Rebuilding the AHA Hippocampal Model from Its Paper: A Reproduction Study on One-Shot Omniglot

Jacob Krajewski
Independent researcher

October 2026

Abstract

People can learn a new handwritten character from a single drawing. Lake and colleagues measure this ability in machines with a one-shot task on the Omniglot dataset: a model studies one drawing of each of 20 characters, then must match each to a new drawing of the same character by a different person, where guessing scores 5%. Kowadlo, Ahmed and Rawlinson's Artificial Hippocampal Algorithm (AHA) attempts the task with a model of the hippocampus, the brain region that stores single experiences quickly. A vision network, trained in advance on other alphabets, feeds a fast-learning memory whose parts are named after hippocampal regions. Their paper reports 86.4% for AHA and 71.6% for its vision network used alone, so the memory adds 14.8 percentage points. I rebuilt AHA in PyTorch from the text of that paper and scored it on the same task. With every setting the paper states, plus centring each drawing on its ink, which the authors' later code does, AHA scores 64.7% and its vision network 52.2%: the memory adds 12.5 points, but both scores sit about 20 points below the paper's. Dividing each vector by its length before comparing drawings, a choice the paper does not specify, raises the vision network to 63.0% and AHA to 70.6%, and shrinks the memory's contribution to 7.6 points. No option I tested brings either score to the paper's level, and in the best setting, 8.6 of the 15.8 points AHA falls short are already missing from its vision network. I list the places where the paper is silent or contradicts itself, with page references. Code: https://github.com/sachio222/aha4

1. Introduction

People learn new concepts from very little. Shown one drawing of a handwritten character they have never seen, a person can pick out another drawing of the same character from a set of 20, and gets it right 95.5% of the time (Lake et al., 2015, p. 1335). Lake and colleagues built this test into a benchmark for machines. Their Omniglot dataset holds 1,623 handwritten characters from 50 alphabets, each character drawn by 20 different people (Lake, 2015). Thirty of the alphabets form a background set, which a model may learn from in advance; the other 20 form an evaluation set, kept for testing. In their one-shot classification task, a model is shown one drawing of each of 20 characters from one evaluation alphabet, which I call the study drawings. It then sees 20 new drawings of the same characters by a different person, the test drawings, and must pair each study drawing with the test drawing of its character. Guessing scores 5%. The task repeats this 20 times, in 20 runs over 10 alphabets, for 400 pairings in all (Lake et al., 2015, p. 1335; Kowadlo et al., 2020, p. 8).

A long-standing account of how a brain learns this fast is complementary learning systems theory. As Kowadlo et al. (2020, pp. 1–2) summarise it, the neocortex learns slowly, building overlapping representations

of what many experiences have in common, while the hippocampus learns a single experience at once and keeps it distinct from similar ones, acting as a short-term memory that can later replay what it stored to the neocortex. Input reaches the hippocampus from the entorhinal cortex (EC), the main route between neocortex and hippocampus. It passes through the dentate gyrus (DG), which gives similar inputs very different sparse codes, codes in which only a few cells are active. Keeping similar inputs apart in this way is called pattern separation. DG drives CA3, a region whose cells connect to one another. These recurrent connections let a partial or noisy cue bring back the whole pattern stored from an earlier input, which is called pattern completion. CA1 then maps CA3's pattern back to the form the input had in EC (Kowadlo et al., 2020, p. 3). EC also connects to CA3 directly. In the version of this account that Kowadlo et al. (2020, p. 4) adopt, from Rolls, input arriving through DG decides what CA3 stores, and the direct path from EC supplies the cue that recalls it.

Kowadlo et al. (2020) built this account into a working model, the Artificial Hippocampal Algorithm (AHA). A vision network, trained in advance on the background alphabets and then frozen, stands in for neocortex and EC. Four fast-learning parts stand in for the hippocampus: a DG layer that separates patterns; a CA3 built as a Hopfield network, a recurrent network whose stored patterns are stable states, so that activity started from a cue changes step by step until it settles on the closest stored pattern; a pattern retrieval network (PR) that learns the direct path from EC to CA3; and a CA1 that turns CA3's pattern back into an image. On the one-shot task, the paper reports that the vision network used alone scores 71.6%, and that AHA scores 86.4% (Kowadlo et al., 2020, p. 8). The gain of 14.8 points, which the paper calls "almost 15%", is its evidence that a memory organised like the hippocampus improves one-shot learning over the vision network it builds on. AHA's 86.4% also matches a simple convolutional network trained with labels, at 86.5% (Kowadlo et al., 2020, p. 9), though AHA uses no labels.

AHA is a natural starting point for anyone who wants a hippocampal model that works on real images, because each of its parts corresponds to a named brain region. Work built on it inherits its numbers, so those numbers need to be reachable by someone other than the authors. The paper describes the model in its text, figures and a table of hyperparameters (Kowadlo et al., 2020, Table 3, p. 19). The authors later published a PyTorch implementation, pt-aha (Cerenaut, 2021), which differs from the paper in places described below.

