TIME WILL GIVE YOU THE ANSWER: CONTROLLABLE FACTORIAL PERMUTATIONS IN SPIKING NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The brain works through spiking neural networks. In this paper, we investigate the relative order of the first spikes in spiking neurons. The relative order of first spikes across neurons can emit a crazy density of information. If we read by which spiking neuron fires or not, M neurons represent $2^M - 1$ states, but if we read by the full first-firing relative order, they can represent $M!$ discrete states. If M is 10, it's a comparison of 1 thousand to 3.6 million, and the ratio grows even more as M grows.

We show the ordering space is controllable and trainable. By updating input current or through intrinsic parameters alone—membrane time constant, firing threshold, and membrane resistance—we can reach all permutations. We also design an optimizer that moves the neurons into any target order by updating each neuron's membrane time constant.

We picked a 30-class dataset. In a 6-spiking-neuron system, we give each dataset class a set relative ordering of 2 neurons. For example, a set (A, B) means we want neuron A to fire first and neuron B to fire second; neurons firing after the second are ignored. We reach an accuracy of $27.7\% \pm 1.03$ across 5 different seeds, compared to 3.33% random chance.

1 INTRODUCTION

The rate or count of a spiking neuron is mostly how we utilize and apply it. But we come up with a question: what if, instead of rate, we look at the relative ordering of the spiking neurons?

Before being able to utilize the relative ordering, we successfully make all relative orderings in a 6-neuron system reachable by adjusting the parameters in each spiking neuron, which include membrane time constant, firing threshold, and membrane resistance.

Afterwards, we come up with an optimizer to update spiking neurons' parameters to shift the relative order from random input current to the target relative order, allowing us to control the network by only providing the input current and target relative order.

On a 30-class classification task, we used a 6-neuron system and gave all 30 classes their own target relative ordering of 2 neurons in class-index order. In this case, the first neuron has 6 choices, and the second neuron has 5 choices aside from the first one, giving 30 possibilities in top-2 neuron sets. In this experiment, we reach only 11.9% accuracy, with the issue being that similar relative orderings, such as $(1, 2)$ and $(1, 3)$, need to serve similar dataset classes. After manually giving similar classes similar relative orderings, we get $27.7\% \pm 1.03$ as the result, proving that relative order is a usable method for machine learning tasks.

Contributions.

- **Controllability of relative ordering in spiking neurons.** Able to reach any target relative ordering through input current or parameters of spiking neurons, including membrane time constant, firing threshold, or membrane resistance alone (Section 4).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- **An optimizer for relative ordering.** A trained optimizer that only has two constants. By updating only the membrane time constants, it moves the neurons from a random starting order to any target relative ordering in a 6-neuron system, and can transfer to 7- and 8-neuron systems without retraining (Section 4).
- **Data class similarities matter.** With everything else fixed, changing the given relative ordering alone increased accuracy from 11.9% to 27.7% (Section 6).

2 Related Work

Temporal coding. Comşa et al. (Comsa et al., 2020) trained spiking neural networks using spike timing, where the first output neuron to fire determines the prediction. We go beyond the first spike and use the relative ordering of several neurons as the representation.

Rank order coding. Thorpe and Gautrais (Thorpe & Gautrais, 1998; Gautrais & Thorpe, 1998) proposed that information can be carried by the order in which neurons fire, and noted that M neurons have $M!$ possible orders. VanRullen and Thorpe (VanRullen & Thorpe, 2001) showed that the first spikes of retinal ganglion cells, read in order, carry enough information to reconstruct an image. We build on this idea. We show that every order can be reached by changing spiking-neuron parameters, give an optimizer that reaches any target order, and use the order of the first two spikes as the output of a classifier.

Spike-driven Transformers. Yao et al. (Yao et al., 2023) introduced Spike-driven Transformer, showing that spiking neural networks can be integrated into Transformer architectures. Their approach focuses on efficient spike-based computation, while we focus on the information carried by relative order.

