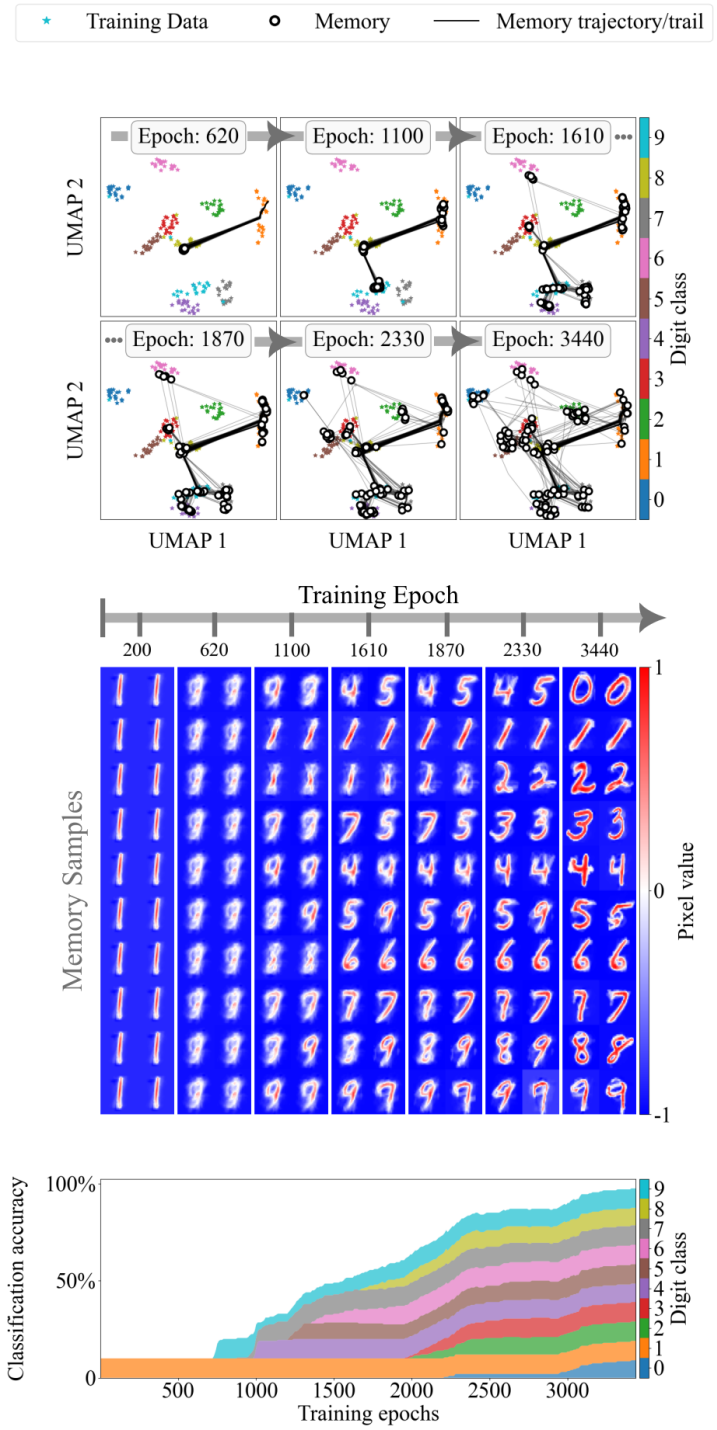

Figure S.4: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 40$, and $T_r = 0.85$.

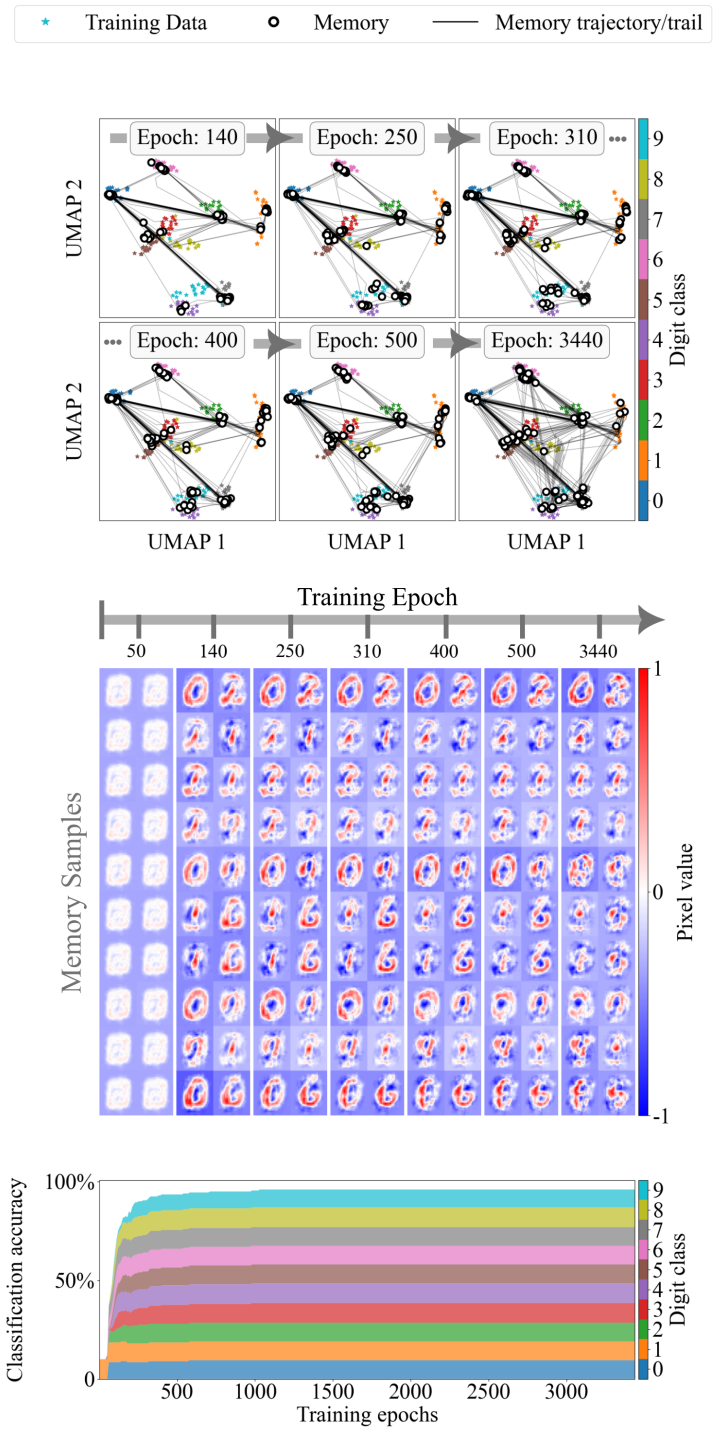

Figure S.5: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 3$, and $T_r = 0.85$, here however, the 200 digit training set is split into 4 miniBatchs each containing 50 digits (5 of each class).

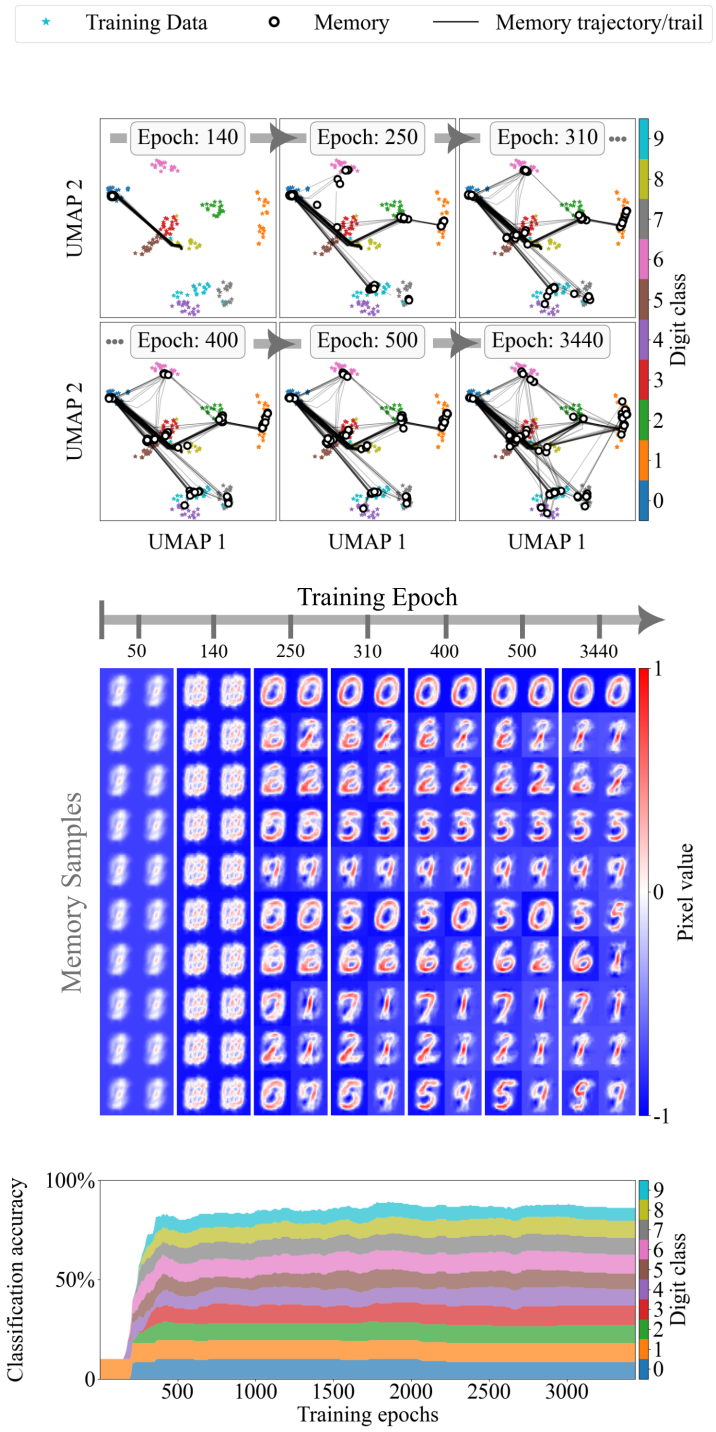

Figure S.6: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 15$, and $T_r = 0.85$, here however, the 200 digit training set is split into 4 miniBatchs each containing 50 digits (5 of each class).

