Population Coding: Using Fixed Additive Lattice Group of 2×2 Matrices as Data Composite

Derrick Donkor

January 7, 2026

Abstract

We introduce a finite, structure-driven framework for neural population coding based on a fixed additive lattice of 2×2 matrices. The model defines a predetermined set of sixteen basis matrices whose admissible linear combinations generate stimulus representations in the real number space $\mathbb{R}$. Information encoding is achieved by selecting lattice combinations that maximize entropy, consistent with efficient coding principles. In this proposal, Population coding is formulated as a probabilistic inference problem governed by a Markov state-space model, where transitions occur over lattice states (distinct matrices in the lattice group), as shown in equation (1). Model parameters are inferred via maximum likelihood estimation. Since the compositional structure of the lattice is fixed *a priori*, the framework decouples population coding computational capacity from stringent network connectivity, enabling a fully computable and classical probabilistic formulation of population coding. Beyond the conventional role of population codes as output representations in [11], the proposed population-coded lattice is conceived as a structured input representation for deep learning architectures, with cross-entropy optimization serving as an objective for pattern classification. This work demonstrates a direct relation between random analog timed-series data and it's probability encoding, $P(s_i(t))$. This work unifies population coding and information theory within a finite matrix-based framework, offering a computational reinterpretation of neural representation and inference.

Keywords: Population Coding, Information Maximization, Lattice Basis, Matrix Representation, Computational Neuroscience, Markov Processes, Deep Learning, Entropy Optimization.

1 Introduction

The pursuit of computational frameworks capable of supporting biologically plausible intelligence on classical machines remains a central challenge in artificial intelligence [9]. A population code is a way of representing information through the simultaneous activity of a large set of neurons sensitive to a specific feature of the given information, and it is useful for increasing the organism's certainty about a feature as well as for encoding multiple features simultaneously [13]. While modern deep learning systems have achieved remarkable empirical success, they rely heavily on large-scale optimization and lack principled analytic structures that resemble biological information processing [16][17]. This study introduces a *composing lattice matrix group* as a compact, analytically tractable population entity whose inherent structure defines a constrained yet expressive space of attainable state transitions.

The elementary matrix unit from the lattice group is defined as

$$M_{a,b,c,d} = \begin{bmatrix} a \cdot 1 \cdot e^{-1} & b \cdot 2 \cdot e^{-1} \\ c \cdot 1 \cdot e^{-2} & d \cdot 2 \cdot e^{-2} \end{bmatrix}, \quad a, b, c, d \in \{0, 1\}. \tag{1}$$

Each binary coefficient activates or suppresses a weighted exponential basis element. The lattice induced by combinations of (a, b, c, d) as placeholders for the binary equivalent of hexadecimal digits defines a discrete field over which probabilistic inference and learning can be performed. This formulation allows the matrix to serve as a minimal population entity with explicit geometric and probabilistic interpretations.

1.1 Motivation and Background

Neural network research has increasingly focused on incorporating elements of biological realism into artificial systems [1]. Efforts range from spiking neural models to neuromorphic hardware; however, most contemporary systems still lack a principled connection between biological computation and formal analytic models [2]. In the broader context of artificial general intelligence, large-scale architectures and benchmark-driven optimization have yet to yield systems with autonomous, transferable reasoning abilities.

A growing body of evidence suggests that biological intelligence relies less on individual neuron precision and more on collective representations [3]. *Population coding* offers a compelling alternative to one-hot or sparse symbolic representations by distributing information across ensembles of units. Population codes are known to improve robustness, generalization, and noise tolerance in learning systems [11].

A population code, in this proposal, consists of a set of classifiable matrix elements whose collective activity under Bayesian statistics encodes information. While biological populations neurons often exhibit complex connectivity and interaction dynamics, such structural details are not strictly necessary for effective probabilistic inference [3]; unless the goal is direct neural decoding from biological substrates, population codes can be treated as abstract computational objects rather than anatomically faithful models.

1.2 Analytic Foundations

Mathematical concepts such as *formulaic alphas* attempt to construct predictive frameworks by combining elementary mathematical expressions, particularly in financial and time-series analysis [4]. Although this approach is conceptually challenging because considerable trial and error is needed to generate good alphas [5], it has provided conceptual intuition for constructing structured representations capable of supporting inference.

