4G3 Computational Neuroscience Coursework 1

Candidate: 5641G

February 28, 2025

Declaration This report were developed with the assistance of ChatGPT for code implementation and report text organization. All the prompt information used is provided in the Appendix.

Question I: Reinforcement Learning

1. Lookup Table TD as a Special Case of Function Approximation We start from the TD update with function approximation:

$$w_k \leftarrow w_k + \epsilon \left[r_t + \gamma \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t) \right] \frac{\partial f(s_t; \mathbf{w})}{\partial w_k}. \tag{1}$$

where $f(s_t; \mathbf{w})$ is approximation of the value function $\hat{V}_\pi(s_t)$.

For a lookup table representation we choose $f(s; \mathbf{w}) = w_s$ such that each state has its own parameter. Then the estimated value for state $s_t = k$ and the gradient are:

$$\hat{V}_\pi(s_t = k) = f(s_t = k; \mathbf{w}) = w_k, \quad \frac{\partial f(s; \mathbf{w})}{\partial w_k} = \delta_{s,k} \tag{2}$$

Substituting these into the function approximation TD update (equation 1) gives:

$$w_k \leftarrow w_k + \epsilon \left[r_t + \gamma \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t) \right] \delta_{s_t,k}. \tag{3}$$

Since $\delta_{s_t,k} = 1$ only when the current state $s_t = k$ and 0 otherwise, this reduces to:

$$\hat{V}_\pi(s_t = k) \leftarrow \hat{V}_\pi(s_t = k) + \epsilon \left[r_t + \gamma \hat{V}_\pi(s_{t+1}) - \hat{V}_\pi(s_t = k) \right]. \tag{4}$$

Equation 4 is exactly the lookup table form of the TD algorithm discussed in lectures, where the parameter w_k represents the value estimate for state k.

2. Linear Function Approximation When using linear function approximation, we define the value estimate as

$$\hat{V}_\pi(s) = f(s; \mathbf{w}) = \sum_k w_k \phi_k(s), \tag{5}$$

where $\phi_k(s)$ are feature detectors. The gradient of the function with respect to w_k is

$$\frac{\partial f(s; \mathbf{w})}{\partial w_k} = \phi_k(s). \tag{6}$$

Substituting these into equation 1 yields:

$$w_k \leftarrow w_k + \epsilon \left[r_t + \gamma \sum_j w_j \phi_j(s_{t+1}) - \sum_j w_j \phi_j(s_t) \right] \phi_k(s_t). \tag{7}$$

This update rule adjusts each weight w_k according to the discrepancy between the predicted value and the TD target, modulated by the feature $\phi_k(s_t)$.

3. Tapped Delay Line Representation We implement the tapped delay line representation using the linear function approximation update from Equation **??**. In this approach, the stimulus is defined as a spike at 10 s:

$$y(t) = g(t - 10), \tag{8}$$
$$g(x) = \begin{cases} 1, & \text{if } x = 0, \\ 0, & \text{otherwise.} \end{cases}$$

The state $\mathbf{S}$ is represented as a binary 1-by-25 vector corresponding to a memory span of $T_{\text{mem}} = 12\,\text{s}$ with a time step of $\Delta t = 0.5\,\text{s}$ (plus the current step). Defining the index

$$\tau = \frac{k-1}{2}, \quad k \in \{1, \ldots, 25\},$$

the τ element of state vector is given by

$$\phi_\tau(S_\tau) = S_\tau(t) = y(t - \tau) \tag{9}$$
$$= \begin{cases} 1, & \text{if } t = \tau + 10 = \frac{k-1}{2} + 10, \\ 0, & \text{otherwise.} \end{cases}$$

The reward is smoothed around 20 s:

$$r(t) = \frac{1}{2} \exp\Big(-\frac{(t-20)^2}{2}\Big), \tag{10}$$

and the value estimate is computed as

$$\hat{V}(t) = \sum_{k=1}^{25} w_k \, S_{\frac{k-1}{2}}(t) \;=\; \mathbf{w}^\top \mathbf{S}(t). \tag{11}$$

Since only one element of $\mathbf{S}(t)$ is nonzero (specifically, when $k = 2t - 19$), the value estimate reduces to

$$\hat{V}(t) = \begin{cases} w_{2t-19}, & \text{if } 10 \le t \le 22, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

Accordingly, the TD update from Equation 7 becomes

$$w_k \leftarrow w_k + \epsilon \, \delta(t) \, S_{\frac{k-1}{2}}(t), \tag{13}$$
$$\delta(t) = r(t - \Delta t) + \gamma \, \hat{V}(t) - \hat{V}(t - \Delta t).$$

Because $S_{\frac{k-1}{2}}(t) = 1$ only when $k = 2t - 19$, only the corresponding weight is updated at time t

$$w_k \leftarrow w_k + \epsilon \left(r\Big(t = 9 + \frac{k}{2}\Big) + w_k - w_{k-1} \right), \tag{14}$$

where w_k represents the estimated value $\hat{V}(t)$ at time $t = \frac{k-1}{2} + 10$.