3 Method

3.1 First-spike time to relative ordering

Each spiking neuron is a leaky integrate-and-fire (LIF) neuron with membrane time constant τ_i, membrane resistance R_i, firing threshold $V_{\mathrm{th},i}$ and constant input current I_i, starting from $V_i(0) = 0$. Its first spike time has a closed form:

$$\tau_i \frac{dV_i}{dt} = -V_i + R_i I_i, \qquad t_i^* = -\tau_i \log\left(1 - \frac{V_{\mathrm{th},i}}{R_i I_i}\right). \tag{1}$$

The neuron fires only if $R_i I_i > V_{\mathrm{th},i}$. A neuron may fire again after reset, but we use only its first spike. Its firing time is computed directly from Equation (1), without time-step simulation, so spike times are exact.

The state we study is the firing order,

$$\pi = \mathrm{argsort}\left(t_1^*, \ldots, t_M^*\right) \in S_M, \tag{2}$$

which lists neurons from earliest to latest firing, with ties broken by neuron index.

The order depends only on which neuron fires before another, not on the absolute spike times. When all neurons share the same τ, R, and V_{th}, a larger input current always produces an earlier spike. Therefore,

$$\mathrm{argsort}(t^*) = \mathrm{argsort}(-I),$$

so the firing order is simply the ranking of the input currents. To test whether neuron dynamics can produce more than this simple ranking, we also consider neuron-specific τ, R, and V_{th}, where the firing order is no longer determined by the current ranking alone.

3.2 Relative ordering of k neurons

We can represent information using the first k neurons to fire. By focusing only on the first k neurons, we do not need to consider the full $M!$ relative ordering, reducing both the code space and

computation. For M neurons, the number of possible ordered prefixes is

$$P(M,k) = \frac{M!}{(M-k)!}. \tag{3}$$

For $M = 6$, this gives 6, 30, 120, and 720 codes for $k = 1, 2, 3, 4$. We focus on $k = 1$ and $k = 2$. When $k = 2$, each code is an *ordered pair* of neurons.

The $k = 2$ code naturally has two levels. The first spike selects one of 6 groups, and the second spike selects one of 5 possible classes within that group. Therefore, assigning classes to ordered pairs decides which classes share the same first spike. We test different assignments to study how this grouping affects classification.

3.3 Objectives

All objectives are margin losses on spike-time gaps, smoothed with a softplus so that they are differentiable everywhere. Throughout, $\delta = 0.1$ is the required gap and T is the smoothing temperature.

For *control*, given a target order π^*, let $g_k = t^*_{\pi^*_{k+1}} - t^*_{\pi^*_k}$ be the gap between the k-th and $(k+1)$-th neuron of the target. The target is met strictly when every gap is at least δ. Only the $M-1$ adjacent gaps are needed, because they fix the whole order:

$$L_{\text{ctrl}} = \frac{1}{M-1} \sum_{k=1}^{M-1} T \,\text{softplus}\left(\frac{\delta - g_k}{T}\right). \tag{4}$$

For *classification*, let (a, b) be the ordered pair assigned to the true class: neuron a must fire first and b second. L_1 pushes every other neuron at least δ behind a; L_2 pushes every neuron other than a and b at least δ behind b:

$$L_1 = \frac{1}{M-1} \sum_{j \neq a} T \,\text{softplus}\left(\frac{\delta - (t^*_j - t^*_a)}{T}\right), \quad L_2 = \frac{1}{M-2} \sum_{j \notin \{a,b\}} T \,\text{softplus}\left(\frac{\delta - (t^*_j - t^*_b)}{T}\right). \tag{5}$$

We minimize $L = \frac{1}{2}(L_1 + L_2)$ with $\delta = 0.1$ and $T = 0.05$. Both terms are active from the first step; there is no staged schedule. First-rank-only classification uses L_1 alone, with the true class as a.

Inputs set the currents through $u_i = W_i x + b_i$ and $I_i = 1.05 + 1.95\,\sigma(u_i)$, which keeps every neuron above threshold and bounded. Gradients flow $L \to t^* \to I \to (W, b)$. No classifier, decoder or embedding follows the spiking population: the prediction is read off the spike order.