This paper asks whether AHA, rebuilt from the text of its paper, reaches the paper's numbers on the one-shot task. Where it does not, I ask which details the paper leaves open account for the difference, and whether the paper's main claim, that the hippocampal parts add accuracy over the vision network alone, still holds.

With every setting the paper states, plus centring each drawing on its centre of ink, a step the paper never mentions but pt-aha performs, the rebuilt AHA scores 64.7% and its vision network alone 52.2%. The hippocampal parts add 12.5 percentage points (from here on, points), close to the paper's 14.8, but both scores sit about 20 points below the paper's. Comparing drawings after dividing each vector by its length raises the vision network to 63.0% and AHA to 70.6%; the vision network gains more, so the memory's contribution falls to 7.6 points. No setting I tested raises the vision network above 67.3% or AHA above 72.0%. The contributions are:

1. A PyTorch implementation of AHA built from the text of its paper, with 44 tests, released with the scripts that regenerate every number here (see Code and data availability).
2. Scores at every stage of AHA on the one-shot task, against the paper's (Section 4.1).

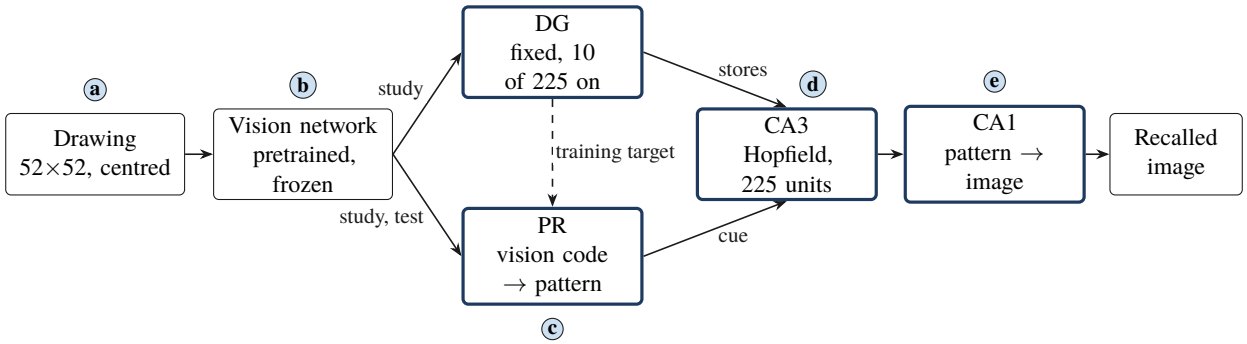


Figure 1: AHA's parts and scoring points. A drawing passes through the vision network, which was trained in advance on the background alphabets and stays fixed. The four parts with heavy outlines learn during each run and are reset after it. While studying, DG turns each drawing's vision code into a sparse pattern, CA3 stores it, PR learns to produce it from the vision code (dashed arrow), and CA1 learns to turn it back into the drawing. At test, PR turns each new drawing's vision code into a cue, CA3 settles from that cue, and CA1 draws what CA3 settled on. Circled letters mark where the model can be scored: (a) raw pixels, (b) vision network alone, the paper's baseline, (c) PR, the stage behind the paper's headline 86.4%, (d) CA3 after settling, and (e) CA1's recalled image.

3. Measurements of how much each detail the paper leaves open moves those scores: scaling before comparison, the direction of matching, the pretraining loss and seed, centring and the interest filter (Sections 4.2 to 4.6).
4. A list of places where the paper is silent or contradicts itself, with page references and the choice made here for each (Section 3.2).

2. AHA and how it is scored

Figure 1 shows AHA's parts and the points at which the paper scores it. Table 1 gives each part's size and the name the paper uses for it. The paper writes each part as Function(Region): PS(DG) for pattern separation in DG, PC(CA3) for pattern completion in CA3, PR(EC-CA3) for pattern retrieval along the EC-to-CA3 path, PM(CA1) for pattern mapping in CA1, and VC(EC) for the vision component that stands in for neocortex and EC (Kowadlo et al., 2020, p. 4). This paper uses the shorter names DG, CA3, PR, CA1 and vision network.

Study and test. In each run, AHA studies the 20 study drawings at once. The fast parts train for 60 steps on that one batch, then are tested on the 20 test drawings, then are reset to fresh random weights before the next run (Kowadlo et al., 2020, p. 6). The vision network does not learn during runs.

Scoring. AHA is never trained to classify. To score a stage, I take the state that stage produces for each study drawing and for each test drawing, compare them by mean squared error (MSE), and pair each study drawing with the test drawing whose state is closest. Accuracy is the share of correct pairings, averaged over runs (Kowadlo et al., 2020, pp. 7–8). The paper calls the vision network the long-term memory (LTM), and scoring its own output gives the baseline it describes as "the LTM on its own" (Kowadlo et al., 2020, p. 7). Scoring PR's output gives AHA's headline 86.4%: the paper credits that figure to PR, which "boosts performance over the baseline significantly by almost 15% to 86.4%" (Kowadlo et al., 2020, p. 8). Scoring CA3 compares each stored pattern with the state CA3 settles into from each test cue; the paper reports that CA3's accuracy "is no better than" the baseline (Kowadlo et al., 2020, p. 8). Throughout this paper, AHA's score means its score at PR, and the memory's gain means AHA's score minus the vision network's score in the same setting.