Figure 1 shows the stimulus and reward profiles, and Figure 2 illustrates the evolution of key variables $\hat{V}(t)$, the temporal difference

$$\Delta\hat{V}(t) = \gamma \, \hat{V}(t) - \hat{V}(t - \Delta t), \tag{15}$$

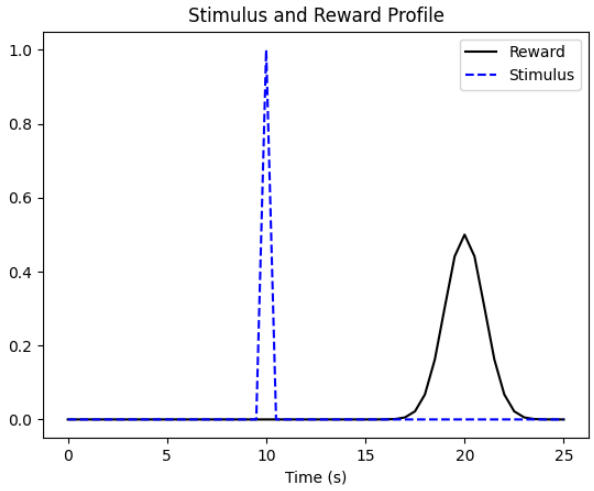


Figure 1: Stimulus and reward profiles.

and the TD error

$$\delta(t) = r(t - \Delta t) + \Delta\hat{V}(t), \tag{16}$$

recorded for every 10th trial out of $N = 201$ simulated trials.
As learning progresses, $\hat{V}(t)$ gradually evolves from mimicking the instantaneous reward to representing the expected sum of future rewards. Notably, during an intermediate stage of learning the value between the stimulus and the reward increases with time, even though it should remain constant. This behavior arises because the feature detectors $\mathbf{S}$ are delta functions (see Equation 9). In particular, at time t the temporal difference becomes

$$\begin{aligned}\Delta\hat{V}(t) &= \mathbf{w}^\top \Big[\mathbf{S}(t) - \mathbf{S}(t - 0.5)\Big] \\ &= w_{2t-19} - w_{2t-20},\end{aligned} \tag{17}$$

and the TD update depends solely on $r(t - 0.5)$ and $\Delta\hat{V}(t)$. Consequently, $\Delta\hat{V}(t)$ must be non zero between two events and the proper propagation of the value estimation from the reward time is gradual.
In the TD model, the TD error $\delta(t)$ is assumed to correspond to dopamine activity. However, the tapped delay line model fails to replicate the observation that *no dopamine activation is usually seen between the reward and stimulus times.*

4. Boxcar Representation In the boxcar representation, the feature detectors are defined as the cumulative sum over the past history:

$$\phi_\tau(\mathbf{S}(t)) = \sum_{u=0}^{\tau} S_u(t) = \begin{cases} 1, & \text{if } 10 \le t \le 10 + \tau, \\ 0, & \text{otherwise.} \end{cases}$$

The change in the feature between time $t - 0.5$ and t is

$$\phi_\tau(\mathbf{S}(t)) - \phi_\tau(\mathbf{S}(t - 0.5)) = \begin{cases} +1, & \text{if } t = 10, \\ -1, & \text{if } t = 10.5 + \tau, \\ 0, & \text{otherwise.} \end{cases}$$