Kolmogorov's [6] axiomatic foundation of probability theory emphasizes that rigorous probabilistic reasoning requires well-defined analytic primitives, even if the physical interpretation of those primitives remains abstract. In this sense, the composing lattice matrix field proposed here serves as an admissible analytic base: the elements are well-defined mathematically, combinatorial, and closed under additive compositions, enabling probabilistic reasoning without reliance on biologically detailed mechanisms.

Heisenberg discovered that quantum physics requires matrix mathematics [14]. Khrennikov and Yamada revealed that neural systems exhibit quantum-like statistical patterns [15]. This work bridges these insights by introducing a matrix lattice framework for population coding, providing the concrete mathematical structure that implements neural matrix mechanics on classical computers.

1.3 Population Codes as Computational Primitives

Previous studies by [12] have shown that neural population properties are largely determined by the tuning and coding characteristics of individual units, often abstracting away the explicit functional interactions between neurons. This observation motivates a computational approach given in this proposal in which artificial population neurons are treated as encoding units rather than as structurally faithful replicas of biological neurons.

Given the known limitations of classical machines in emulating biological neural microstructures [9][7], this proposal adopts a computationally grounded perspective. The focus is placed on what popula-

tion codes *compute*, rather than how biological neurons *implement* those computations. The composing lattice matrix field thus functions as a population-level primitive that supports inference, learning, and decision-making.

1.4 Inference, Learning, and Open Problems

A persistent challenge in the literature concerns the use of population codes for Bayesian inference and the mechanisms by which such codes can be refined through learning. Questions remain regarding how likelihoods are represented, how priors are incorporated, and how population structures adapt to data [8, 13].

This proposal directly addresses these issues by providing a theoretical framework for inference and computation over the defined lattice matrix population. Bayesian updates are performed over matrix configurations, enabling structured uncertainty representation and cue integration.

2 MDP Components and Algorithms

A Markov Decision Process (MDP) is a mathematical framework used for modeling decision-making problems in which outcomes are partly random and partly under the control of a decision-maker, It involves a process which is characterized by a set of states, actions, transition probabilities, and rewards [19]. This paper outlines an MDP algorithm based on matrix representations and pool selection. The approach uses Value Iteration to determine the optimal policy for decision-making.

The core components of the MDP are as follows:

- **States (S)**: Represented by a set of basis matrices B. Each matrix in B describes a particular state of the system.
- **Actions (A)**: Represented by selecting from different pools. For each state, there are possible actions that correspond to transitions to other states, section 2.3.2 uses three pools.
- **Transition Model (T)**: The transition model is deterministic, and the state transitions depend on the action taken. In this case, each action selects a pool, which determines the next state.
- **Policy** π: The strategy that defines the best action to take from each state in order to maximize the expected cumulative reward.

The MDP is solved using Value Iteration to compute the optimal state values, followed by Policy Extraction to find the optimal policy.

2.1 Value Iteration Algorithm

Value Iteration is an iterative method for computing the value function, which represents the expected reward for being in each state [19]. The value function is updated based on the Bellman equation [20], which relates the value of a state to the values of its possible successor states.

Algorithm 1 Value Iteration Algorithm for MDP

Input: States $S = \{s_1, s_2, \ldots, s_n\}$, Actions $A = \{a_1, a_2, a_3\}$, Transition probabilities $P(s' \mid s, a)$, Rewards $R(s, a)$, Discount factor γ, Convergence threshold ϵ
Initialize value function $V(s) = 0$ for all $s \in S$
repeat
  $\Delta \leftarrow 0$
  for each state $s \in S$ **do**
    $v \leftarrow V(s)$
    Calculate action values:
    for each action $a \in A$ **do**
      $Q(s, a) = \sum_{s'} P(s' \mid s, a) \cdot [R(s, a, s') + \gamma V(s')]$
    end for
    $V(s) \leftarrow \max_a Q(s, a)$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
until $\Delta < \epsilon$
Output: Optimal value function V

2.2 Policy Extraction Algorithm

The policy extraction algorithm iteratively finds the best actions (policy) for an agent in a MDP by alternating between Policy Evaluation and Policy Improvement, repeating until the policy stabilizes [19]. Once the value function is obtained, we can extract the optimal policy. The policy selects the action that maximizes the expected value for each state.