3.4 The order optimizer

The optimizer moves the neurons into a target order π^* by changing only their membrane time constants. In this setting, I, R and V_{th} are fixed with $R_i I_i = 2V_{\text{th},i}$, so Equation (1) gives $t^*_i = \tau_i \ln 2$.

At each step, it computes the gap g_k between the k-th and $(k+1)$-th neuron of the target, as in L_{ctrl}, and turns each gap into a soft error

$$q_k = \sigma\left(\frac{\delta - g_k}{T_{\text{opt}}}\right), \qquad k = 1, \ldots, M-1. \tag{6}$$

q_k is close to 0 when the pair is in order by at least δ, and close to 1 when it is reversed. Each pair pushes its earlier neuron earlier and its later neuron later, so the neuron at target position k is updated by

$$\tau_{\pi^*_k} \leftarrow \text{clip}\Big(\tau_{\pi^*_k} + \eta\,(q_{k-1} - q_k),\ \tau_{\min},\ \tau_{\max}\Big), \qquad q_0 = q_M = 0, \tag{7}$$

with $\tau \in [0.25, 2.5]$. The first neuron is only pulled earlier, the last only pushed later, and a middle neuron is moved by the difference between the pair in front of it and the pair behind it. The optimizer stops when every $g_k \geq \delta$.

The optimizer has two constants, the step size η and the softness T_{opt}, shared by all neurons and all targets. We fit them with Adam by running the update for a fixed number of steps and minimizing L_{ctrl} over 500 of the 720 target orders, which gives $\eta = 0.1779$ and $T_{\text{opt}} = 0.1068$. They are then frozen. Nothing in the update depends on M, so the same constants apply to any number of neurons.

4 THE FACTORIAL STATE SPACE IS CONTROLLABLE

We first ask whether any target firing order can be produced on demand. For $M = 6$ neurons, there are 720 possible relative orderings. We test every one and find that all 720 orders can be reached.

Every firing order is reachable. Given a target relative ordering, we optimize the neuron parameters until the neurons fire in the desired order. Starting from random states, all 720 target orders were reached. We also started from each possible ordering and targeted every other ordering. All $518{,}400$ transitions succeeded, showing that the system can move between any pair of states.

The firing order can be controlled through different neuron parameters. Changing only the input current I, membrane time constant τ, firing threshold V_{th}, or membrane resistance R is sufficient to reach every target order. The same result also holds with recurrent coupling between neurons. Full experimental settings and robustness tests are given in Appendix C.

A simple optimizer. We then ask whether the same task can be solved with a simple rule designed around relative order. Our optimizer compares neighboring neurons in the target order. If two neurons are reversed or too close in firing time, their parameters are adjusted so that the neuron that should fire earlier moves earlier and the neuron that should fire later moves later. This process continues until every neighboring pair is correctly ordered with a minimum separation δ.

The optimizer changes only the membrane time constants τ. It keeps no state between steps and uses only two constants: a step size and a softness parameter (Section 3.4).

Generalization to larger populations. We then increase the population size without changing or retraining the optimizer. The same two constants reach all $5{,}040$ possible permutations of seven neurons and all $40{,}320$ possible permutations of eight neurons.

The optimizer works by comparing neighboring neurons, so the same rule can be applied to populations of different sizes. Our experiments exhaustively verify this for six, seven, and eight neurons.

The optimizer is given the desired target order at every step; it does not decide which order should be produced. These experiments therefore show that relative firing order is a controllable state space and that a simple control rule can generalize across target orders and population sizes.

5 SPIKE ORDER COMPUTES DIRECTLY

We now test whether the first neuron to fire can directly represent the predicted class. With one neuron assigned to each class, the prediction is simply

$$\hat{y} = \arg\min_i t_i^*. \tag{8}$$

No additional classifier or decoder is used after the spiking neurons.

Synthetic classification. On a six-class synthetic dataset, six neurons reach 86.4% accuracy, compared with 16.67 chance. The neurons therefore directly produce the prediction through their firing order.

MotionSense. We also test the same method on MotionSense (Malekzadeh et al., 2019), using accelerometer and gyroscope data with participants separated between training and test sets. Six neurons reach 44.1% balanced accuracy, compared with 16.67% chance.