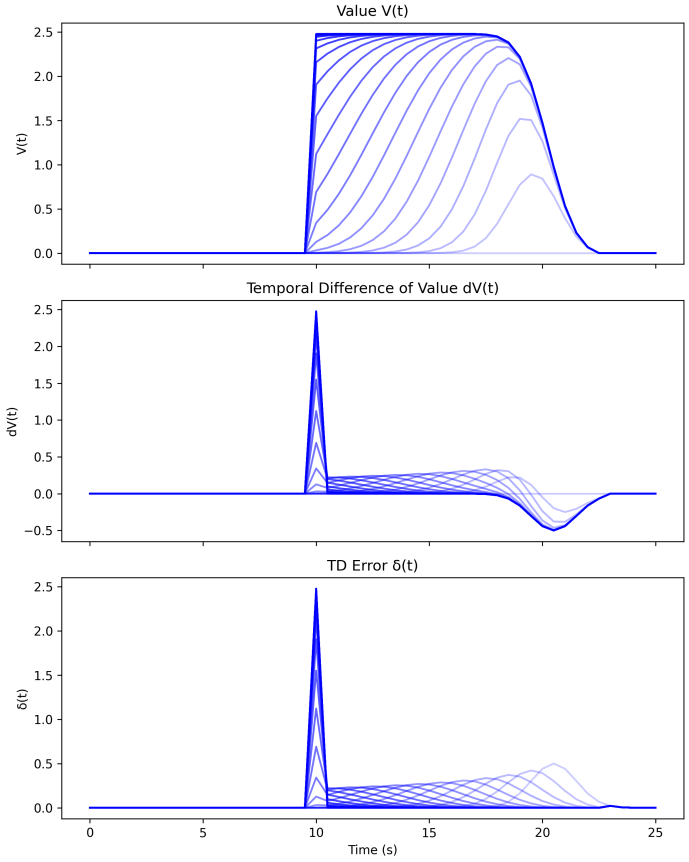


Figure 2: Key variables from the tapped delay line implementation for selected trials.

Thus, the temporal difference in the value estimate becomes

$$\Delta\hat{V}(t) = \mathbf{w}^\top \Big[\mathbf{\Phi}(t) - \mathbf{\Phi}(t-0.5)\Big]$$
$$= \begin{cases} \mathbf{w}^\top \mathbf{1}, & \text{if } t = 10, \\ -w_\tau, & \text{if } t = 10.5 + \tau, \\ 0, & \text{otherwise.} \end{cases}$$

Boxcar Representation: With the boxcar representation, the first and last trials yield identical results to the tapped delay line representation. However, in the intervening trials, the TD error $\delta(t)$ remains zero between the stimulus and reward, and the value estimate remains constant during these events (see Figure 3). the weights w are updated over multiple time steps within each trial (potentially 1 to 26 times), rather than only once as in the tapped delay line case. The cumulative nature of $\phi_\tau(\mathbf{S}(t))$ implies that the difference $\Delta\hat{V}(t)$ reflects the integrated effect of the stimulus over time. Consequently, the learned weights w come to capture the temporal profile—or "shape"—of the reward. In particular, the TD error $\delta(t)$ (and thus the weight updates) occurs primarily at the onset of the stimulus (at $t = 10$) and at the termination points of the step functions, while between these events the difference is zero, leading to $\delta(t) = 0$ in that interval. This result closely relates to the observation that *no dopamine activation is usually seen between the reward and stimulus times* [**?**].

Learning Rate Adjustment: In the tapped delay line representation, the state $\mathbf{S}$ is a delta function so that only one parameter is updated per trial—and only when w_{k-1} has been sufficiently activated. In contrast, with the boxcar representation the step-function features lead to updates across many time steps from the very first trial. Therefore, to prevent overly large updates trial and ensure learning stability, a smaller learning rate (e.g., $\epsilon = 0.01$) is required.

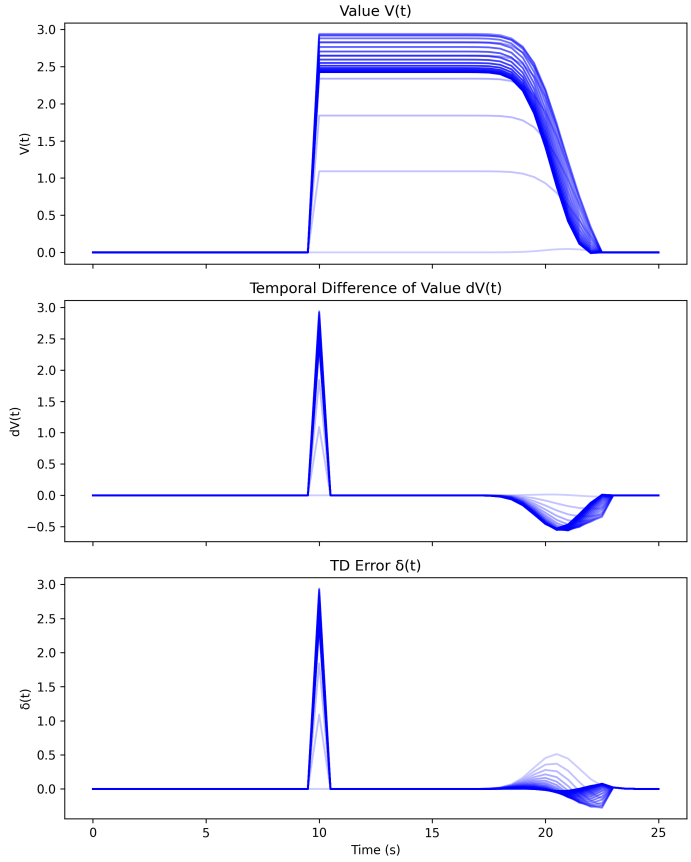


Figure 3: Value estimates, TD of the value, and TD errors for the boxcar representation.