Table 1: AHA's parts. Sizes from Kowadlo et al. (2020, Table 3, p. 19, and Figure 13, p. 20); where those two sources disagree, Table 2 gives both values and the one used here.

Part	Paper's name	Region	What it does	Size
Vision network	VC(EC)	neocortex, EC	Sparse convolutional autoencoder that learns pen-stroke filters on the background alphabets, then stays fixed. An interest filter keeps its output only near the strongest edges.	121 filters, 10×10; output 20,449
DG	PS(DG)	dentate gyrus	Fixed random layer that keeps the 10 most active of 225 units, giving each drawing its own sparse pattern	225 units, 10 on
CA3	PC(CA3)	CA3	Hopfield network that stores DG's patterns and settles from a cue over 70 iterations	225 units
PR	PR(EC-CA3)	EC-to-CA3 path	Two-layer network trained to output DG's pattern for each vision code	20,449 → 1,000 → 225
CA1	PM(CA1)	CA1	Two-layer network trained to turn a CA3 pattern back into the drawing	225 → 100 → 2,704

3. Methods

3.1. Implementation

I implemented AHA in PyTorch 2.14 from version 5 of its paper (Kowadlo et al., 2020), taking every hyperparameter from its Table 3 (Kowadlo et al., 2020, p. 19): for example DG's 225 units with 10 active, an inhibition decay of 0.95 and a weight knockout rate of 0.25; CA3's gain of 2.7, 20 units updated per step and 70 iterations; PR's learning rate of 0.01 and L2 penalty of 0.000025; and the vision network's 121 filters of 10×10, sparsity of 1 during pretraining and 4 afterwards, and 2,000 pretraining batches of 128 drawings. Drawings are resized from 105×105 to 52×52 pixels. The vision network is a convolutional version of the k-sparse autoencoder, which keeps only the k strongest activities (Makhzani & Frey, 2013), as the paper's Appendix B.1 describes (Kowadlo et al., 2020, p. 17). CA3 sets its weights with the pseudo-inverse rule, which computes every weight from all 20 stored patterns together (Kowadlo et al., 2020, p. 5). Figure 2 shows the filters the vision network learns.

3.2. Details the paper leaves open

Rebuilding AHA from its text meant deciding details that the paper leaves out or states in two ways. Table 2 lists them, with what the paper says, the choice made here, and, where I measured it, how much the choice moves the scores. Where pt-aha (Cerenaut, 2021) settles a detail, I note what it does; I read it at commit f832833 (January 2024).

Table 2: Details the paper leaves open or states in two ways. Page numbers refer to version 5 of Kowadlo et al. (2020).

Detail	What the paper says	Choice here	Effect measured
Centring	Not mentioned. pt-aha shifts each drawing so its centre of ink sits mid-frame; Lake's own baseline also compares centred images (Lake et al., 2015, p. 1335).	Centre each drawing	Without it: vision network −22.4, AHA −23.4 (Section 4.6)

Detail	What the paper says	Choice here	Effect measured
Matching direction	One study ("train") character is presented and the task "is to identify the matching character" among 20 test characters (p. 8), so each study drawing looks for its test partner. pt-aha's configuration scores the vision network and CA3 this way and PR the other way, and its code also scores PR study-first under the comment "this is the version used in the paper".	Study-to-test	Section 4.3
Scale before matching	"Minimum MSE of a given internal state" (p. 8); no scaling stated.	Raw vectors by default; dividing by length tested	Section 4.2
Scale of the vision code fed to DG and PR	Not mentioned. pt-aha divides the vision code by its length before passing it to DG and PR.	Raw by default; scaled version tested	Section 4.2
Pretraining loss	Not named (Appendix B, p. 17). pt-aha uses MSE on a linear output.	Binary cross-entropy; MSE tested	Section 4.4
Lifetime sparsity	Each filter is "trained at least once per mini-batch (i.e. a lifetime of 1 sample per mini-batch)" (p. 17). This could mean one position in one drawing, or a filter's whole response map on its best drawing.	One position in one drawing	Not tested
Interest filter	A mask from difference-of-Gaussians filters is applied, then "a smoothing operation" (pp. 17–18); it does not say whether the smoothing acts on the mask or on the masked vision code. Table 3 gives 20 features (p. 19). pt-aha includes the filter but its configuration leaves it off.	Smooth the masked code; 20 points each from the positive and negative masks, up to 40	Without it: CA3 falls to 18.3% (Section 4.6)
PR hidden layer	1,000 units in the text (p. 6) and Table 3 (p. 19); 800 in Figure 13 (p. 20).	1,000	Not tested
CA1 output size	Table 3 gives 100 output units (p. 19). The text says the output layer "is constrained to the size of VC(EC)" while CA1 is trained to reconstruct the input images (p. 6). Figure 13 gives 2,704, one unit per pixel of a 52×52 image (p. 20).	2,704	Not tested
DG output	$y = M \cdot z$ (p. 5), which keeps the winning units' values.	1 for each winning unit, 0 elsewhere	Not tested; CA3 stores only the sign of the pattern (Appendix A.1, p. 17)