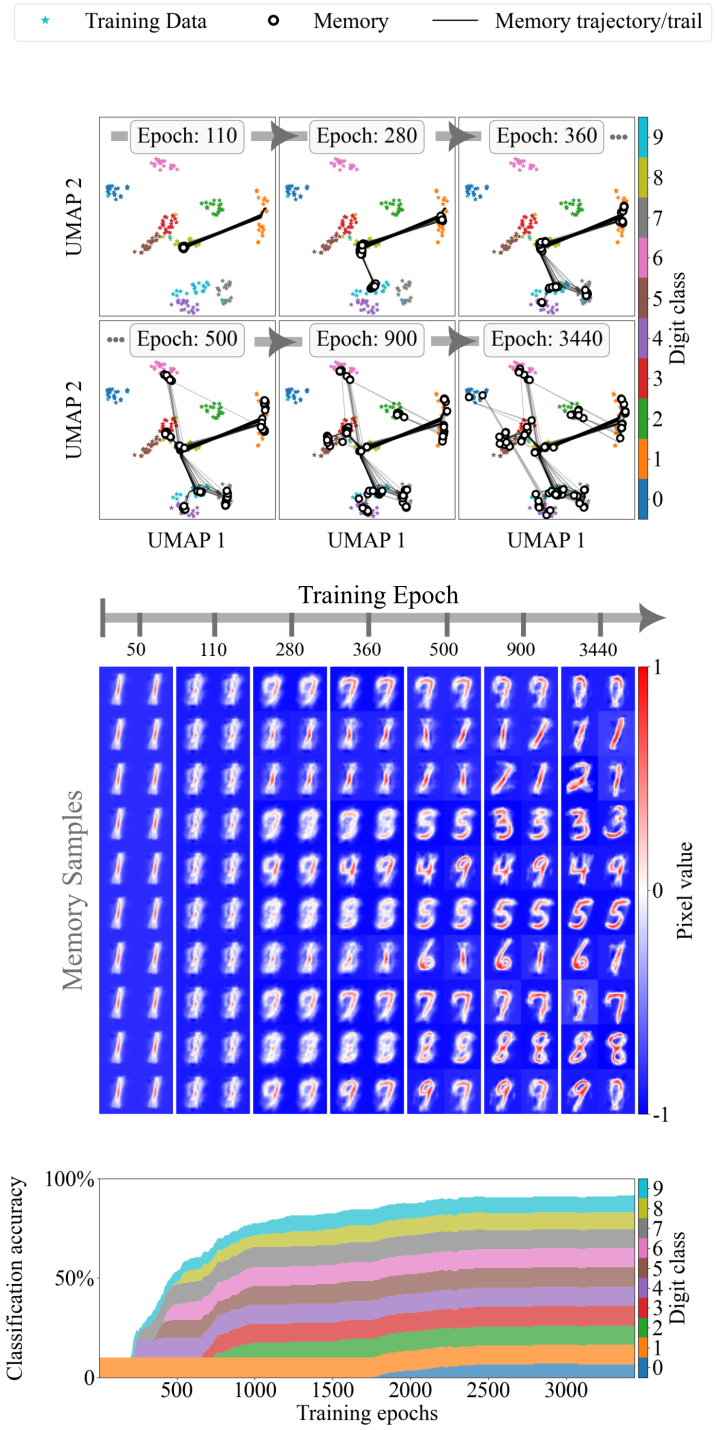

Figure S.7: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 30$, and $T_r = 0.85$, here however, the 200 digit training set is split into 4 miniBatchs each containing 50 digits (5 of each class).

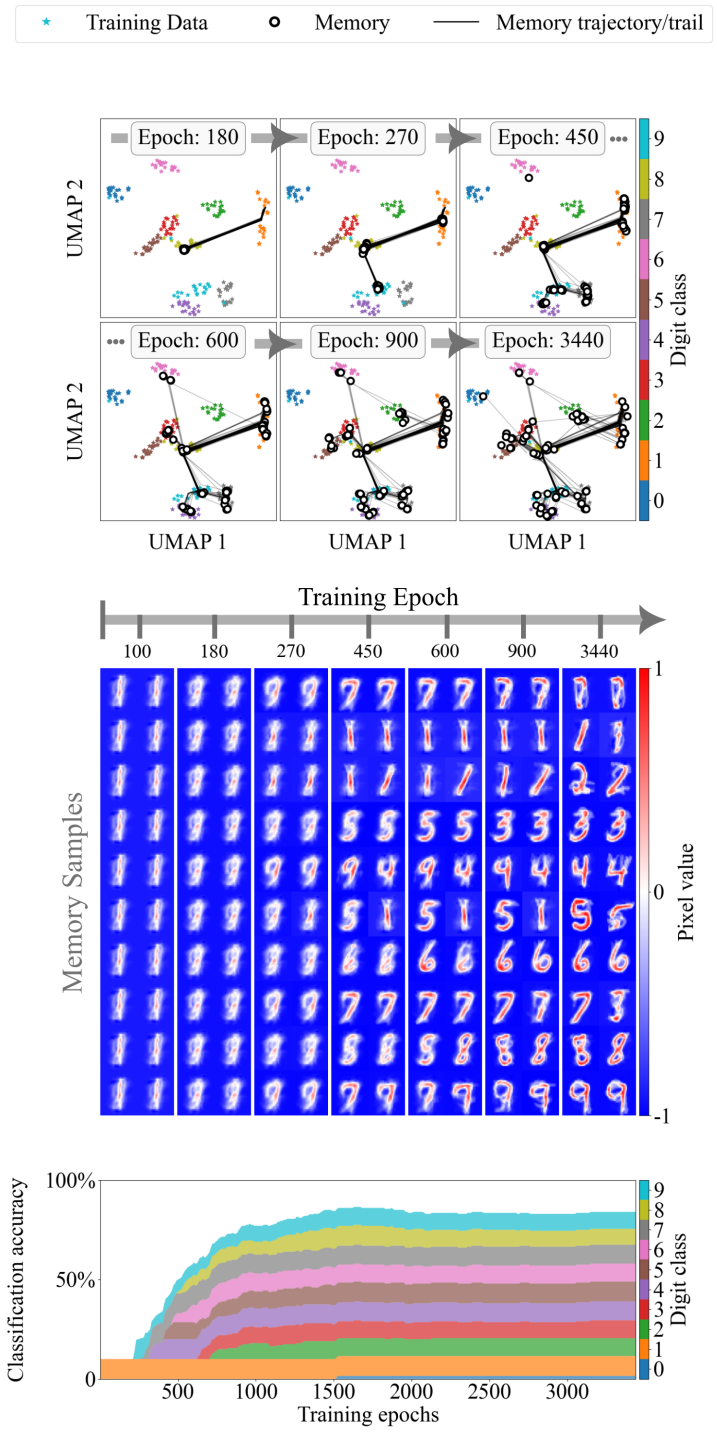

Figure S.8: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 40$, and $T_r = 0.85$, here however, the 200 digit training set is split into 4 miniBatchs each containing 50 digits (5 of each class).

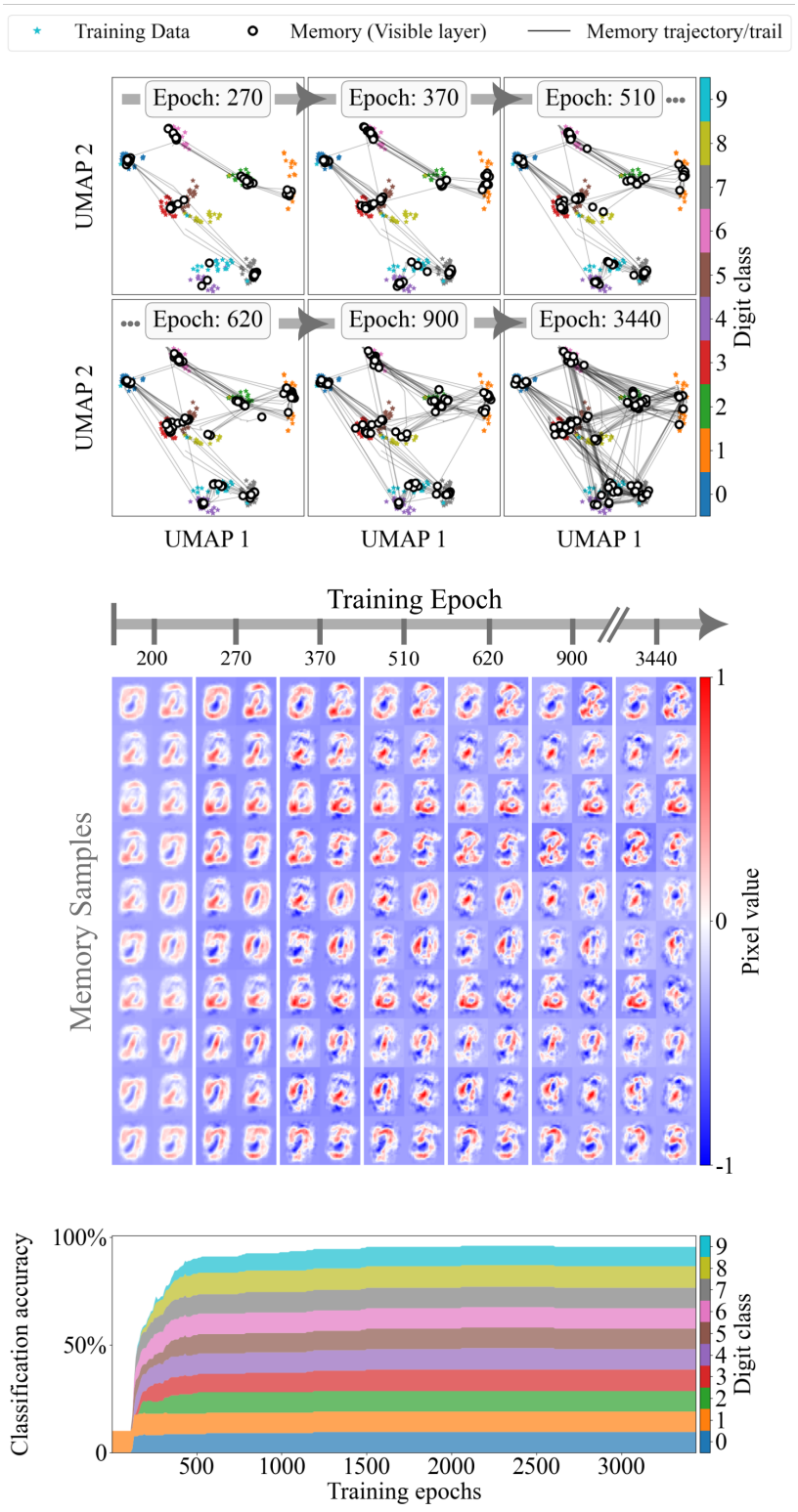

Figure S.9: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 3$, and $T_r = 0.85$, here however, with a momentum parameter set to 0.6.

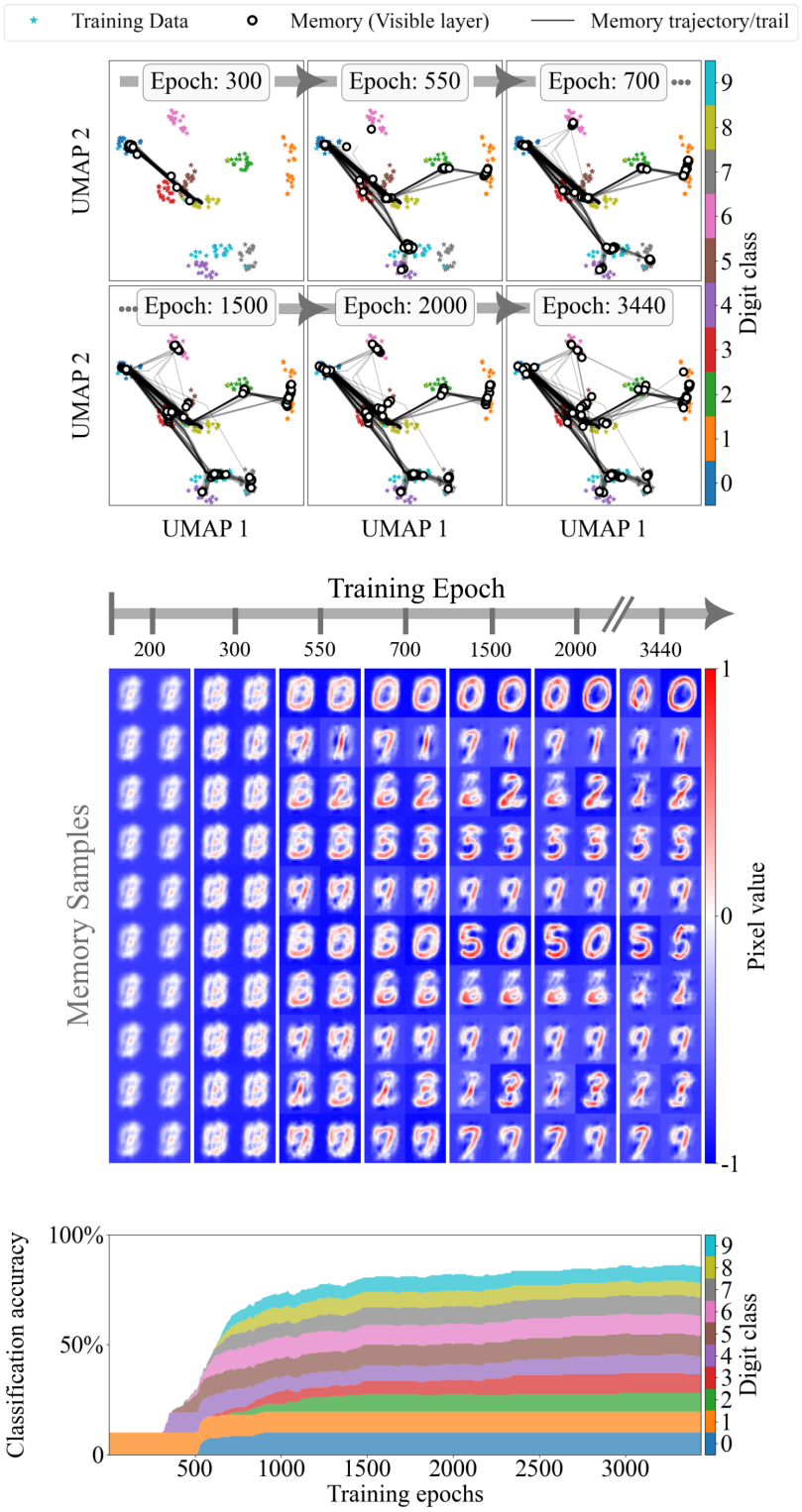

Figure S.10: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the parameters are $n = 15$, and $T_r = 0.85$, here however, with a momentum parameter set to 0.6.