Algorithm 2 Policy Extraction Algorithm

Input: Value function $V(s)$, States S, Actions A, Transition probabilities $P(s' \mid s, a)$, Rewards $R(s, a)$, Discount factor γ
Initialize policy $\pi(s) = 0$ for all $s \in S$
for each state $s \in S$ **do**
  Calculate action values:
  for each action $a \in A$ **do**
    $Q(s, a) = \sum_{s'} P(s' \mid s, a) \cdot [R(s, a, s') + \gamma V(s')]$
  end for
  $\pi(s) \leftarrow \arg\max_a Q(s, a)$
end for
Output: Optimal policy π

2.3 State and Transition Representation

In this MDP, the states are represented by matrices, and the transitions between states are determined by selecting actions from predefined pools. Each matrix in the state set B is an individual state, and the transitions are deterministic, meaning the next state is fully determined by the current state and action selected.

2.3.1 State Representation

The states in the MDP are represented by matrices $B = \{B_0, B_1, \ldots, B_{15}\}$, where each matrix corresponds to a specific configuration of the system. For example:

$$B_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad B_1 = \begin{bmatrix} 0 & 0 \\ 1 \times 10^{-2} & 0 \end{bmatrix}, \quad \cdots$$

2.3.2 Action Representation

The actions are represented by selecting from the following pools:

$$\text{pool\_0} = \left\{ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \right\}$$

$$\text{pool\_1} = \left\{ \begin{bmatrix} 0 & 0 \\ 1 \times 10^{-2} & 2 \times 10^{-2} \end{bmatrix}, \quad \cdots \right\}$$

$$\text{pool\_2} = \left\{ \begin{bmatrix} 1 \times 10^{-1} & 2 \times 10^{-1} \\ 1 \times 10^{-2} & 2 \times 10^{-2} \end{bmatrix}, \quad \cdots \right\}$$

$$\text{pool\_3} = \left\{ \begin{bmatrix} 1 \times 10^{-1} & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 2 \times 10^{-1} \\ 0 & 0 \end{bmatrix}, \quad \cdots \right\}$$

2.4 Conclusion

This section provides a framework for modeling decision-making problems using Markov Decision Processes (MDPs), where each state is represented by a matrix and actions correspond to selections from pools of matrices. The transitions between states are deterministic, and the system is optimized using Value Iteration and Policy Extraction algorithms.

3 The Encoding Probability Algorithm

The core idea behind the algorithm is to compute the cumulative probability of various paths by multiplying specific probabilistic terms, which are derived from custom encoding of the stream matrices. Given a scalar stimulus, the objective is to find the probability of the base matrices firing in response to the given stimulus.

3.1 Markovian Matrix Selection for Population Coding based on [12]

This approach provides a stochastic formulation consistent with [12], incorporating Gaussian likelihood and firing probability output.

Algorithm 3 Markovian Matrix Selection for Population Coding with Gaussian Likelihood and Firing Probability Output

Input:
  Basis matrices $\Phi = \{\Phi_1, \ldots, \Phi_{16}\}$
  Stimulus sequence $X = \{x(1), \ldots, x(T)\}$
  Initial parameters $\theta = \{\lambda_i, J_{ij}, \Sigma\}$
  Inverse temperature β, maximum iterations max_iter, learning rate η
Output:
  Population states $S = \{s(1), \ldots, s(T)\}$
  Stimulus approximations $\hat{x}(t)$
  Optimized parameters θ^*
  Firing probabilities $P(s_i(t) = 1 \mid s(t-1))$

Initialize $s(0)$ randomly: $s_i(0) \in \{0, 1\}, i = 1..16$
for iter = 1 to max_iter **do**
  for t = 1 to T **do**
    for i = 1 to 16 **do**
      Compute input field:

$$H_i(t) = \sum_k \Phi_{ik} x_k(t) + \sum_j J_{ij} s_j(t-1)$$

      Compute firing probability:

$$P(s_i(t) = 1 \mid s(t-1)) = \frac{1}{1 + \exp(-\beta H_i(t))}$$

      Sample matrix activation: $s_i(t) \sim \text{Bernoulli}(P(s_i(t) = 1))$
    end for
    Compute population response $r(t)$ from $s(t)$
    Approximate stimulus: $\hat{x}(t) = \Phi^T r(t)$
  end for
  Compute log-likelihood for the sequence:

$$\log L(\theta) = \sum_{t=1}^{T} \log \mathcal{N}(x_t; \Phi^T r(t), \Sigma) + \sum_{t=1}^{T} \sum_{i=1}^{16} \log \frac{\exp(\beta s_i(t) H_i(t))}{2 \cosh(\beta H_i(t))}$$