Detail	What the paper says	Choice here	Effect measured
DG inhibition	Inhibition multiplies each unit's input and decays as $\phi_i(t+1) = \phi_i(t) \cdot \gamma$ (p. 5). Read literally, multiplying by $\gamma = 0.95$ makes a unit's input smaller each step, which strengthens inhibition rather than letting it fade, and the value set when a unit fires is not given.	Full inhibition when a unit fires, recovering by a factor of 0.95 with each new drawing	Not tested
CA3 cue offset	Shift the cue so that "at least k bits" are positive (Appendix A.2, p. 17).	Shift only when fewer than k are positive, placing zero between the kth and $(k+1)$th values	Not tested
CA1 training input	Not stated whether CA1 learns from stored CA3 patterns or recalled ones (p. 6).	Stored patterns	Not tested
States compared	"Comparing internal states between memorisation and retrieval patterns" (p. 7).	PR: its output on study drawings after learning against its output on test drawings. CA3: stored pattern against settled state.	–
Convolution padding	Filters 10 wide at stride 5 for pretraining and 1 afterwards (p. 19); padding not stated.	Stride 5: 4 each side. Stride 1: 4 before, 5 after, keeping 52×52	Not tested
Seeds	"Every test is repeated with 10 random seeds" (p. 8); not stated whether pretraining is repeated too.	10 seeds for the fast parts; three pretraining seeds in some rows	Section 4.4
Scope	Also reports an instance task, noise and occlusion curves, and a baseline memory, FastNN (pp. 7–9).	One-shot classification without noise or occlusion only	–

3.3. Protocol

Each setting runs all 20 runs of the one-shot task with 10 seeds, 0 to 9. The seeds set the fast parts' random weights. Unless a row says otherwise, the vision network comes from one pretrained checkpoint, pretraining seed 0, so its own score has no spread across seeds. Rows marked "pretraining seeds 0, 1 and 2" repeat all 10 seeds on each of three vision networks pretrained with different seeds; their ± is one standard deviation across those three networks. Everywhere else, ± is one standard deviation across the 10 seeds. Runs used an Apple M-series GPU through PyTorch's MPS backend.

The default setting follows every value the paper states, with centring, study-to-test matching, raw vectors and binary cross-entropy pretraining. Each other setting changes one or two of the open details in Table 2: matching direction, dividing each vector by its length before matching (normalised matching), passing a normalised vision code to DG and PR, MSE pretraining, the pretraining seed, and, as ablations, turning off centring, turning off the interest filter, and replacing CA3's pseudo-inverse rule with the Hebbian rule, which strengthens a connection in proportion to how often its two units are active together.

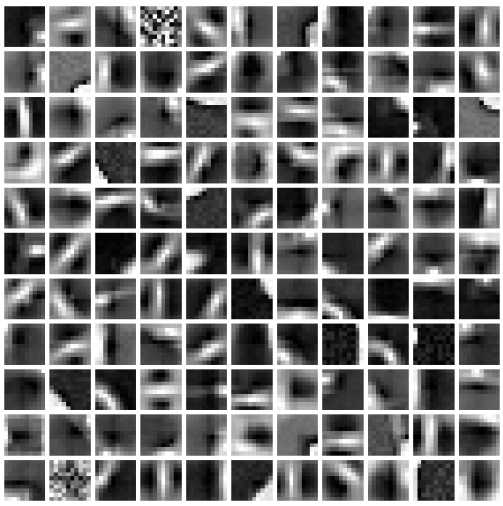


Figure 2: The 121 filters the vision network learns on the background alphabets. Most are pen strokes: edges, curves and junctions, as in the paper's Figure 11 (Kowadlo et al., 2020, p. 18).

Table 3: Accuracy at each stage, default setting, against the paper. Mean ± one standard deviation over 10 seeds. The vision network and raw pixels do not depend on the seed. Paper values from Kowadlo et al. (2020, pp. 8–9). Source: `results/oneshot.json`.

Stage	What it measures	This rebuild	Paper
Raw pixels	Comparison with no model	44.2%	not reported
Vision network alone	The paper's baseline	52.2%	71.6%
PR	AHA's headline score	**64.7 ± 1.2%**	**86.4%**
CA3, after settling	Stored against settled pattern	61.6 ± 0.9%	"no better than" baseline
CA1	Recalled image	61.9 ± 0.7%	not reported

The target is the paper's two reported scores, 71.6% for the vision network alone and 86.4% for AHA. Omniglot's one-shot runs have no separate validation set, so I could only compare settings on the test runs themselves. The best setting reported below was therefore chosen by looking at test scores, and its score is an upper estimate of what these options reach, not a held-out result.