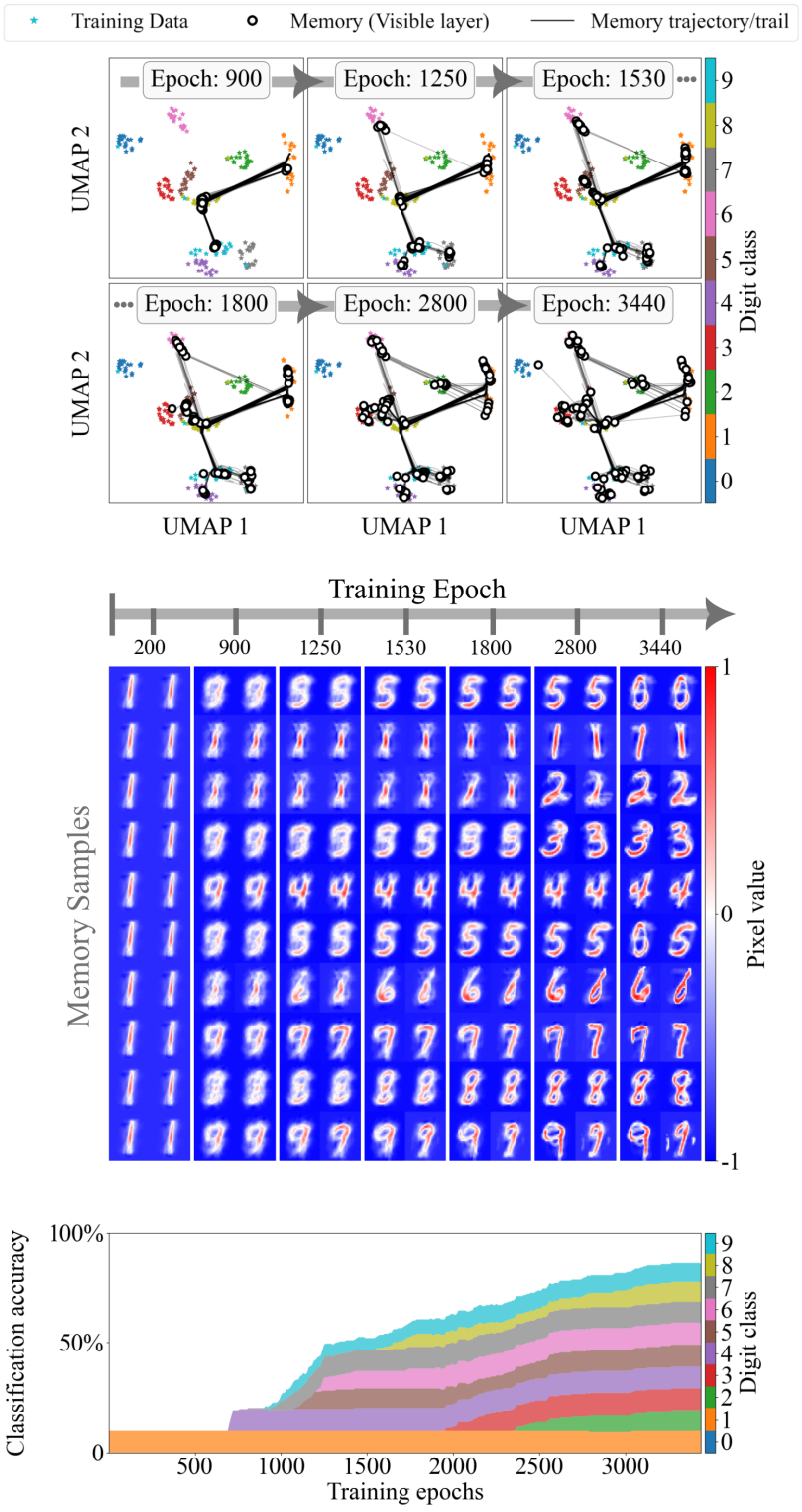

Figure S.11: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 30$, and $T_r = 0.85$, here however, with a momentum parameter set to 0.6.

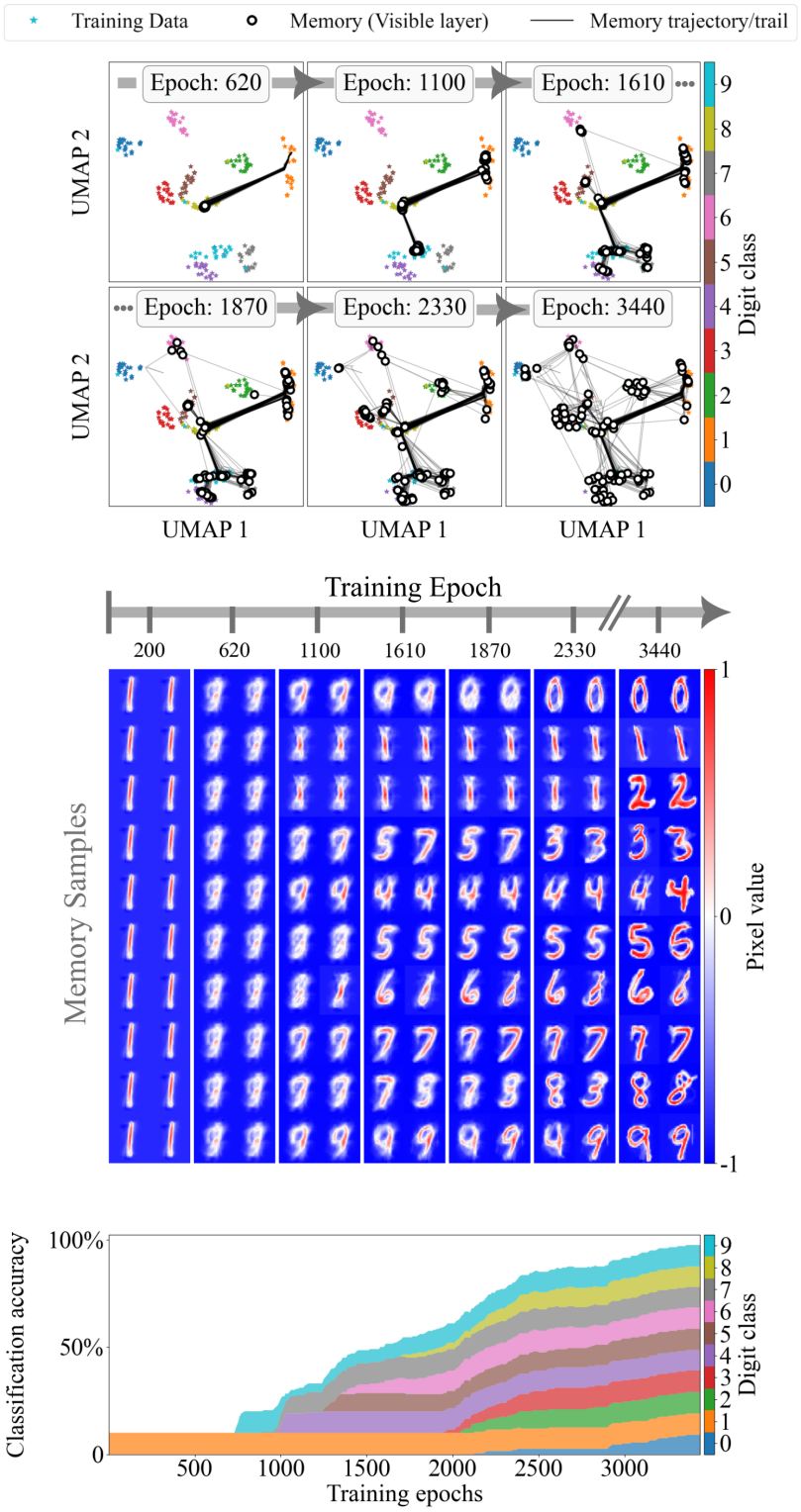

Figure S.12: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 40$, and $T_r = 0.85$, here however, with a momentum parameter set to 0.6.

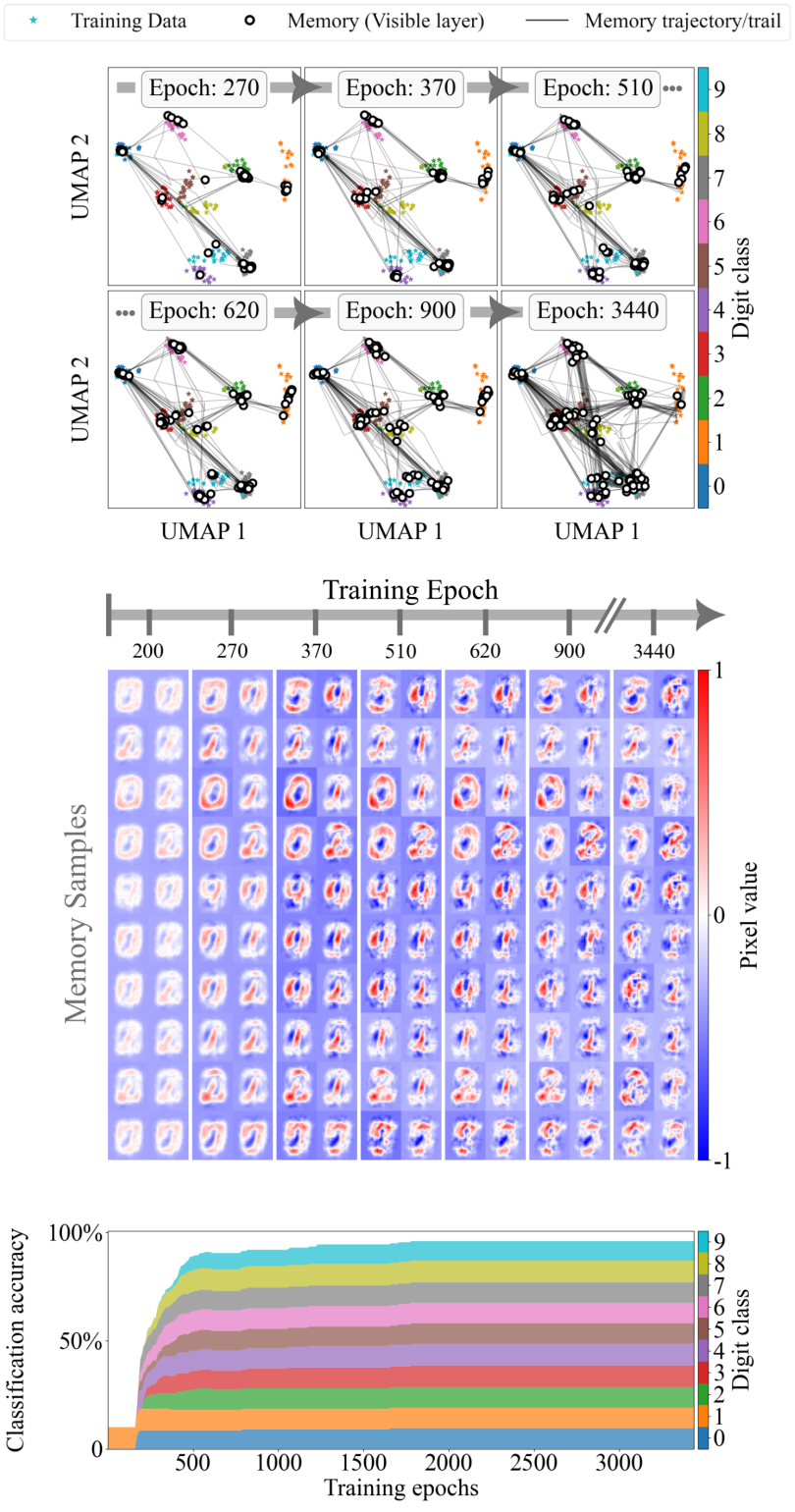

Figure S.13: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 3$, and $T_r = 0.85$, here however, with added noise at every epoch. The noise is Gaussian distributed with mean 0 and standard deviation 10^{-5}.