The prediction is produced directly by the spiking neurons: whichever neuron fires first determines the class. This shows that firing order can be used directly for classification without an additional readout.

Table 1: Classification using the first neuron to fire. Six neurons are used for six classes, with no classifier or decoder after the spiking neurons. Results are averaged over 5 seeds.

Task	Method	Accuracy (↑)	Balanced (↑)
Synthetic, six classes	chance	16.67	16.67
	first spike (ours)	**86.367**	—
MotionSense, six activities	chance	16.67	16.67
	train-majority class	22.12	—
	raw features, linear	50.02	37.59
	first spike (ours)	**54.27**	**44.12**

6 READING TWO SPIKES OF THE FACTORIAL ORDER

Using only the first neuron to fire gives 6 possible outputs. If we instead use the first two neurons in order, the number of possible outputs becomes

$$P(6, 2) = 30.$$

For six neurons, this gives 30 ordered pairs, allowing us to represent 30 classes. We use a balanced 30-class subset of EMNIST (Cohen et al., 2017).

Classifying with two spikes. We assign each of the 30 classes to one ordered pair. For a target pair (A, B), neuron A should fire first and neuron B second. The remaining neurons are ignored.

With only the first spike, six neurons can distinguish at most six of the 30 classes. For a balanced dataset, the maximum possible accuracy is therefore

$$100 \times \frac{6}{30} = 20\%.$$

Using two spikes allows all 30 classes to have their own output.

The class assignment matters. For two-spike ordering, the first neuron divides the classes into groups, and the second neuron distinguishes classes within each group. Therefore, which classes share the same first neuron may affect how easily the network can learn the ordering.

We test three ways of assigning the 30 classes to the 30 ordered pairs. *Lexicographic* assigns them by class index. *Random balanced* assigns them randomly. *Geometry-aligned* groups similar classes together, so similar classes share the same first neuron. To build it, we compute the centroid μ_c of each class on the training split and split the 30 classes into 6 groups of five with small within-group distance $\|\mu_c - \mu_{c'}\|_2^2$. Each group gets one first neuron, and the five classes in the group get the five possible second neurons. The assignment is fixed before training and uses only the training split.

Only the class-to-pair assignment is changed between these experiments; the dataset, network, loss, and training settings remain the same.

Results. The lexicographic assignment reaches 11.9% accuracy, while the random balanced assignments reach 14.5% on average. The geometry-aligned assignment reaches

$$27.7\% \pm 1.0,$$

compared with $1/30$ random chance.

Changing only the assignment therefore increases accuracy from 11.9% to 27.7%. This shows that having more possible firing orders is not enough by itself; how the classes are mapped to those orders also matters.

Where the improvement comes from. Top-2 accuracy requires both the first and second neurons to be correct. The geometry-aligned assignment mainly improves the first spike, while the accuracy of the second spike given a correct first spike remains similar to the random assignments. This

Table 2: Different class-to-pair assignments on balanced 30-class EMNIST using six neurons and the first two spikes.

Assignment	Top-2 (↑)	Rank-1	$\mathrm{Acc}_{2\|1}$
Lexicographic	11.878± 0.49	30.707	38.683
Random balanced (n=10)	14.456± 1.53	33.407	43.298
Geometry-aligned (ours)	**27.682± 1.03**	59.863	46.241

matches the purpose of the geometry-aligned assignment: similar classes are grouped under the same first neuron and then separated by the second.

Together, these results show that relative firing order can represent more classes than the number of neurons, but the structure of the mapping between classes and firing orders strongly affects how well those states can be learned.

7 Limitations

We design the class assignment. The geometry-aligned assignment is built from training-set class centroids, so it uses labels. We also tried to let the network find a useful assignment without labels, and it failed (Appendix E).

We use 30 of the 720 orders. Six neurons have 720 relative orderings, but classification reads only the first two spikes, which gives 30 ordered pairs.

Classification uses shared neuron parameters. In the classification model, all neurons share τ, R, and V_{th}, so the firing order is the ranking of the input currents (Section 3). We have not tested whether neuron-specific parameters improve classification.