4. Results

4.1. With the paper's stated settings and centring, AHA scores 64.7% and its vision network 52.2%

Table 3 and Figure 3 give the default setting's score at every stage. AHA scores 64.7% against the paper's 86.4%, and the vision network alone scores 52.2% against 71.6%. The hippocampal parts add 12.5 points over the vision network, where the paper reports 14.8. Comparing raw pixels, with no model at all, scores 44.2%. Scored after CA3 settles, AHA gives 61.6%, below its PR score; scored at CA1's recalled image, 61.9%. Figure 4 shows the signals for one run.

4.2. Normalised matching raises both scores, the vision network's more

Table 4 and Figure 5 give every setting. Normalised matching changes the scores most. It raises the vision network from 52.2% to 63.0%, a gain of 10.8 points, and AHA from 64.7% to 70.6%, a gain of 5.9. Because the vision network gains more, the memory's contribution shrinks from 12.5 points to 7.6. A likely reason

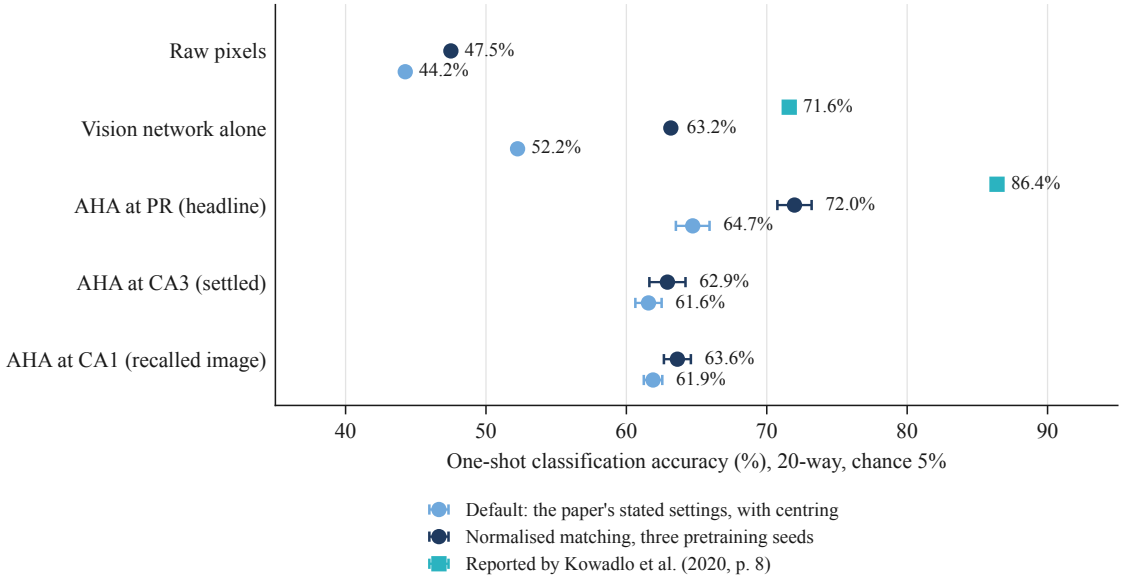


Figure 3: One-shot accuracy at each stage. Light points: default setting. Dark points: normalised matching, averaged over three pretrained vision networks (Section 4.4). Squares: the two scores the paper reports. Error bars show one standard deviation over seeds (light) or over pretrained networks (dark). Chance is 5%.

is that, at raw scale, a drawing with more ink produces a longer vision vector, and that length can outweigh which strokes are present when two vectors are compared; dividing by length removes it.

Passing a normalised vision code into DG and PR, as pt-aha does, raises AHA to 66.0% at raw-scale matching and changes little once matching is also normalised (70.3% against 70.6%). DG ranks its inputs within each drawing and has no bias, so it picks the same winning units either way; only PR sees a different input.

With normalised matching and the checkpoint used in every single-network row, AHA scores 70.6% and the vision network alone 63.0%. In that setting AHA falls 15.8 points short of the paper, and 8.6 of those points are already missing from the vision network.

4.3. Matching direction moves the vision network and AHA in opposite directions

Pairing each test drawing with its closest study drawing, instead of the reverse, raises the vision network from 52.2% to 56.0% and lowers AHA from 64.7% to 61.3%. With normalised matching the effect is larger: the vision network reaches 67.3%, its highest score in any setting, while AHA falls to 63.2%, below the vision network, so the memory's contribution becomes negative (-4.1 points). The two directions can differ because a test drawing can be the closest match for two study drawings at once, or none. pt-aha's configuration mixes the two directions, scoring the vision network study-to-test and PR test-to-study; under that mix with normalised matching, the scores here would be 63.0% for the vision network and 63.2% for AHA.