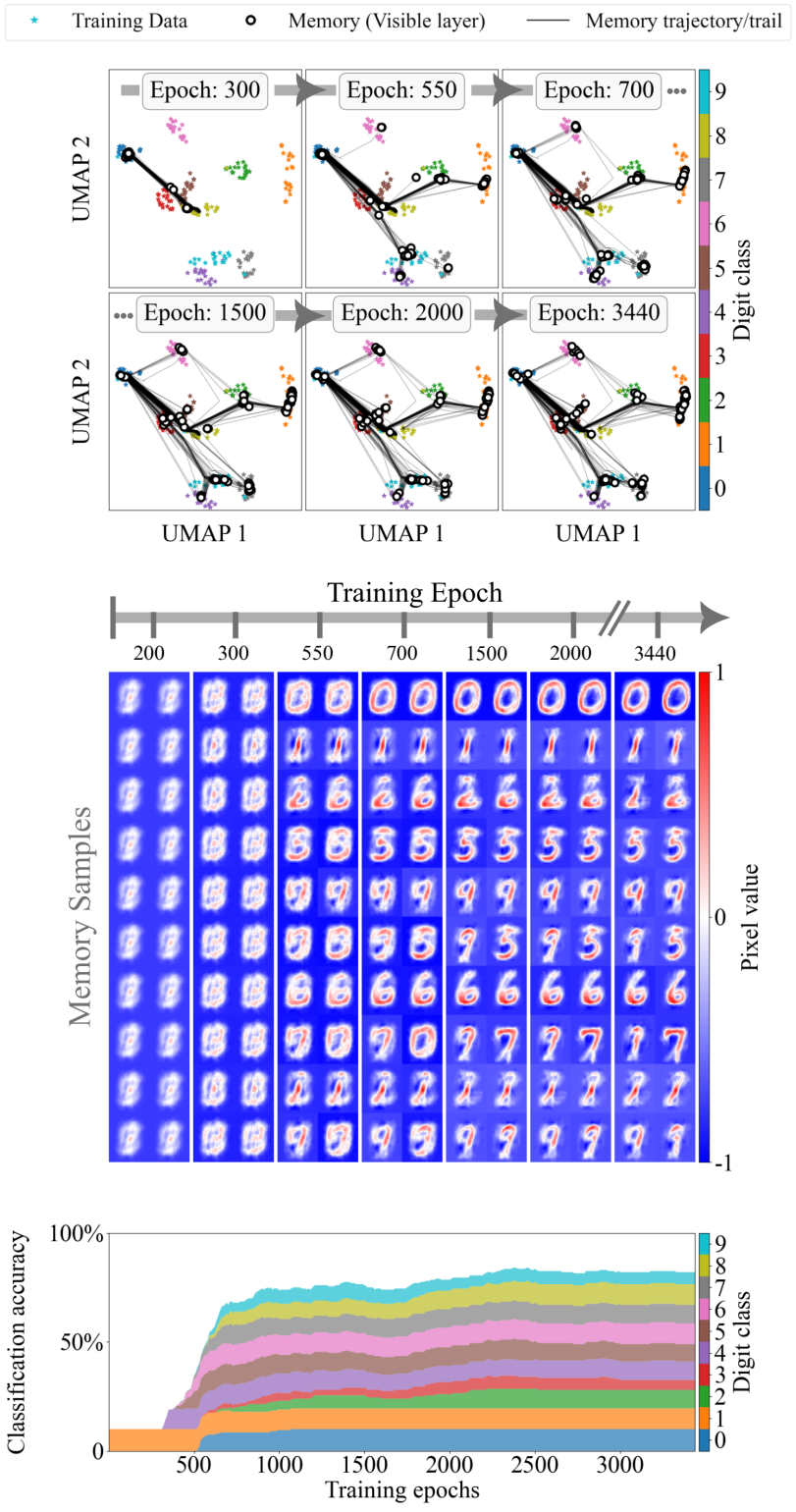

Figure S.14: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 15$, and $T_r = 0.85$, here however, with added noise at every epoch. The noise is Gaussian distributed with mean 0 and standard deviation 10^{-5}.

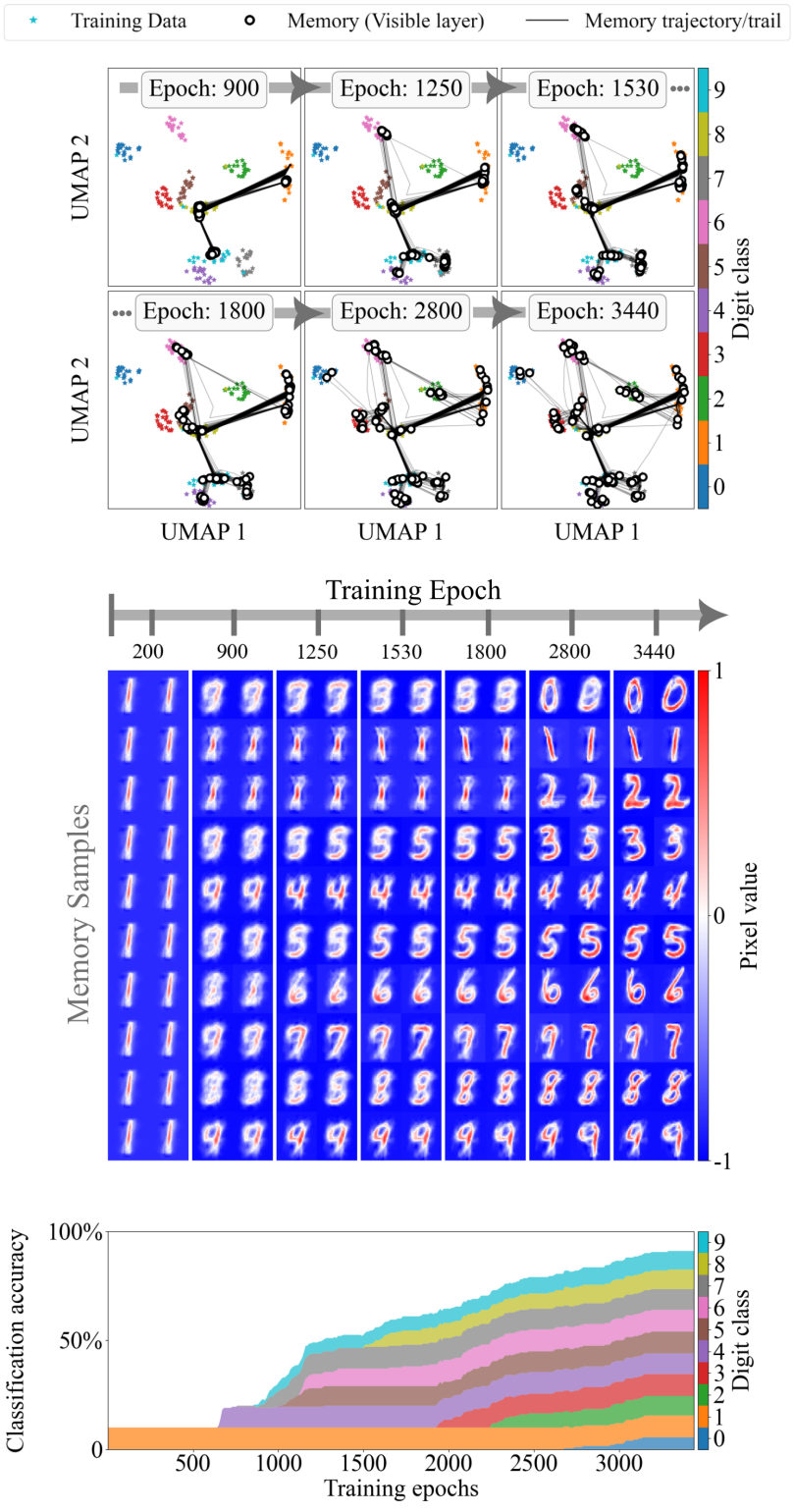

Figure S.15: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 3$, and $T_r = 0.85$, here however, with added noise at every epoch. The noise is Gaussian distributed with mean 0 and standard deviation 10^{-5}.

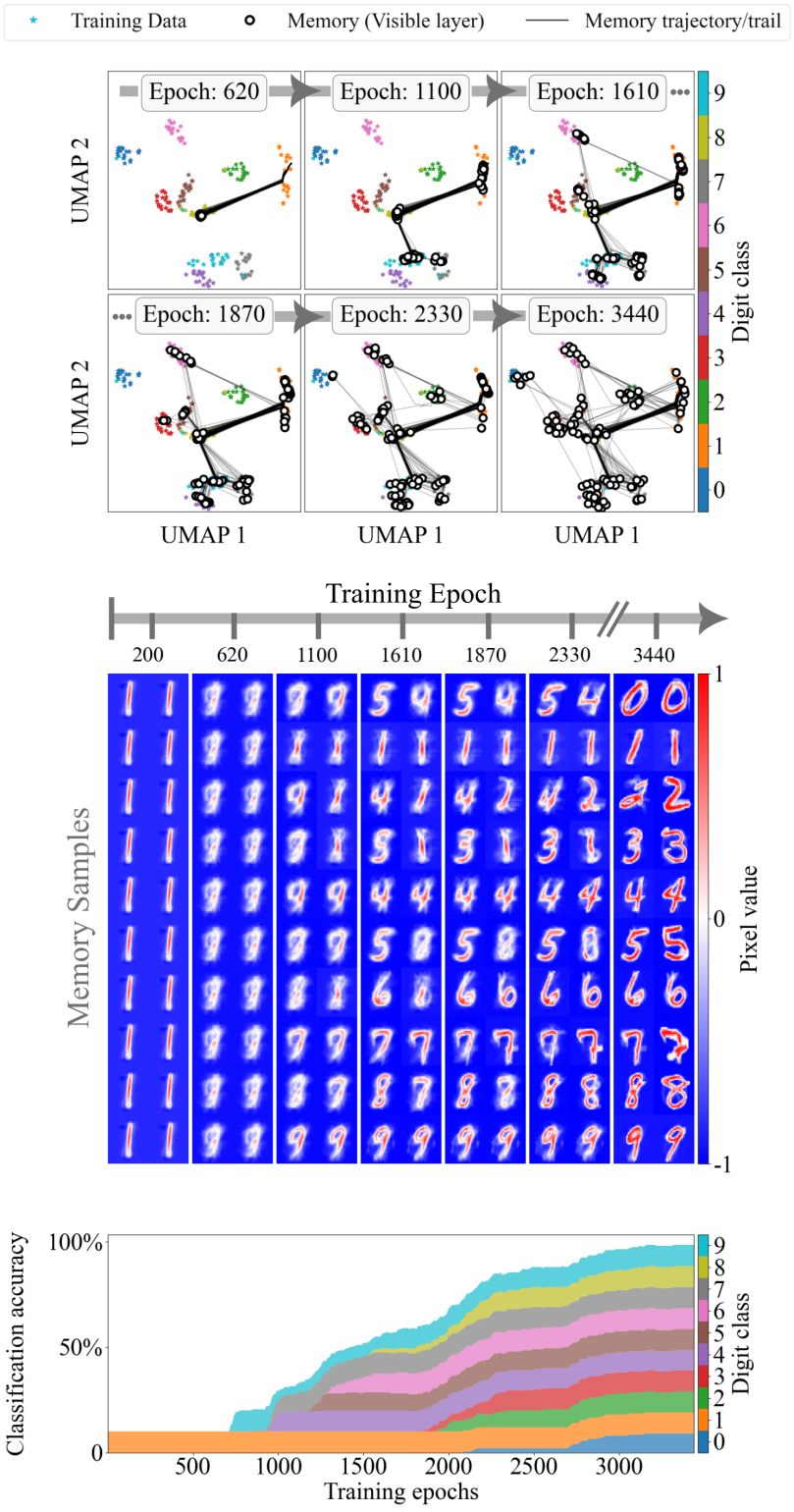

Figure S.16: A 100-memory system training on 200 digits (20 samples of each class) the same as in Fig. 3, the hyperparameters are $n = 40$, and $T_r = 0.85$, here however, with added noise at every epoch. The noise is Gaussian distributed with mean 0 and standard deviation 10^{-5}.