5. Partial Reinforcement (p=0.5, N=1000 Trials) Let $R = \int_0^\infty r(t)\,dt$ denote the total reward. Then the expected value is defined as

$$\hat{V}(t) = E\left[\int_t^\infty r(t')\,dt'\right] = \begin{cases} p\Big(R - \int_0^t r(t')\,dt'\Big), & t \geq 10\,\text{s}, \\ 0, & t < 10\,\text{s}, \end{cases} \tag{18}$$

where p is the probability of receiving a reward on a trial. The temporal difference in the value is given by

$$\Delta\hat{V}(t) = p\left(-r(t)\right) + \delta_D(t-10)\,pR, \tag{19}$$

so that the TD error becomes

$$\begin{aligned} \delta(t) &= \Delta V(t) + r(t) \\ &= \begin{cases} \delta_D(t-10)\,pR + (1-p)r(t), & \text{if reward is present,} \\ \delta_D(t-10)\,pR - pr(t)+, & \text{if reward is absent.} \end{cases} \end{aligned} \tag{20}$$

Taking the expectation over trials yields

$$\begin{aligned} E[\delta(t)] &= p\Big[\delta(t-10)\,pR + (1-p)r(t)\Big] + (1-p)\Big[p\left(-r(t)\right) + \delta(t-10)\,pR\Big] \\ &= \delta(t-10)\,pR. \end{aligned} \tag{21}$$

This result implies that, at the time of reward, the TD error is essentially canceled, and only the stimulus drives the value update.

We then simulate the dopamine (DA) signal by applying a nonlinear function to the TD error. The dopamine signal is computed as

$$\mathrm{DA}(t) = \mathrm{DA}\big(\delta(t)\big), \tag{22}$$

and its average across trials is given by

$$E[\mathrm{DA}(t)] = p\,\mathrm{DA}(\text{TD error in rewarded trials}) + (1-p)\,\mathrm{DA}(\text{TD error in unrewarded trials}). \tag{23}$$

Combining these formulas, we can explain the differences between the curves plotted in Figure 4 and Figure 5:

- $\hat{V}(t)$ remains essentially identical across trial types because it represents the expectation of future rewards and depends solely on whether the stimulus is present.
- $\Delta V(t)$ exhibits similar behavior to $\hat{V}(t)$ since it is computed as the difference between successive value estimates.
- In contrast, the TD error $\delta(t)$ (see Equation 20) varies with the reward outcome: in unrewarded trials it mimics $\Delta V(t)$, while in rewarded trials it shows an upward bump due to the addition of $r(t)$. Overall, at the reward time the TD error is near zero.
- Applying the nonlinear function to obtain the DA signal, as plotted in Figure 5. Notably, the dopamine signal exhibits distinct characteristics at 10 s and 20 s. At 10 s, the TD error in each trial is strongly positive and may exceed the saturation threshold of the dopamine model; consequently, the resulting DA signal is compressed. In contrast, at 20 s, even though the average TD error is nearly flat, the asymmetric processing of positive and negative errors (rewarded trials versus unrewarded trials) produces a modest positive bump. This behavior mirrors experimental observations regarding dopamine responses in partial reinforcement paradigms.