4.4. Pretraining with MSE lowers every stage after the vision network, and the pretraining seed moves AHA by up to 3.8 points

Pretraining the vision network with MSE on a linear output, pt-aha's choice, leaves the vision network's own score nearly unchanged (54.6% against 54.4% across three pretraining seeds) but lowers AHA from 67.1% to 60.2% and CA3 from 63.0% to 53.6%. With binary cross-entropy, AHA scores 64.7%, 68.5% and 68.0% on

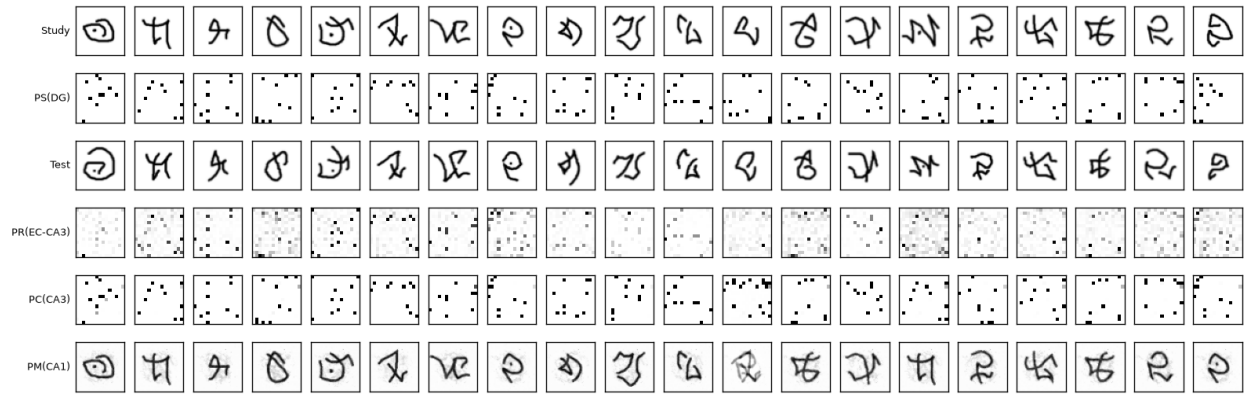


Figure 4: Signals through AHA on one-shot run 1, default setting. Rows from top: the 20 study drawings, the DG pattern stored for each, the test drawings by a different person, PR's output for each test drawing, the state CA3 settles into, and the image CA1 recalls. Each column is one character; 16 of the 20 recalled images show the right character.

vision networks pretrained with seeds 0, 1 and 2, and the vision network alone 52.2%, 55.0% and 56.0%. The checkpoint used for every single-network row, seed 0, gives the lowest score of the three. With normalised matching, the three networks give AHA 70.6%, 72.4% and 72.9%, a mean of 72.0% and the highest AHA score in this paper.

4.5. After CA3 settles, AHA is no better than its vision network, as the paper reports

With normalised matching, CA3 scores 61.5%, below the vision network's 63.0%, and across three pretrained networks 62.9% against 63.2%. This agrees with the paper's statement that CA3's accuracy is "no better than" the baseline (Kowadlo et al., 2020, p. 8). At raw scale CA3 scores 61.6%, above the weaker raw-scale vision network but below PR. In every setting with study-to-test matching, PR scores higher than CA3.

4.6. Centring and the interest filter hold up the scores, and Hebbian learning leaves CA3 at chance

Table 5 gives three ablations of the default setting. Without centring, with a vision network also pretrained without it, the vision network falls by 22.4 points to 29.8% and AHA by 23.4 points to 41.3%. Without the interest filter, the vision network barely changes (53.8%), but CA3 falls to 18.3% and CA1 to 26.1%. With the Hebbian rule in place of the pseudo-inverse rule, CA3 and CA1 score 5.8%, which is chance: DG's patterns each have 10 of 225 units on, so any two stored patterns agree on most of their inactive units, and Hebbian weights cannot keep patterns that similar apart.

4.7. Results match across devices

One seed of the default setting run on the CPU gives the same accuracy as the same seed on the GPU at every stage except CA1, where one of the 400 pairings differs (`results/cpu_seed0.json`).

5. Discussion

The paper's claim that AHA's hippocampal parts add accuracy over the vision network holds in the rebuild, in every setting that matches drawings study-to-test as the paper describes and keeps the interest filter: the gain ranges from 3.0 to 12.7 points, against the paper's 14.8. The paper's second observation about the one-shot task also holds: after CA3 settles, AHA is no better than its vision network. What does not reproduce is the

Table 4: Accuracy (%) in every setting. Gain is AHA's score at PR minus the vision network's score in the same row. Each row runs 10 seeds; rows marked "seeds 0, 1 and 2" also vary the pretrained vision network, and their ± is the spread across the three networks. BCE: binary cross-entropy. Source: `results/grid.csv`.