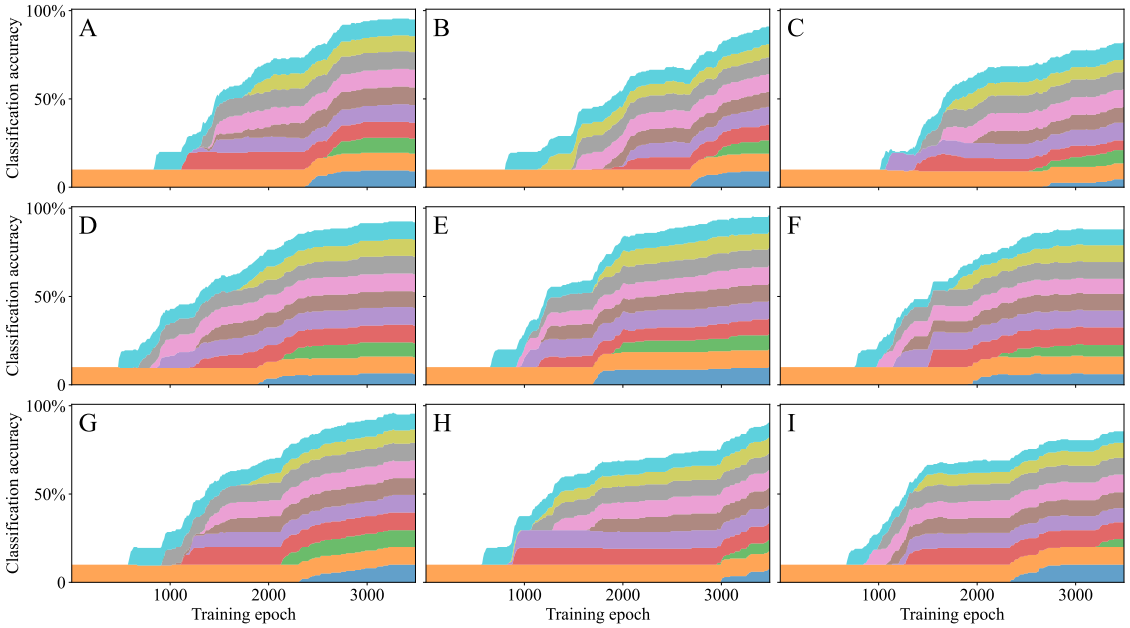

Figure S.17: Each subplot (A-I) represents a separate 100-memory network with $n = 30, T_r = 0.85$, training on 200 digits (10 per class) randomly selected for each subplot. Other network hyperparameters are as in Fig. 3.

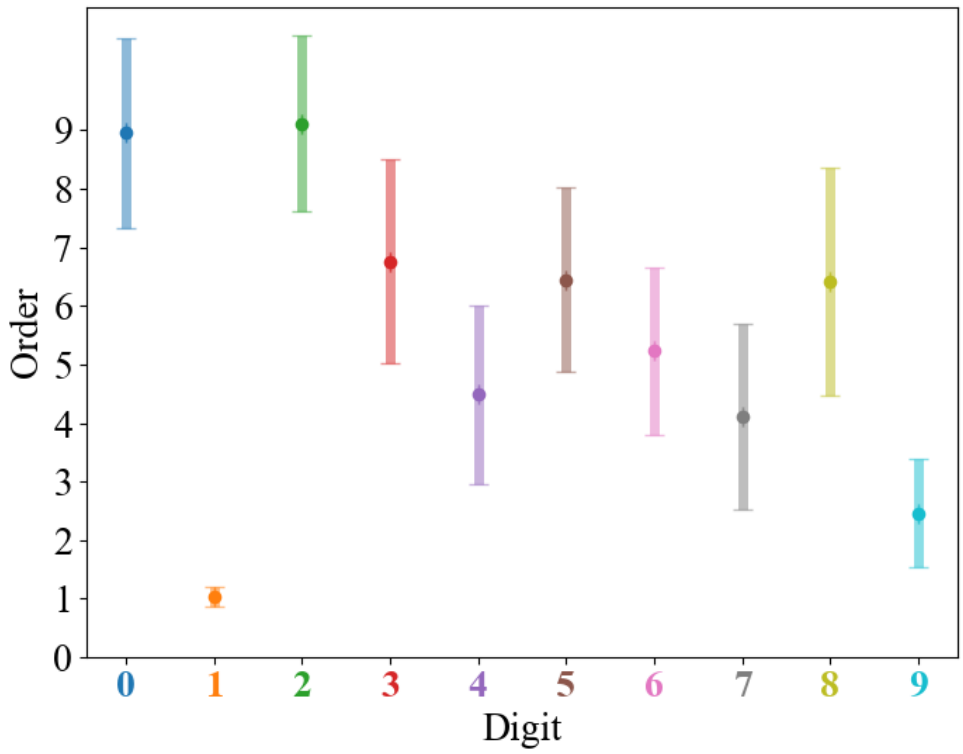

Figure S.18: The digit order statistics of 100 different 100-memory systems with $n = 30, T_r = 0.85$, each training on 200 digits (10 per class) randomly selected for each subplot. Other hyperparameters are as in Fig. 3.

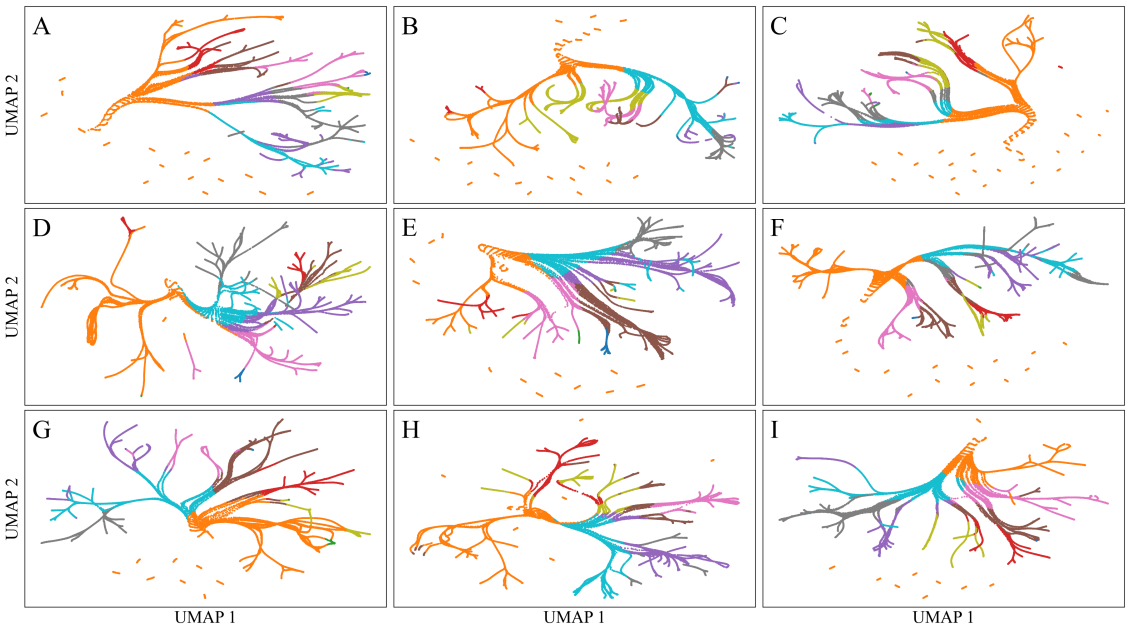

Figure S.19: Each subplot (A-I) represents a separate 100-memory network with $n = 30, T_r = 0.85$, training on 200 digits (10 per class) randomly selected for each subplot. Other network hyperparameters are as in Fig. 3. The UMAP parameters are left as default except for the number of neighbors set to 200. The rest of the methodology is as described in Section 7.4.

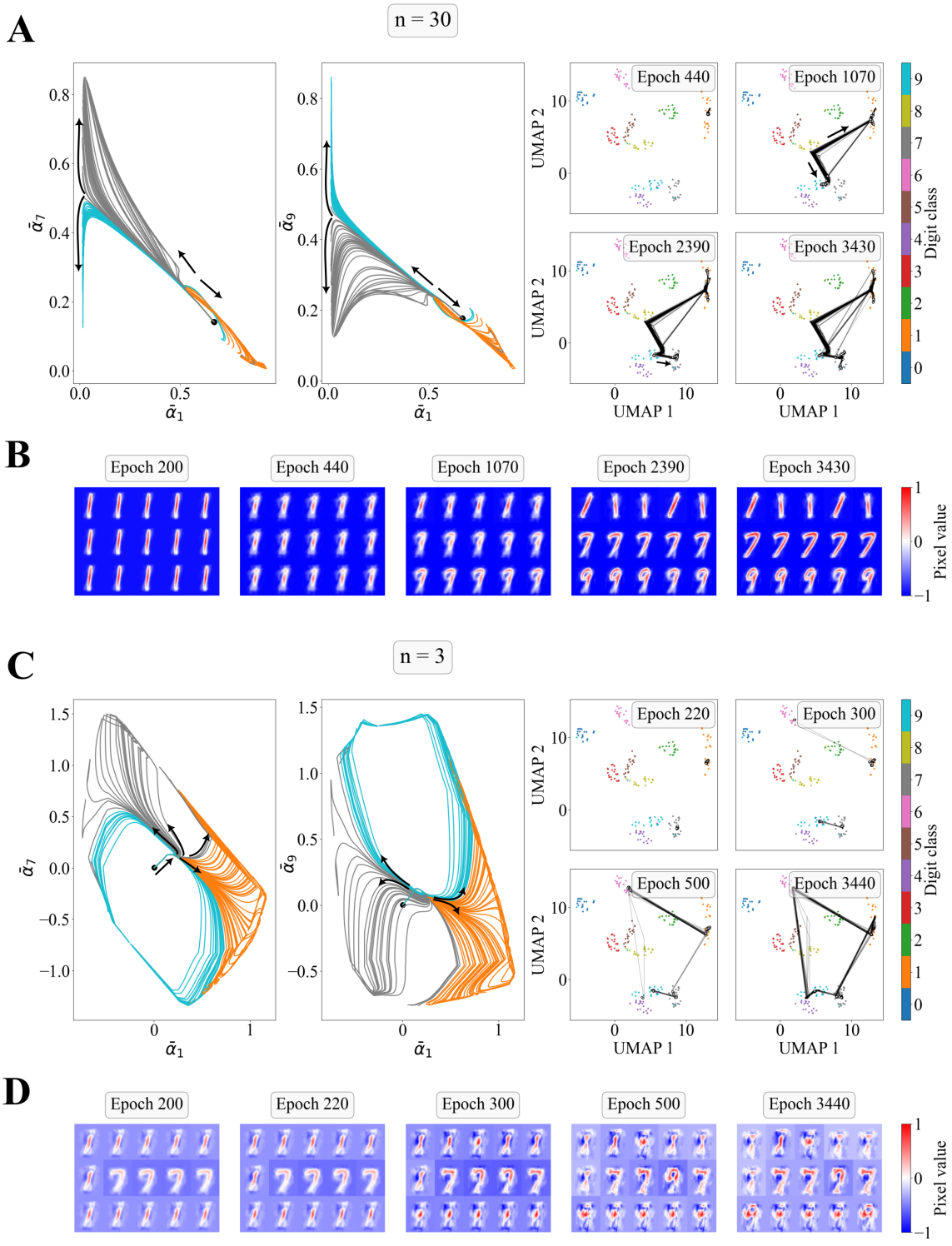

Figure S.20: A 100-memory system training on a set composed of 60 digits (20 of each class), the classes included in the training set are 1, 7 and 9. This system recapitulates the dynamics shown in Fig. 5.