  Update parameters: $\theta \leftarrow \theta + \eta \nabla_\theta \log L(\theta)$
  if log-likelihood converged **then**
    break
  end if
end for
Return: $S, \hat{x}(t), \theta^*, P(s_i(t) \mid s(t-1))$

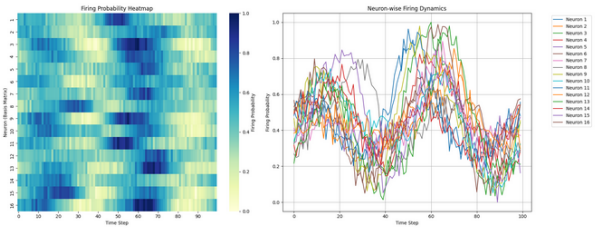

Figure 1: Left: Heatmap representation of firing probabilities across neurons (rows) and time steps (columns). Right: Temporal firing probability trajectories for individual neurons, generated from the same underlying data.

Figure 1 illustrates the relationship between population-level firing activity and individual neuron firing dynamics using same random data as the one used in Figure 2. The figure has the matrix group on the vertical axis and the random analog data on the horizontal axis. The heatmap together with the temporal firing probability on the right, provides a compact visualization of collective neural behavior, revealing structured temporal patterns such as globally synchronized oscillations and localized transient bursts that are distributed across the neuronal population. Regions of elevated intensity in the heatmap correspond to periods of increased firing probability shared by multiple units.

The corresponding firing rate plots show the same data unfolded along the temporal dimension for each neuron individually. These curves reveal neuron-specific phase shifts and amplitude variations superimposed on a common global oscillatory component. Peaks observed in individual firing rates align temporally with high-activation regions in the heatmap, demonstrating a strong qualitative correlation between single-neuron dynamics and emergent population activity.

This combined visualization highlights how structured population-level patterns arise from heterogeneous but correlated neuron-wise firing processes. Such representations are particularly useful for analyzing basis-matrix activation dynamics, neural synchrony, and probabilistic firing models, where both global coherence and local variability play critical

roles.

4 Actual Data and Its Probability Encoding

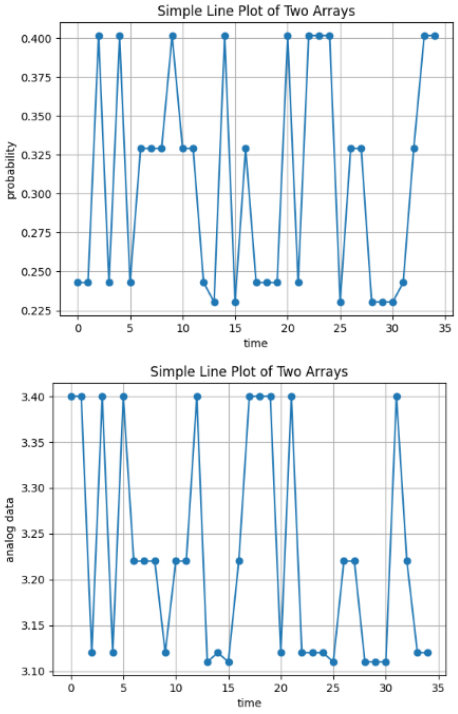

Figure 2: Left: The actual data or stimulus representation. Right: Probability encoding of the data stimulus.

Figure 2 illustrates the relationship between the original stimulus data and its probabilistic encoding of the range [0, 1) using the proposed lattice matrix framework.

4.1 Applications and Scope

The proposed population code framework has been applied to several domains, including time-series modeling, image classification, and machine noise classification. Across these applications, data of all forms can be brought to a common probability distribution, allowing a unified treatment under the same analytic framework. Cue integration, which is not in the scope of this proposal, will be considered in future analysis.

4.2 Contributions

The primary contributions of this work are as follows:

- Introduction of a composing lattice matrix field as minimal population entities.
- A theoretical framework for Bayesian inference using proposed population entities.
- Demonstration of the framework across heterogeneous data modalities.