Setting	Matching	Vision	AHA (PR)	CA3	CA1	Gain
Paper (Kowadlo et al., 2020, p. 8)		71.6	86.4			14.8
Default	study-to-test	52.2	64.7 ± 1.2	61.6 ± 0.9	61.9 ± 0.7	12.5
Test-to-study matching	test-to-study	56.0	61.3 ± 0.9	62.5 ± 0.8	61.5 ± 0.9	5.3
Normalised matching	study-to-test	63.0	70.6 ± 1.4	61.5 ± 0.7	62.5 ± 1.1	7.6
Normalised matching	test-to-study	67.3	63.2 ± 1.2	62.5 ± 0.8	61.7 ± 0.8	−4.1
Normalised code into DG and PR	study-to-test	63.0	66.0 ± 1.1	61.5 ± 1.4	61.3 ± 0.9	3.0
and normalised matching	study-to-test	63.0	70.3 ± 1.5	61.3 ± 1.4	61.9 ± 1.0	7.3
Default, seeds 0, 1 and 2	study-to-test	54.4 ± 1.9	67.1 ± 2.0	63.0 ± 1.3	63.0 ± 1.0	12.7
Normalised matching, seeds 0, 1 and 2	study-to-test	63.2 ± 0.1	72.0 ± 1.2	62.9 ± 1.3	63.6 ± 1.0	8.8
MSE pretraining	study-to-test	54.0	59.3 ± 1.7	53.3 ± 1.9	54.4 ± 1.4	5.3
normalised matching	study-to-test	59.0	63.8 ± 1.2	53.2 ± 1.7	54.5 ± 1.3	4.8
seeds 0, 1 and 2	study-to-test	54.6 ± 1.7	60.2 ± 1.2	53.6 ± 1.6	54.6 ± 1.9	5.6
no interest filter	study-to-test	57.7	57.1 ± 1.6	23.2 ± 2.6	30.0 ± 1.6	−0.6
normalised code and matching	study-to-test	59.0	63.9 ± 1.5	53.2 ± 2.1	54.3 ± 1.6	4.9

Table 5: Ablations of the default setting (study-to-test matching, raw vectors, 10 seeds). Source: `results/ablation_*.json`.

Setting	Vision	AHA (PR)	CA3	CA1
Default	52.2	64.7 ± 1.2	61.6 ± 0.9	61.9 ± 0.7
No centring	29.8	41.3 ± 1.5	38.1 ± 1.1	37.6 ± 0.9
No interest filter	53.8	52.5 ± 1.0	18.3 ± 1.6	26.1 ± 1.7
Hebbian rule in CA3	52.2	64.7 ± 1.2	5.8 ± 0.5	5.8 ± 0.6

level of either score. The rebuilt vision network reaches 63.0% with the paper's matching direction against 71.6% reported, and AHA reaches 70.6% to 72.0% against 86.4%.

The size of the memory's gain depends on how states are compared. At raw scale the gain is 12.5 points, close to the paper's. Once vectors are divided by their length, the vision network improves by 10.8 points and AHA by 5.9, and the gain falls to 7.6. The paper does not say how vectors are scaled before comparison, so its 14.8 points cannot be placed on either side of this choice from the text. A reader comparing a hippocampal model against its own input network should report the scaling used for both, because the scaling alone can halve the measured benefit.

Most of the remaining shortfall is likely to lie in the vision network. No setting tested here lifts it above 67.3%, and that score comes only with test-to-study matching, the direction the paper does not describe. Its training is the least specified part of AHA: the paper names neither the pretraining loss nor the exact reading of its lifetime sparsity rule, and the interest filter that shapes the vision code is described in one short paragraph. Each of these could change the filters the network learns, and so the score of every stage after it. The untested readings in Table 2 are the first places a further attempt should look. The two outright conflicts in the paper, the size of PR's hidden layer and CA1's output size, sit in the fast parts and are less likely to explain a shortfall that is already present before those parts act.