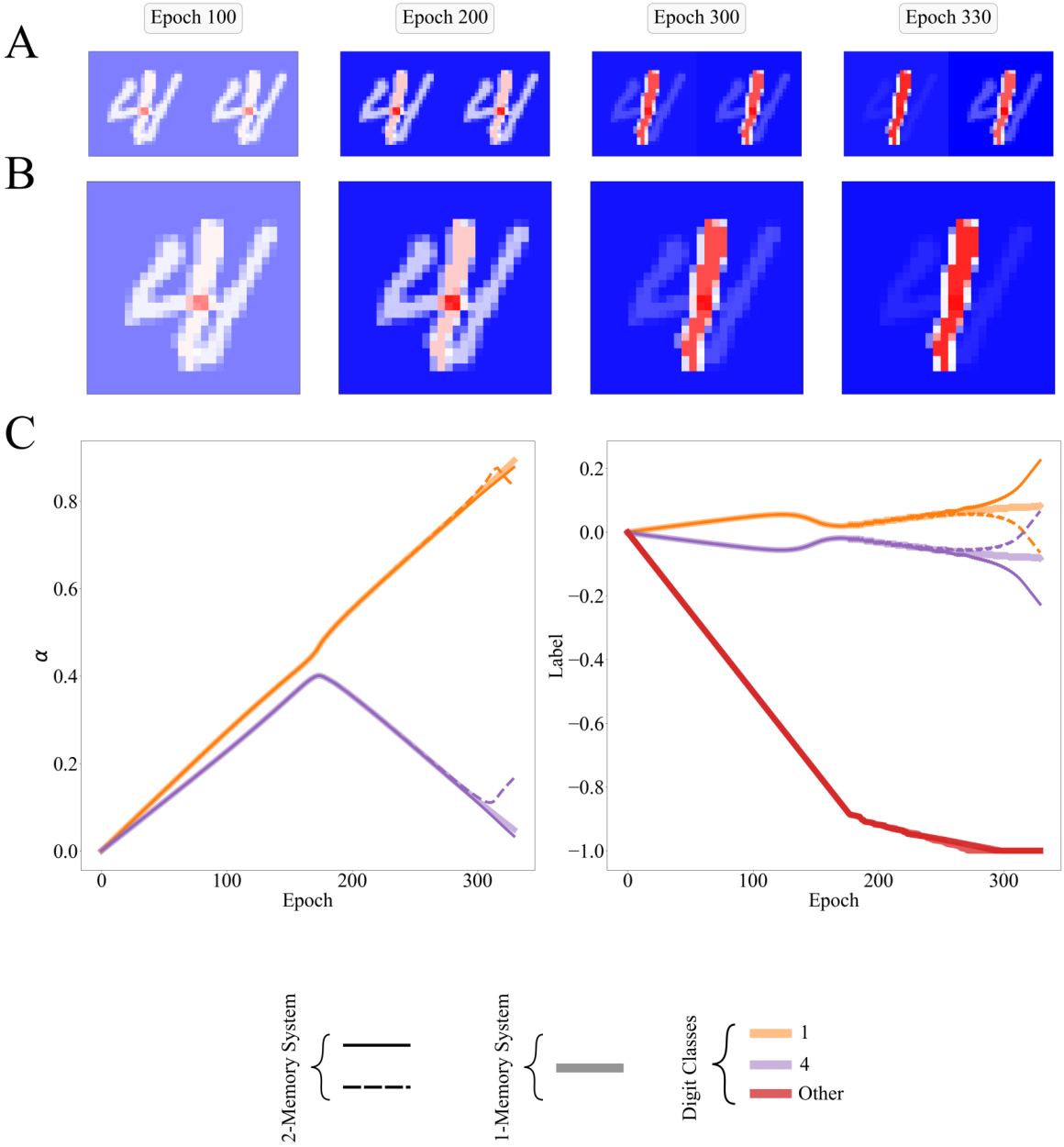

Figure S.21: The same training set as in Fig. 6, two training samples, a 1 and a 4. In A, a two memory system up to the split. In B, an equivalent 1-memory system. Both A, B indeed vary identically until the split. In C (left), the α's of both systems. In C (right), the labels of both systems. Here $n = 3$, $T_r = 0.89$.

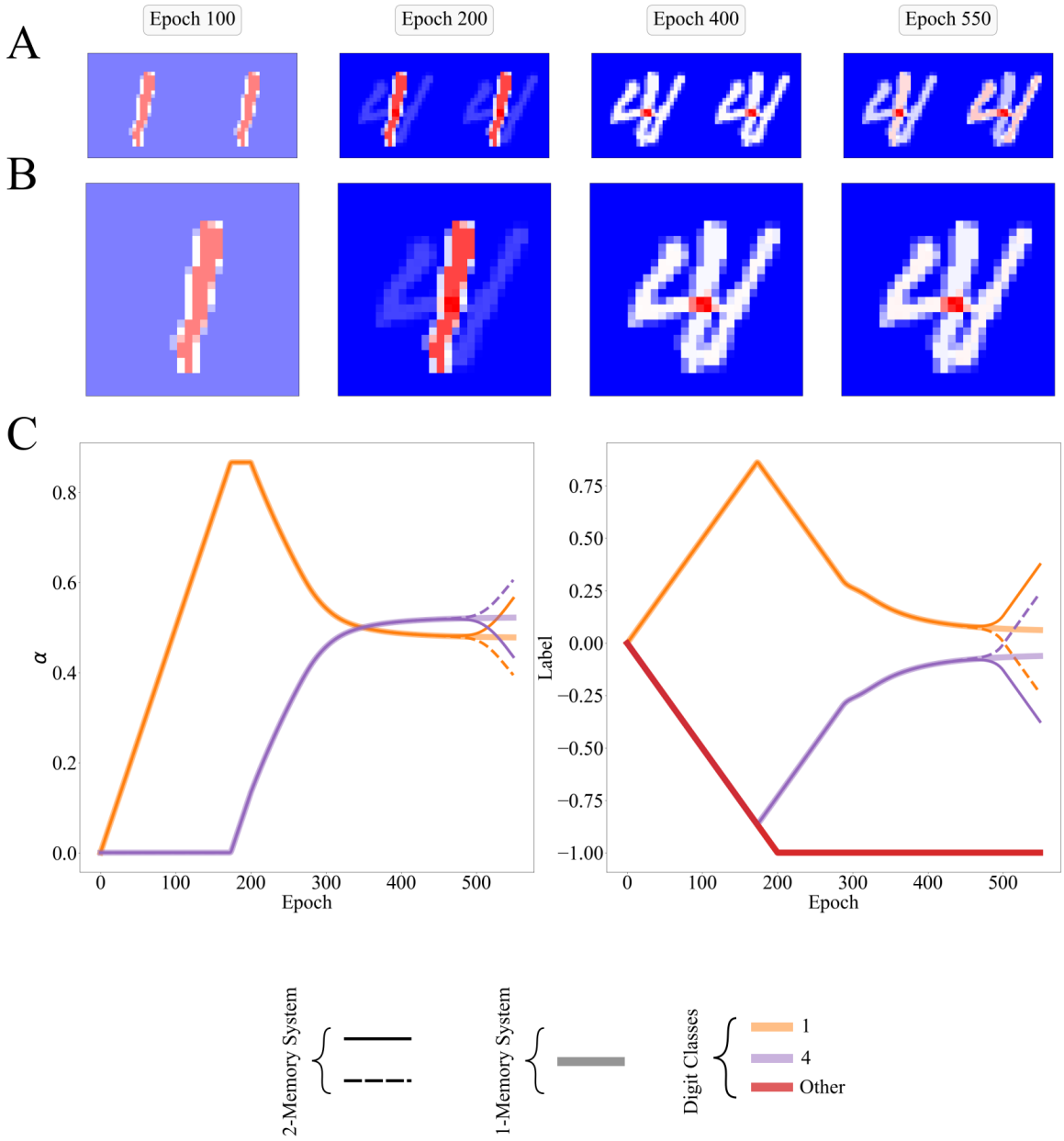

Figure S.22: The same training set as in Fig. 6, two training samples, a 1 and a 4. In A, a two memory system up to the split. In B, an equivalent 1-memory system. Both A, B indeed vary identically until the split. In C (left), the α's of both systems. In C (right), the labels of both systems. Here $n = 30$, $T_r = 0.89$.

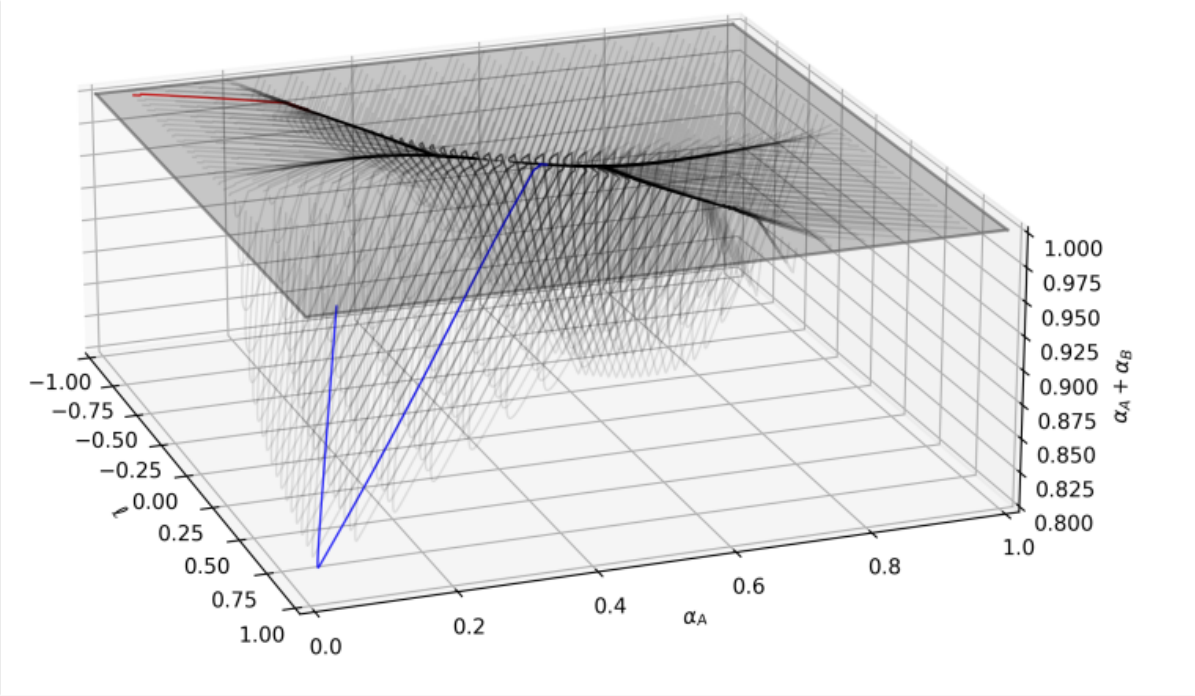

Figure S.23: Flow lines of the 1-memory system for $n = 15, T_r = 0.89$. All flow lines start in the $\alpha_A + \alpha_B = 1$ subspace, and are generated by simulating the 1-memory dynamics. Note that although some flow lines leave the $\alpha_A + \alpha_B = 1$ subspace, they eventually converge to it. In particular, two flow lines are highlighted in red and blue, the former never leaves the subspace, while the blue line drops to $\alpha_A + \alpha_B \sim 0.8$ before converging to the nullclines.