Accuracy is low. A 30-output linear model reaches 73.1% on the same task with $23{,}550$ parameters, while our six neurons use $4{,}710$. Our goal is to show that relative ordering can carry class information.

The optimizer is given the target. The optimizer receives the target order at every step. It does not decide which order to produce.

Other datasets. On a ten-class Fashion-MNIST task, two spikes did not beat the first-spike ceiling. On KMNIST, training the firing threshold did not give a reliable gain across seeds. Both are in Appendix E. All results use 5 seeds.

8 Conclusion

We studied the relative order of first spikes. With M neurons, there are $M!$ possible orders. In a 6-neuron system, every order can be reached by changing the input current, τ, V_{th}, or R alone. Our optimizer has only two constants and reaches every order of 7 and 8 neurons without retraining.

Reading the first two spikes of six neurons, we classify 30 classes at 27.7%, above the 20% limit of reading only the first spike. How classes are assigned to orders matters: changing only the assignment moved accuracy from 11.9% to 27.7%. Finding a good assignment without labels is the next step.

Ethics Statement

We use MotionSense, which was collected from human participants. We use only the released, de-identified sensor data, and no participant appears in both training and test sets. The other experiments use public image datasets (EMNIST (Cohen et al., 2017), Fashion-MNIST (Xiao et al., 2017), KMNIST (Clanuwat et al., 2018), STL-10 (Coates et al., 2011)) and synthetic data.

Reproducibility Statement

Every experiment was run under a protocol fixed before the run. Each result directory keeps its configuration, data splits, model files, and hashes of all of them, and a replay script regenerates the spike times and predictions from the saved models. Protocols, hyperparameters, and every preregistered gate, including failed ones, are in Appendices B and E. An error we found in an earlier run and its correction are in Appendix F. A script re-derives every number in the paper from the saved results. We will release the code and this script.

LLM Usage Statement

We used LLM-based coding assistants (Claude and OpenAI Codex) in this work. They helped implement the experiments and the analysis code, launch and monitor runs on a compute cluster, write the script that checks every number in the paper against the saved results, check references, and give feedback on experimental design. They also drafted an earlier version of the paper; the authors rewrote the paper, and LLMs edited parts of the final text (the limitations, conclusion, statements, and the description of the optimizer). The research question and the choice of experiments are the authors'. The authors checked all code, results, and text, every number is verified against the saved results by the script above, and the authors take full responsibility for the content.

References

Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical Japanese literature. *arXiv preprint arXiv:1812.01718*, 2018. URL https://arxiv.org/abs/1812.01718.

Adam Coates, Andrew Y. Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, 2011. URL https://proceedings.mlr.press/v15/coates11a.html.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: Extending MNIST to handwritten letters. In *International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, 2017. doi: 10.1109/IJCNN.2017.7966217.

Iulia M. Comsa, Krzysztof Potempa, Luca Versari, Thomas Fischbacher, Andrea Gesmundo, and Jyrki Alakuijala. Temporal coding in spiking neural networks with alpha synaptic function. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8529–8533, 2020. doi: 10.1109/ICASSP40776.2020.9053856.

Jacques Gautrais and Simon Thorpe. Rate coding versus temporal order coding: A theoretical approach. *BioSystems*, 48(1–3):57–65, 1998. doi: 10.1016/S0303-2647(98)00050-1.

Mohammad Malekzadeh, Richard G. Clegg, Andrea Cavallaro, and Hamed Haddadi. Mobile sensor data anonymization. In *International Conference on Internet of Things Design and Implementation (IoTDI)*, pp. 49–58, 2019. doi: 10.1145/3302505.3310068.

Simon Thorpe and Jacques Gautrais. Rank order coding. In James M. Bower (ed.), *Computational Neuroscience: Trends in Research*, pp. 113–118. Springer US, Boston, MA, 1998. doi: 10.1007/978-1-4615-4831-7_19.

Rufin VanRullen and Simon J. Thorpe. Rate coding versus temporal order coding: What the retinal ganglion cells tell the visual cortex. *Neural Computation*, 13(6):1255–1283, 2001.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. URL https://arxiv.org/abs/1708.07747.