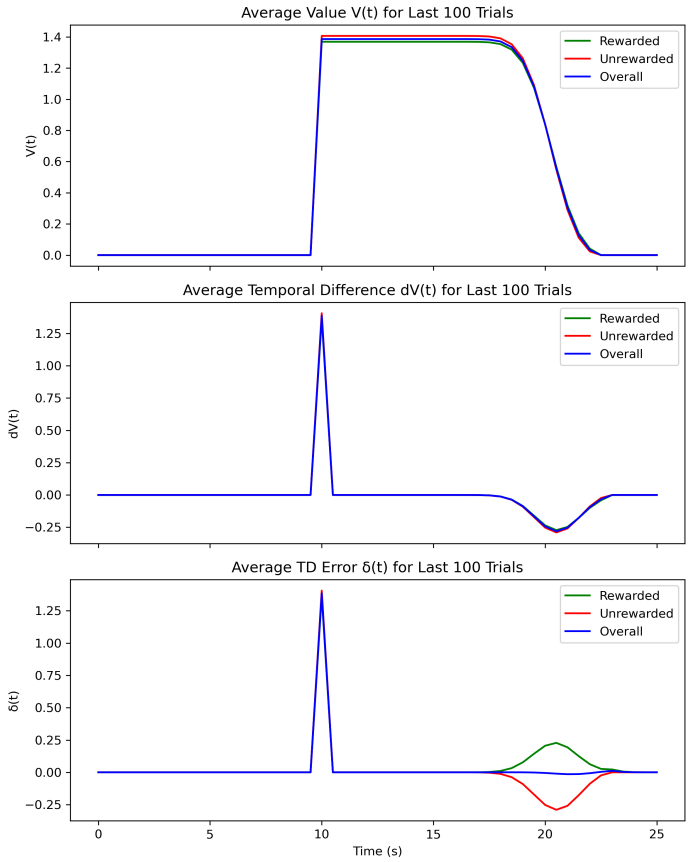


Figure 4: Averages of $\hat{V}(t)$, $\Delta\hat{V}(t)$, and $\delta(t)$ for the last 100 trials, computed separately for rewarded trials, unrewarded trials, and across all trials. Note that $\hat{V}(t)$ and $\Delta\hat{V}(t)$ are nearly identical for all cases, as they primarily depend on the presence or absence of the stimulus. In contrast, $\delta(t)$ differs markedly between rewarded and unrewarded trials due to its dependence on the reward signal.

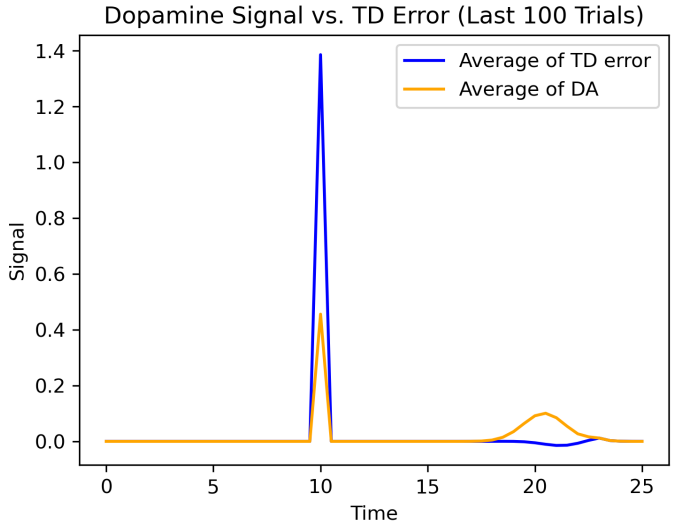


Figure 5: Simulated dopamine signal computed via the nonlinear function applied to the TD error. Note that at 10 s the original TD error, which is strongly positive, is compressed due to saturation, and at 20 s a small bump emerges due to the asymmetric response of the dopamine system.

6. Dopamine Signal for Different Reward Probabilities We simulated the average dopamine signal for different reward probabilities $p \in \{0, 0.25, 0.5, 0.75, 1\}$ over $N = 1000$ trials. Figure 6 shows that although the overall shape of the dopamine signal remains similar across these conditions, the peak dopamine at the stimulus increases with p. In contrast, at the reward time the dopamine response exhibits a bump that is maximal at $p = 0.5$; for $p = 0$ or $p = 1$ there is little to no dopamine response, and for $p = 0.25$ or $p = 0.75$ the bump is modest.

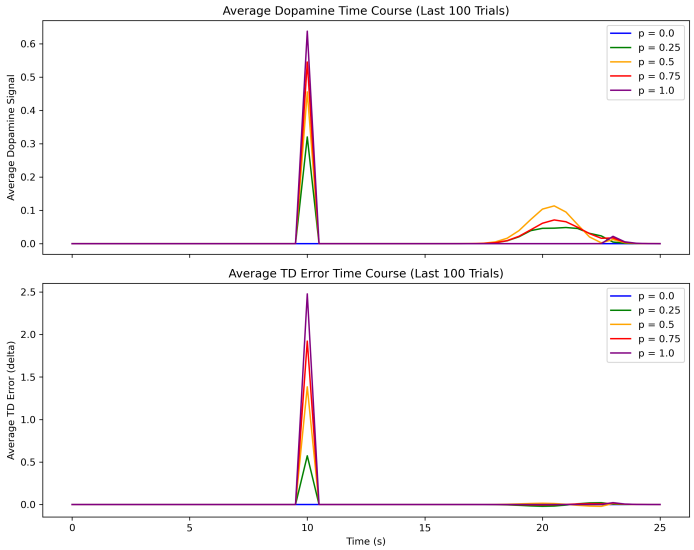


Figure 6: Average dopamine signal for various reward probabilities p. The peak dopamine at the stimulus increases with p, whereas the reward-evoked dopamine bump is maximal at $p = 0.5$.