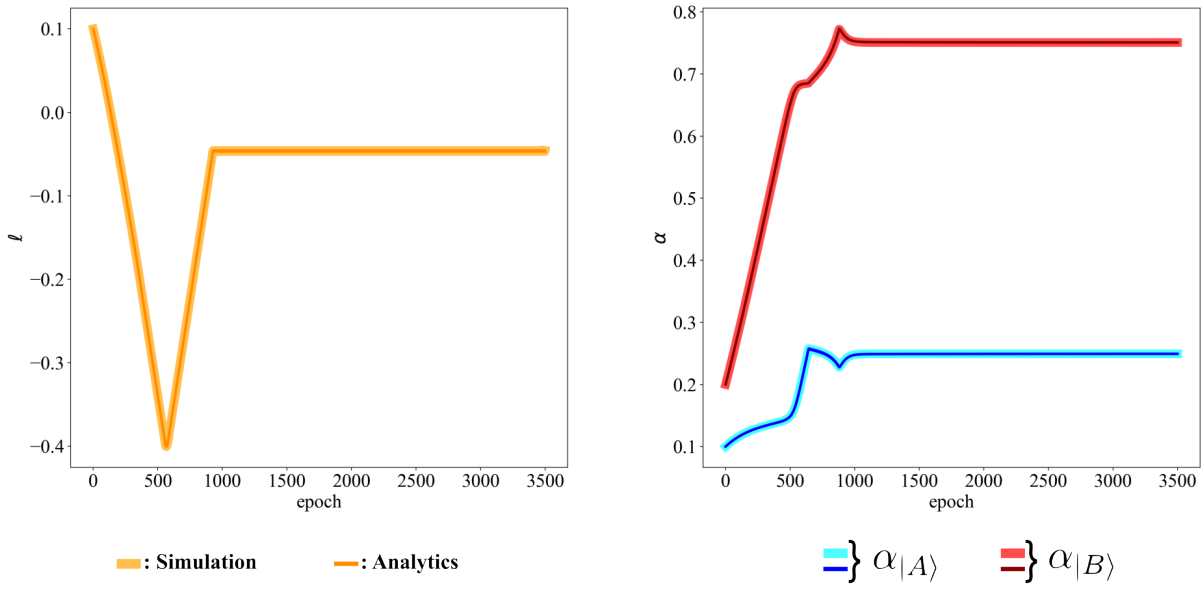

Figure S.24: The same training set as in Fig. 6, two training samples, a 1 and a 4. Comparing the simulations of the complete system to the numerical simulations of the equations derived in 3.

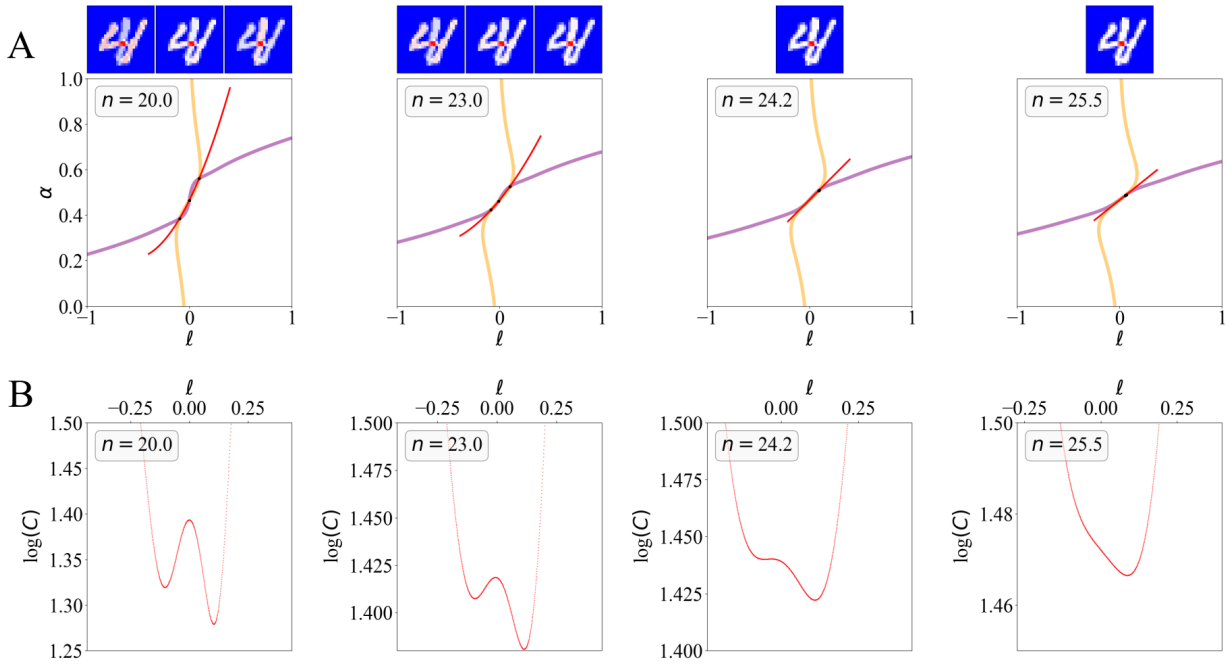

Figure S.25: Using the same training set as in Fig. 6, a 1-memory system with two training samples. In A, we show the nullclines for 4 different n values, near the saddle node bifurcation. We also define a red curve which goes through all fixed points, and varies as little as possible (qualitatively) from one n to the other. In B, we plot the cost function along that red curve to show the saddle-node bifurcation.

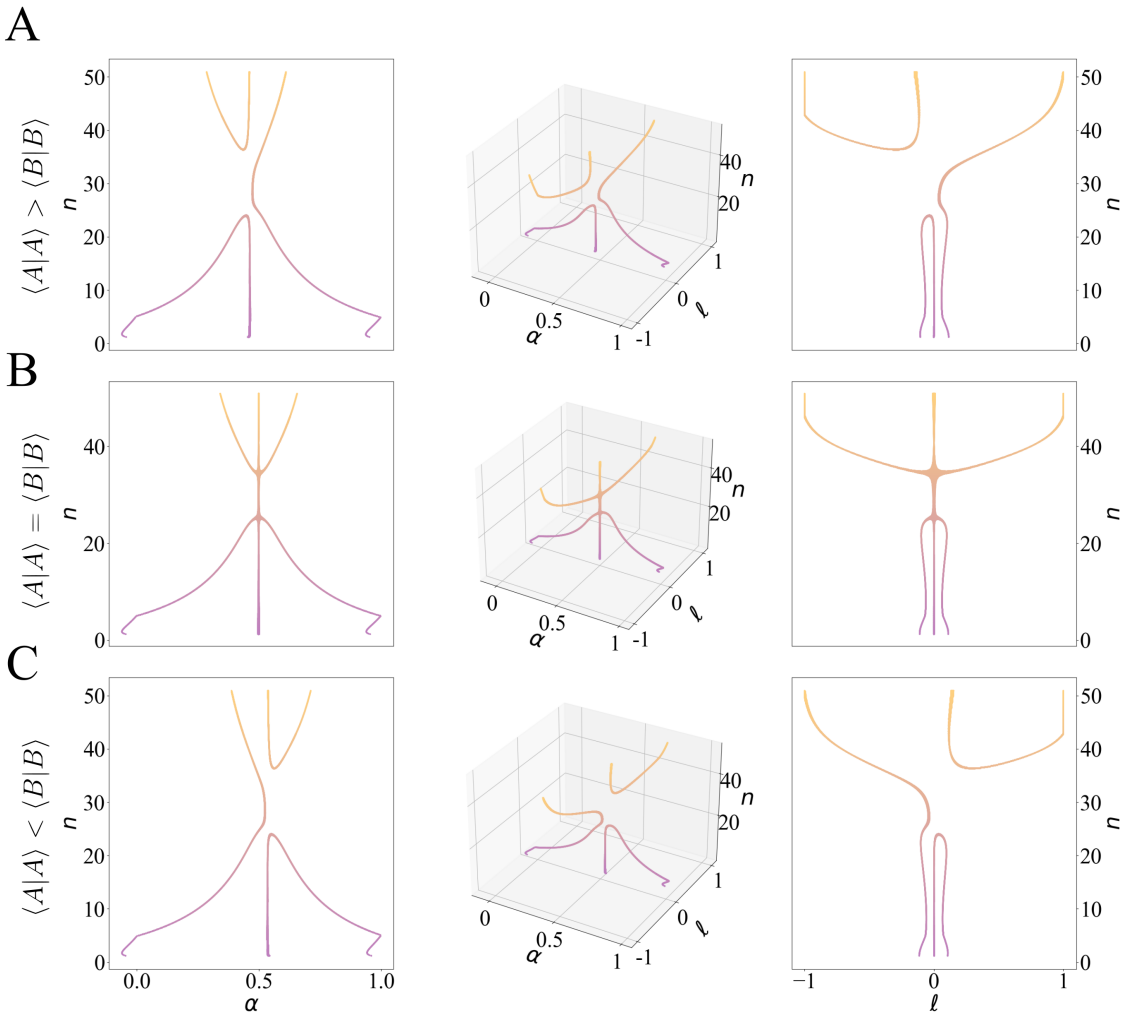

Figure S.26: The bifurcation diagrams of 3 1-memory systems, each with a different $\langle A|A\rangle$, $\langle B|B\rangle$ relationship, while $\langle A|B\rangle$ is fixed. This shows the symmetry required to obtain a pitchfork bifurcation is $\langle A|A\rangle = \langle B|B\rangle$. This is complementary to Fig. 6.

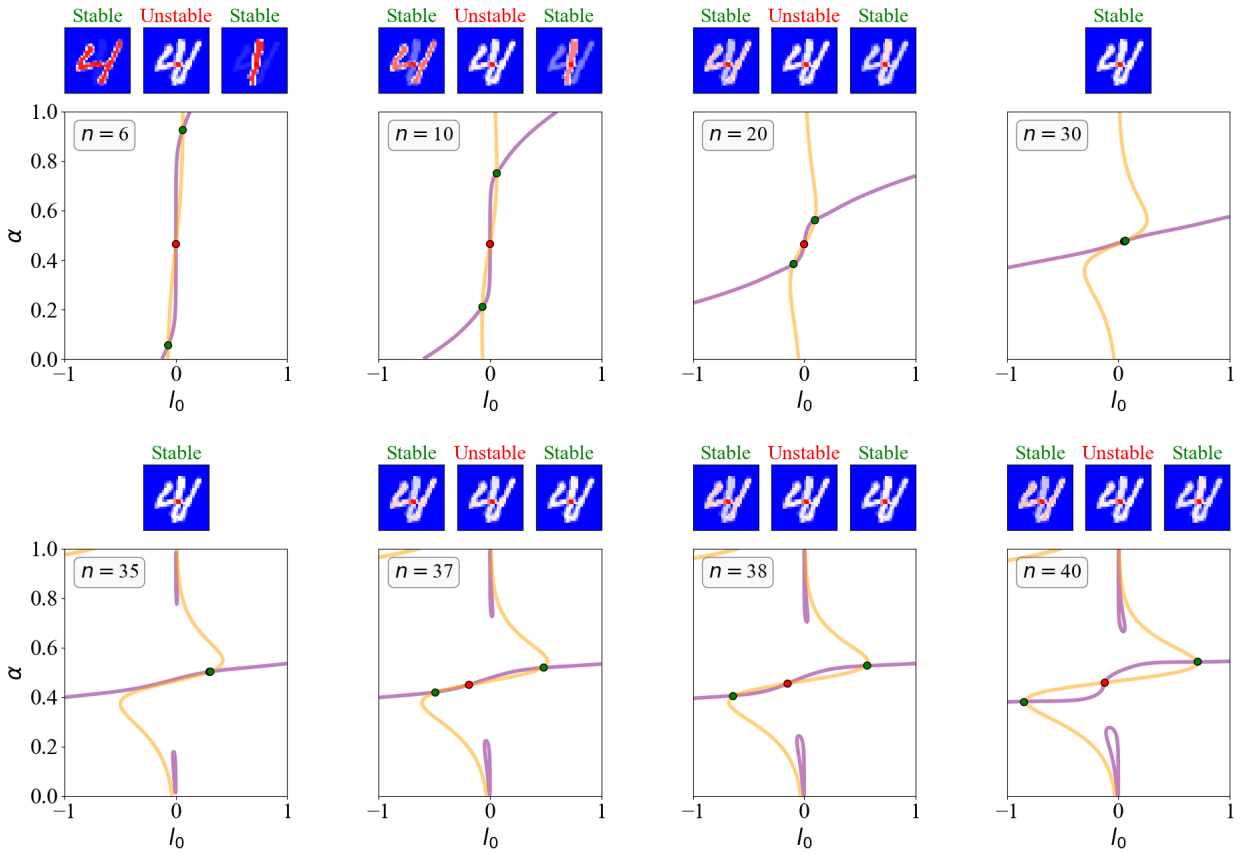

Figure S.27: The same system as in Fig. 6, a 1-memory system with two training samples. Here we show nullclines for more values of n.