Man Yao, Jiakui Hu, Zhaokun Zhou, Li Yuan, Yonghong Tian, Bo Xu, and Guoqi Li. Spike-driven transformer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://arxiv.org/abs/2307.01694.

A Exact First-Spike Time and Parameter Sensitivities

Derivation of Equation (1) from the membrane equation, the existence condition $RI > V_{\mathrm{th}}$, the monotonicity of t^* in each of I, τ, R and V_{th}, and the resulting sensitivity of the induced order to each parameter.

B Protocols, Hyperparameters, and Preregistered Gates

Complete protocol for every experiment reported in the main text: dataset and split construction, network size, current map, loss, optimizer settings, seeds, epoch counts, and the full list of pre-registered gates with their PASS/FAIL outcomes. Gates that failed are listed alongside gates that passed.

C Full Controllability Results

Per-target and per-transition statistics behind Section 4, including the full step-count distributions, the per-variable control results, the coupled-neuron arm, and the complete noise-robustness sweep. With recurrent coupling between neurons, every target was reached in $2{,}160$ tests per arm at a median of 42 steps. Under multiplicative noise on the frozen parameters the order is far more durable than the margin that produced it: at 1% noise the order survives in 100.0% of cases and the full δ margin in 57.4%; at 5%, 89.8% and 17.3%; at 10%, 51.6% and 7.4%. Broken orders at 10% noise were restored by target-aware re-optimization in $83{,}210$ of $83{,}215$ cases at a median of 4 steps. Before its two constants were learned, the optimizer (Section 4) with hand-chosen constants reached every target in $14{,}400$ of $14{,}400$ runs at a median of 8 steps.

The optimizer is defined for any M, but a δ-margin order must fit inside the range of spike times that the clipped time constants allow. With $\tau \in [0.25, 2.5]$ and t^* proportional to τ, all spike times lie in a window of width 1.56, and $M - 1$ gaps of at least δ fit only for $M \leq 16$. The slack left inside this window shrinks as M grows, from 1.06 at six neurons to 0.86 at eight. Larger populations will therefore need a wider τ range or a smaller δ, and may need the two constants to be refit.

D Rank Interference: Validation-Split Diagnostics

These diagnostics were run on a held-out validation split of KMNIST (Clanuwat et al., 2018). The official KMNIST test split was never opened for them, and we report them as mechanism diagnostics, not as test results.

They ask a narrow question: is the second spike learnable in its own right, and does supervising it interact with the first? Supervising only the first rank gives rank-1 accuracy 68.590% and conditional second-rank accuracy 54.497%. Supervising only the second rank reverses this, to 53.992% and 67.730%. The second identity is therefore learnable on its own, and the two ranks compete: moving supervision from the first rank to the second costs $-14.598\,\mathrm{pp}$ of rank-1 accuracy and buys $+13.231\,\mathrm{pp}$ of conditional second-rank accuracy. Supervising both simultaneously, as the main text does, lands between the two extremes (45.122% and 63.419%).

Training the two ranks in stages rather than simultaneously does not resolve the competition: a second stage reaches 72.348% and 32.587%, worse on the joint metric than simultaneous training, and the same pattern appears when the first rank is locked in place, softly (33.662% and 78.789%) or hard (75.172% and 21.537%).

E Negative Results

We report every preregistered outcome, including the ones that failed. Four are relevant to how the main claims should be read.

E.1 Temporal memory in the state is not usable order

A delayed-recall task with neuron-specific membrane time constants did carry information about a past stimulus in the continuous membrane state, above chance. But the induced *ordering* collapsed onto a handful of distinct permutations. Information surviving in the continuous state is not the same as information surviving in the factorial code, and this experiment is the reason we report occupied-permutation counts everywhere else in the paper.

E.2 A ten-class top-2 problem did not clear its ceiling

On Fashion-MNIST (Xiao et al., 2017), a ten-class clothing benchmark, with a fixed ten-pair code, the first-identity ceiling is 60%. Top-2 accuracy reached 59.94% with a bootstrap lower bound of 58.97%, which does not clear the ceiling. The preregistered gate failed and we did not retune. This is part of why the main result is stated for a 30-class problem, where $P(6, 2)$ matches the class count exactly and the ceiling is far from the achievable range.