Together, these contributions establish a foundation for biologically inspired yet computationally feasible neuronal population-based inference on classical machines.

References

[1] Halužan Vasle, A., & Moškon, M. (2024). Synthetic biological neural networks: From current implementations to future perspectives. *BioSystems*, 237, 105164. https://doi.org/10.1016/j.biosystems.2024.105164

[2] Al Abdul Wahid, S., Asad, A., & Mohammadi, F. (2024). A survey on neuromorphic architectures for running artificial intelligence algorithms. *Electronics*, 13(15), 2963. https://doi.org/10.3390/electronics13152963

[3] Panzeri, S., Moroni, M., Safaai, H., & Harvey, C. D. (2022). The structures and functions of correlations in neural population codes. *Nature Reviews Neuroscience*, 23(9), 551–567. https://doi.org/10.1038/s41583-022-00606-4

[4] Kakushadze, Z. (2015, December 9). 101 formulaic alphas. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.2701346

[5] Xu, F., Yin, Y., Zhang, X., Liu, T., Jiang, S., & Zhang, Z. (2024). Alpha2: Discovering logical formulaic alphas using deep reinforcement learning. *arXiv preprint arXiv:2405.13614*. https://doi.org/10.48550/arXiv.2405.13614

[6] Kolmogorov, A. N. (1956). *Foundations of the theory of probability* (2nd ed.). Chelsea Publishing Company.

[7] Tucker, D. M., & Luu, P. (2024). Feasibility of a personal neuromorphic emulation. *Entropy*, 26(9), 759. https://doi.org/10.3390/e26090759

[8] Pouget, A., Dayan, P., & Zemel, R. S. (2003). Inference and computation with population codes. *Annual Review of Neuroscience*, 26(1), 381–410. https://doi.org/10.1146/ annurev.neuro.26.041002.131112

[9] Apparaju, A., & Arandjelović, O. (2022). Towards new generation, biologically plausible deep neural network learning. *Sci*, 4(4), 46. https://doi.org/10.3390/sci4040046

[10] Nakai, T., & Nishimoto, S. (2023). Artificial neural network modelling of the neural population code underlying mathematical operations. *NeuroImage*, 270, 119980. https://doi.org/10.1016/j.neuroimage.2023.119980

[11] Hoffmann, H. (2024). *Advantages of neural population coding for deep learning* (Version 4). arXiv. https://doi.org/10.48550/arXiv.2411.00393

[12] Chen, K. S. (2022). Optimal population coding for dynamic input by nonequilibrium networks. *Entropy*, 24(5), 598. https://doi.org/10.3390/e24050598

[13] Ma, W. J., & Pouget, A. (2009). Population codes: Theoretic aspects. In L. R. Squire (Ed.), *Encyclopedia of neuroscience* (Vol. 7, pp. 749–755). Academic Press.

[14] Heisenberg, W. (1925). Über quantentheoretische Umdeutung kinematischer und mechanischer Beziehungen [On quantum-theoretical reinterpretation of kinematic and mechanical relations]. *Zeitschrift für Physik*, 33(12), 879–893. https://doi.org/10.1007/BF01328377

[15] Khrennikov, A., & Yamada, M. (2025). Quantum-like representation of neuronal networks' activity: Modeling "mental entanglement". arXiv preprint.

[16] Liang, B., Wang, Y., & Tong, C. (2025). AI reasoning in deep learning era: From symbolic AI to neural–symbolic AI. *Mathematics*, 13(11), 1707. https://doi.org/10.3390/math13111707

[17] Jesús García Fernández, Nasir Ahmad, and Marcel van Gerven. (2025). A unified perspective on optimization in machine learning and neuroscience: From gradient descent to neural adaptation. *arXiv preprint*. arXiv:2510.18812.

[18] Panzeri, S., Macke, J. H., Gross, J., & Kayser, C. (2015). Neural population coding: Combining insights from microscopic and mass signals. *Trends in Cognitive Sciences*, 19(3), 162–172. https://doi.org/10.1016/j.tics.2015.01.002

[19] Puterman, M. L. (1994). *Markov decision processes: Discrete stochastic dynamic programming*. Wiley series in probability and mathematical statistics. Applied probability and statistics section. New York: Wiley. ISBN 978-0-471-61977-2.

[20] Bellman, R. (1957). *Dynamic programming*. Dover Publications. ISBN 978-0-486-42809-3.