7. Peak Dopamine at Stimulus and Reward The average dopamine signal is computed via the nonlinear function. At the stimulus, the dopamine signal is given by

$$\mathrm{DA}_{\mathrm{stim}} = \mathrm{DA}(pR), \tag{24}$$

where $R = \int_0^\infty r(t)\,dt$ is the total reward. Thus, the peak dopamine at the stimulus increases with p.
At the reward, the average dopamine response is computed as a weighted combination of the responses in rewarded and unrewarded trials:

$$\mathrm{DA}_{\mathrm{reward}} = p\,\mathrm{DA}\big((1-p)r(t)\big) + (1-p)\,\mathrm{DA}\big(-p\,r(t)\big). \tag{25}$$

Under saturation constraints, this expression reduces to

$$\mathrm{DA}_{\mathrm{reward}} = \min\{x^*, (1-p)r(t)\} \times p. \tag{26}$$

Thus, the peak dopamine at the reward first increases with p and then decreases, reaching its maximum at around $p = 0.5$. Figure 7 illustrates the relationship between p and the peak dopamine at both the stimulus and reward times.

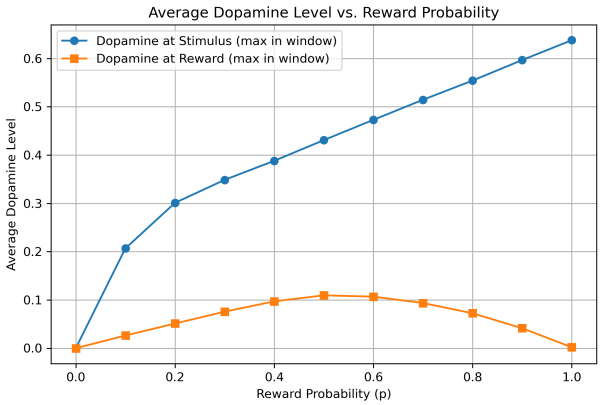


Figure 7: Peak dopamine at the stimulus and reward as a function of the reward probability p. The stimulus peak DA increases monotonically with p, whereas the reward peak DA is maximal at $p \approx 0.5$.

8. Interpretation of the Averaged Dopamine Signal The DA signal exhibits a pattern—first increasing and then decreasing with p—that appears to reflect uncertainty. In our model, this behavior is related to the binary variance $p(1-p)$ in the reward distribution, and thus it can represent uncertainty. It is important to note that for smaller values of p, the dopamine response is additionally constrained by the saturation parameter x^* and thus linear.
However, because the dopamine response to positive prediction errors (i.e., when a reward is delivered) is much stronger than the response to negative errors (when a reward is omitted), the positive signal is not completely canceled out when averaging over multiple trials. This imbalance results in an apparent ramping (gradual increase) of the DA signal over time and might look like encoding uncertainty, even though it is fundamentally driven by prediction errors.

Question II: Representational Learning

0.1. Latent Variable Extraction and Sparsity Analysis

We consider a dataset of N image patches, each of which is represented by a D-dimensional row vector $\mathbf{y}_n^\top \in \mathbb{R}^D$. We arrange these N row vectors into the matrix

$$\mathbf{Y} = \begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_N^\top \end{pmatrix} \in \mathbb{R}^{N \times D},$$

where the n-th row corresponds to the n-th image patch.
We define a feed-forward weight matrix $\mathbf{R} \in \mathbb{R}^{D \times K}$, and a generative weight matrix $\mathbf{W} \in \mathbb{R}^{D \times K}$. The dimension K is the number of latent components (or "features") we wish to extract.
The k-th latent variable $x_{n,k}$ of n-th image $\mathbf{y}_n^\top$ (the n-th row of Y) is

$$x_{n,k} = \sum_d y_{n,d}\, r_{d,k} = \mathbf{y}_n^\top \mathbf{r}_k.$$

, where $\mathbf{r}_k$ is the k-th column of feed-forward weight matrix $\mathbf{R}$.
The full latent representation of n-th image $\mathbf{y}_n^\top$

$$\mathbf{x}_n = \mathbf{y}_n^\top \mathbf{R} \in \mathbb{R}^K$$

Collecting all latent vectors $\mathbf{x}_n^\top$ into a single matrix $\mathbf{X} \in \mathbb{R}^{N\times K}$, we have