E.3 An intrinsic-parameter advantage did not confirm

A fresh confirmatory run on KMNIST (Clanuwat et al., 2018) compared learning the firing threshold against learning synaptic weights alone. The threshold arm reached 27.078% against 25.734%, a difference of $1.344\,\mathrm{pp}$; the sample-paired interval excluded zero but the seed-paired interval did not, and the preregistered two-point threshold was not met. A control restricted to first-rank information reached 32.230% on the same data, above the top-2 arm. We did not freeze this checkpoint and we make no intrinsic-advantage claim from it.

E.4 Unsupervised factorial self-organization does not yet work

We tested whether useful regions of permutation space can be found without a label-derived code. On the unlabeled split of STL-10 (Coates et al., 2011) ($90{,}000$ train, $10{,}000$ validation, 5 seeds, $20{,}000$ Adam steps, no labels, no code and no external classifier anywhere in the training graph), we trained six neurons under a label-free factorial-information objective, against a frozen random synapse-only baseline. Semantic probes were fit on labeled training data only *after* the freeze marker, and evaluated on the untouched official test split.

The result is a clear negative. The learned arm did exceed the random baseline, the rank-decomposition effect was positive, the shuffled-label null was rejected, and the effect grew with data. But both primary gates failed. Pooled J_6 was 0.209 against a preregistered threshold of 0.25, and the non-collapse gate failed outright: 2 of 5 seeds collapsed to one and to two occupied permutations respectively. An artifact-integrity check also failed, because the frozen runner recorded an empty source-hash mapping; the model array replay audit itself passed, so this is a metadata defect rather than a result defect, and we state it for completeness.

We therefore present the geometry-aligned code of Section 6 as supervised design. Whether comparable structure can emerge without labels remains open, and our one preregistered attempt did not answer it.

E.5 A missed gate on the main result

For the main 30-class result we had preregistered a "strong geometry" outcome requiring rank-1 accuracy of at least 60%. We obtained 59.863%, short by $0.137\,\mathrm{pp}$. We did not retune and we report the gate as failed. Every other preregistered gate for that experiment passed.

F Self-Audit and Corrections

An internal audit of an earlier 30-class run found that its implementation applied a single-term first-rank loss where the written specification required the full five-term form. That run is therefore not a protocol-matched baseline, and the lexicographic row of Table 2 — which was trained under the same corrected loss as every other row — is used as the baseline instead. This appendix records the defect, the corrected specification, and the numeric differences it produced.

G CODE CONSTRUCTION

Exact construction of the three class-to-code assignments compared in Section 6: the lexicographic assignment, the ten frozen random balanced assignments with their balance constraints, and the centroid-based grouping procedure. We also ran an intended maximum-incoherence control; on the objective it was meant to maximize, it did not exceed the lexicographic assignment, so it is not a valid incoherence control and we do not report it as one.

H FIRST- AND SECOND-SPIKE OUTCOMES

Table 3 splits every test prediction of the geometry-aligned assignment by whether the first spike and the second spike each came from the target pair's neuron in that position. A prediction is correct exactly when both are right, so the top-left cell is the top-2 accuracy of Table 2. Chance for a single ordered pair is $1/30 = 3.333\%$; the pooled top-2 accuracy is $8.3\times$ that. When the first spike is right, the second is right 46.241% of the time, against 20.0% chance among the 5 remaining neurons. When the first spike is wrong, the target's second neuron still fires second 12.952% of the time.

Table 3: Outcome of the first two spikes, geometry-aligned assignment, balanced 30-class EMNIST, 6 neurons. Counts pooled over 5 seeds $\times$ $12{,}000$ official test images ($60{,}000$ predictions); percentages are of all predictions. "Right" means the spike came from the target pair's neuron in that position.

	Second spike right	Second spike wrong	Total
First spike right	**16,609 (27.682%)**	19,309 (32.182%)	35,918 (59.863%)
First spike wrong	3,119 (5.198%)	20,963 (34.938%)	24,082 (40.137%)
Total	19,728 (32.880%)	40,272 (67.120%)	60,000