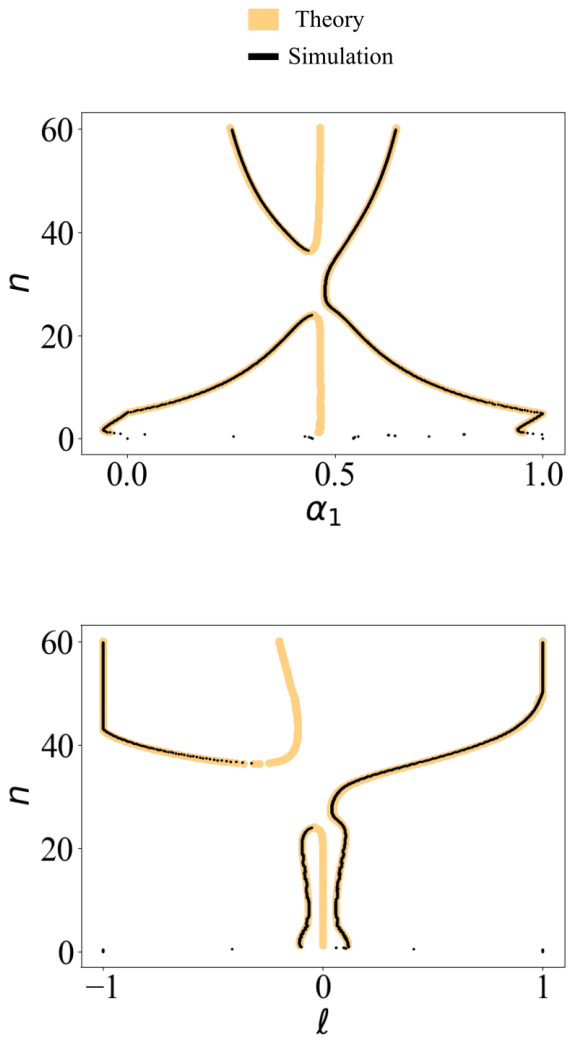

Figure S.28: Using the same training set as in Fig. 6, a 1-memory system with two training samples. We compare the analytical bifurcation diagram to the one obtain through simulation.

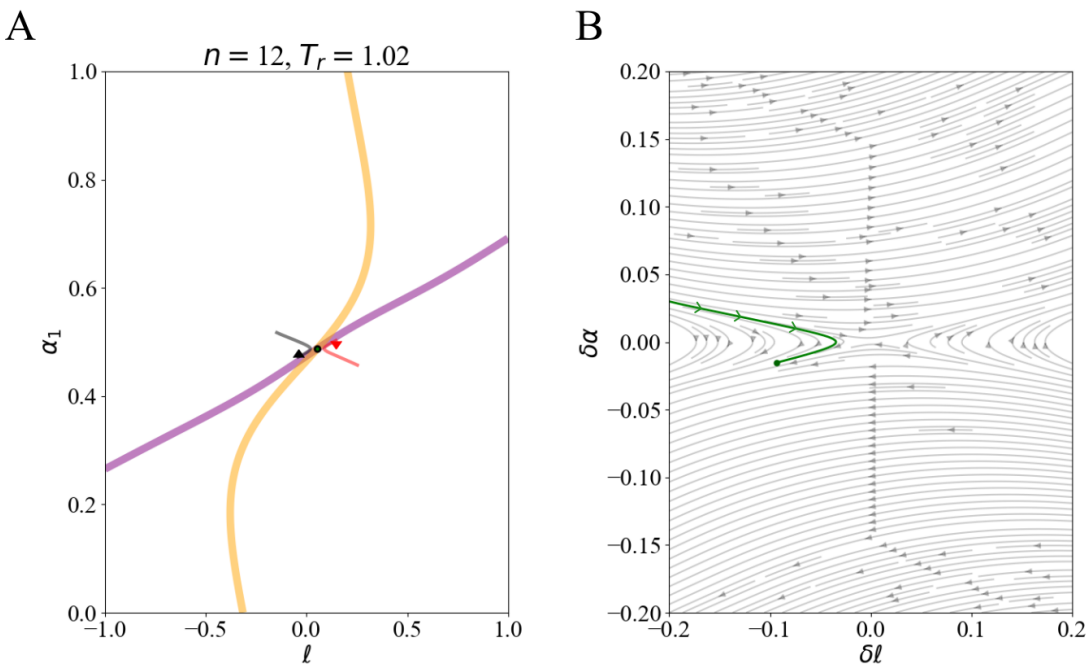

Figure S.29: A 2-memory system with the same training set (2 digits) as in Fig. 6. In A, the dynamics are plotted in the α, ℓ space for each memory (one in red, the other in black). In B, the difference between memories along α ($\delta\alpha$) and ℓ ($\delta\ell$). The flow lines are defined from the first degree expansion of the system as described in section 4. The hyperparameters are $n = 12$, and $T = 0.89$. The system is initialized centered at the fixed point shown in A, with $\delta\alpha = 0.31$ and $\delta\ell = -0.2$.

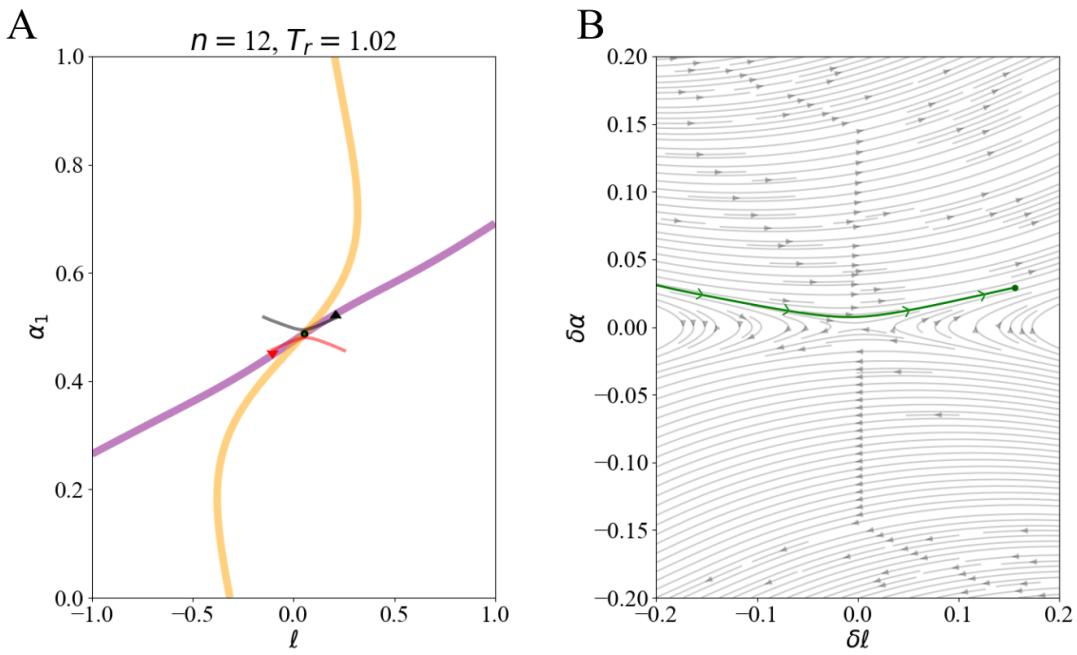

Figure S.30: A 2-memory system with the same training set (2 digits) as in Fig. 6. In A, the dynamics are plotted in the α, ℓ space for each memory (one in red, the other in black). In B, the difference between memories along α ($\delta\alpha$) and ℓ ($\delta\ell$). The flow lines are defined from the first degree expansion of the system as described in section 4. The hyperparameters are $n = 12$, and $T = 0.89$. The system is initialized centered at the fixed point shown in A, with $\delta\alpha = 0.30$ and $\delta\ell = -0.2$.

9 Supplementary Movies

Movie 1

Based on Figure 3, this movie illustrates the learning dynamics for a 100 memory system, trained on 20 digits of each class (200 in total). The rescaled temperature is 0.85, and $n = 3$. On the left we show the labels, each row is a separate class d. In the middle, the memories are plotted in a 10x10 grid. The standard colormap is used (blue: -1, white: 0, red: 1). On the right, the UMAP projection of the memories.

Movie 2

Based on Figure 3, this movie illustrates the learning dynamics for a 100 memory system, trained on 20 digits of each class (200 in total). The rescaled temperature is 0.85, and $n = 30$. On the left we show the labels, each row is a separate class d. In the middle, the memories are plotted in a 10x10 grid. The standard colormap is used (blue: -1, white: 0, red: 1). On the right, the UMAP projection of the memories.

Movie 3

Based on Figure 4, this movie illustrates the dynamics of a system with 400 memories, trained on 1000 digits (100 per class) with $n = 3, T_r = 0.85$. The UMAP embedding is fit using the α coefficients for every memory between epoch 100 to 1000 of training, with a time resolution of 10, see section 7.4 for more details. Points are colored based on the dominating element of the associated label at their respective timepoint.

Movie 4

Based on Figure 4, this movie illustrates the dynamics of a system with 400 memories, trained on 1000 digits (100 per class) with $n = 30, T_r = 0.85$. The UMAP embedding is fit using the α coefficients for every memory between epoch 400 to 2300 of training, with a time resolution of 5, see section 7.4 for more details. Points are colored based on the dominating element of the associated label at their respective timepoint.

Movie 5

Based on Figure 6, this movie illustrates the dynamics of a 2-memory system for many n. As in Figure 6 the rescaled temperature is 0.89 and the training set consists of a 1 and a 4. Each subplot illustrates the dynamics for different n-values (6, 10, 20, 30, 35, 37, 38, 40). For each subplot, six 2-memory systems are shown; each pair of upper/lower triangle markers (of the same color) is an independent simulation. The yellow line represents the label nullcline. The purple line represents the memory nullcline. Green points are stable fixed points of an equivalent 1-memory system; saddles of the 2-memory system. Red points are unstable fixed points of an equivalent 1-memory system. The memories tend to converge towards the nearest nullcline, follow them towards a fixed point and split.

Movie 6

Based on Figure 9, this movie illustrates the final states of a 100-memory system as a function of temperature. The hyperparameter n is fixed to 25, and the rescaled temperature is varied from 0.83 to 1.15. The system is trained on a training set containing all digit classes, with 20 training samples per class. We see that as temperature increases, the system gets stuck at saddles, starting at the most downstream saddle and going up until the system only splits once and contains effectively two types of memories.

Movie 7

Similar to Movie 6, here the hyperparameter n is fixed to 15, and the rescaled temperature is varied from 0.83 to 1.2.

Movie 8

This movie follows the same the same setup as Movie 7, but with the hyperparameter n fixed to 40.