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{pmatrix} = \mathbf{Y}\,\mathbf{R} \in \mathbb{R}^{N\times K}.$$

- The n-th row of X, denoted as $\mathbf{x}_n^\top$, is the full latent representation of the n-th image $\mathbf{y}_n$. That is,

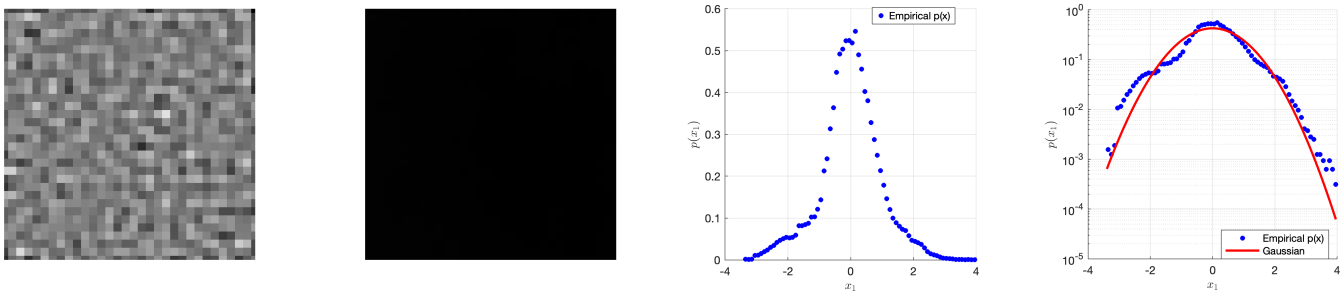
$$\mathbf{x}_n^\top = [x_{n,1},\, x_{n,2},\, \ldots,\, x_{n,K}]\,.$$

- The k-th column of X, $\{x_{n,k}\}_{n=1}^N$, contains the values of the k-th latent variable for all images. This column can be used to construct an empirical histogram to estimate the marginal distribution $p(x_k)$.

1.1 Empirical Estimation of $p(x_k)$: Figure 8–11 present the empirical histograms of the latent variable activations for selected indices $k \in \{1, 3, 30, 227\}$. For each latent variable, the histogram is constructed from the set of values

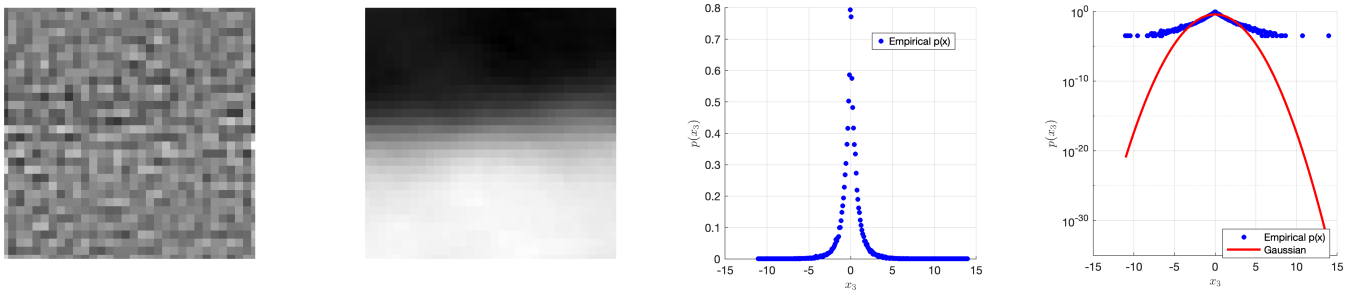
$$\{x_{n,k}\}_{n=1}^N = \{\mathbf{y}_n^\top \mathbf{r}_k\}_{n=1}^N = \mathbf{Y}\mathbf{r}_k \in \mathbb{R}^N,$$

which provides an empirical estimate of the marginal probability distribution $p(x_k)$.
In each figure, the left subfigures show the feed-forward weights $R(:,k)$ and the generative weights $W(:,k)$, while the right subfigures display the corresponding histograms of $p(x_k)$ on linear and logarithmic scales. For comparison, a Gaussian distribution is overlaid. It can be observed that for generative weights with different characteristic frequencies, **the resulting distributions are sparse**.