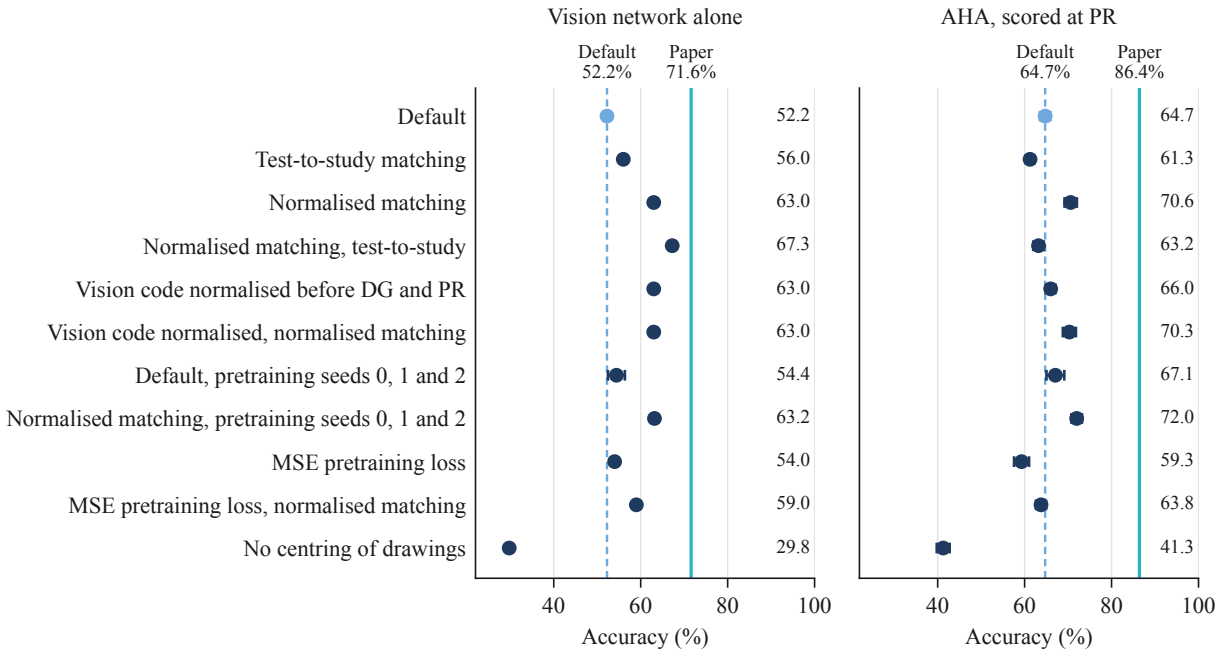


Figure 5: Accuracy of the vision network alone (left) and of AHA scored at PR (right) in each setting. The dashed line marks the default setting and the solid line the paper's score. Rows are described in Section 3; values match Table 4.

Centring and the interest filter show how much depends on steps outside the hippocampal parts. Centring is never mentioned in the paper, yet without it both scores fall by more than 20 points; Lake and colleagues' own baseline also compares centred images (Lake et al., 2015, p. 1335), which suggests the step is common practice in this task rather than a detail particular to AHA. The interest filter barely changes the vision network's own score, but without it CA3 falls to 18.3%, so its main effect is on the patterns DG and CA3 receive.

This study has limits. It covers one task, one-shot classification without noise or occlusion; the paper's instance task, its corruption curves and its FastNN baseline are not rebuilt. The best setting was chosen on the test runs, because the task provides no other data to choose on. Several readings of the paper were not tested, as Table 2 marks. And this is one rebuild by one person: a second independent rebuild, or the code and checkpoints behind the paper's numbers, would show which of the remaining differences come from the paper's text and which from choices made here.

6. Related work

pt-aha (Cerenaut, 2021) is the authors' later PyTorch implementation of AHA. It differs from the paper's text in the ways Table 2 notes: it centres drawings, divides the vision code by its length before DG and PR, pretrains the vision network with MSE, leaves the interest filter off and mixes matching directions. It shows what the authors chose later; it does not by itself show which settings produced the paper's 86.4%.

7. Conclusion

Rebuilt from the text of its paper, AHA scores 64.7% on the one-shot Omniglot task, and 70.6% with normalised matching, against the 86.4% reported. Its hippocampal parts still add accuracy over its vision network, by 7.6 to 12.5 points depending on how vectors are scaled before comparison, against the paper's 14.8. More than half of AHA's shortfall in the best setting is already present in the vision network alone. The details that move the scores most, centring, scaling before comparison and matching direction, are absent from the paper or described in a single sentence.

Code and data availability

Code, tests and result files are at https://github.com/sachio222/aha4 (commit 446d6db), under the MIT license. `evaluate.py` produces each result file in `results/`; `scripts/run_grid.sh` pretrains the extra vision networks and reruns every setting in this paper, in about 85 minutes on an Apple M-series laptop; `scripts/make_tables.py` rebuilds every table from the result files. A pretrained vision network ships in `checkpoints/`. Omniglot and its 20 one-shot runs (Lake, 2015) download on first use. The figures in this paper are built from the same result files by `figures/make_figures.py` in the paper's source folder.

Acknowledgements

Thanks to Gideon Kowadlo and David Rawlinson for discussions about AHA.

References

Cerenaut. (2021). *pt-aha: AHA implemented in PyTorch* [Computer software]. GitHub. https://github.com/Cerenaut/pt-aha

Kowadlo, G., Ahmed, A., & Rawlinson, D. (2020). *AHA! an 'Artificial Hippocampal Algorithm' for episodic machine learning* [Version 5]. arXiv. https://arxiv.org/abs/1909.10340v5

Lake, B. M. (2015). *Omniglot data set for one-shot learning* [Data set]. GitHub. https://github.com/brendenlake/omniglot

Lake, B. M., Salakhutdinov, R., & Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, *350*(6266), 1332–1338. https://doi.org/10.1126/science.aab3050

Makhzani, A., & Frey, B. (2013). *k-Sparse autoencoders*. arXiv. https://arxiv.org/abs/1312.5663