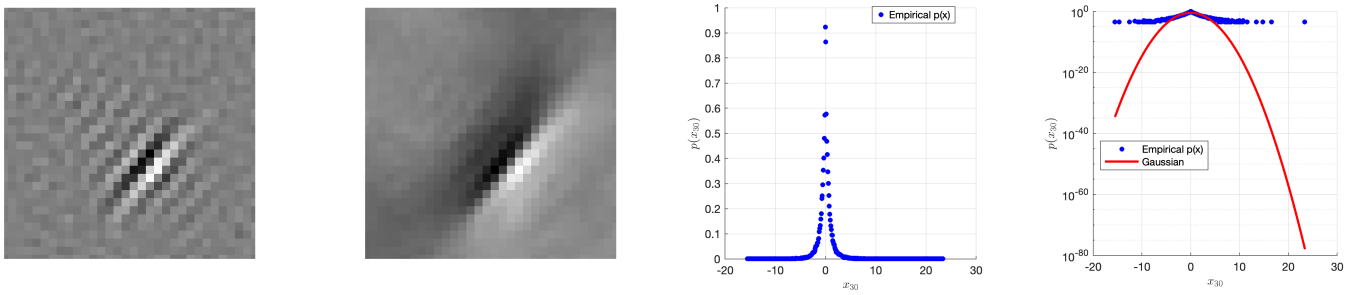
(a) $R(:,1)$ (b) $W(:,1)$ (c) Histogram (linear) (d) Histogram (log)

Figure 8: Empirical results for latent variable $k = 1$.



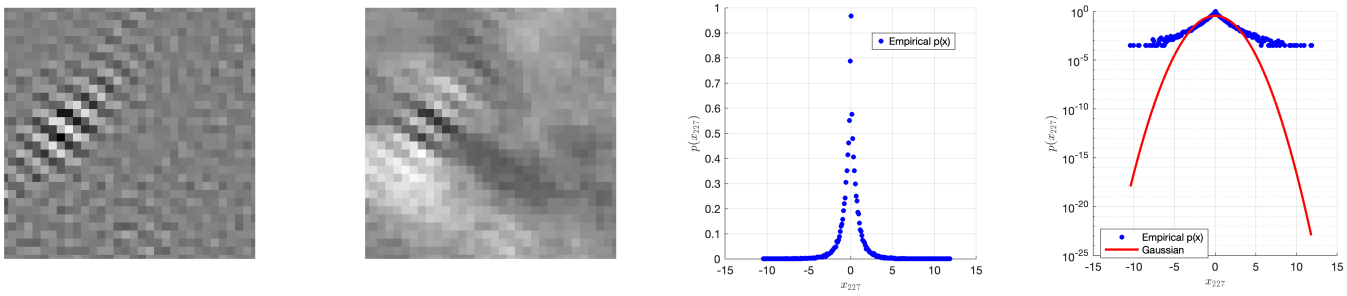
(a) $R(:,3)$ (b) $W(:,3)$ (c) Histogram (linear) (d) Histogram (log)

Figure 9: Empirical results for latent variable $k = 3$.



(a) $R(:, 30)$ (b) $W(:, 30)$ (c) Histogram (linear) (d) Histogram (log)

Figure 10: Empirical results for latent variable $k = 30$.



(a) $R(:, 227)$ (b) $W(:, 227)$ (c) Histogram (linear) (d) Histogram (log)

Figure 11: Empirical results for latent variable $k = 227$.

Empirical Investigation of Latent Component Independence In our analysis, we examined the independence between pairs of latent components. The following pairs were selected for this purpose:

- Components 1 and 51,
- Components 50 and 51,
- Components 108 and 109.

Figures 12–14 display the conditional histograms, where the horizontal axis corresponds to the variable x_{k_1} on which x_{k_2} is conditioned. If the two variables were independent, the color distribution in these figures would remain invariant with respect to x_{k_1}.

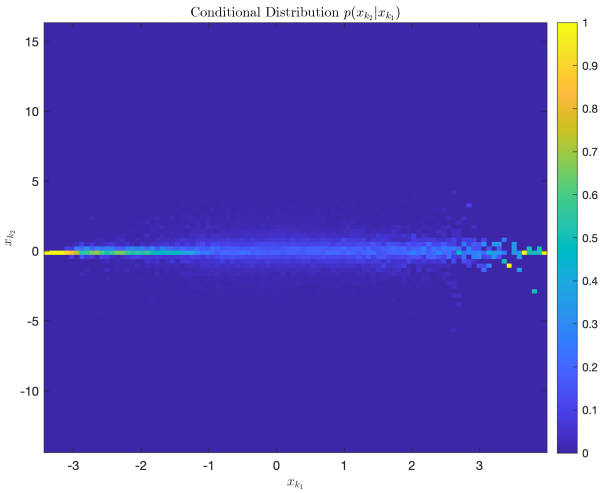


Figure 12: Conditional distribution of x_{51} given x_1.

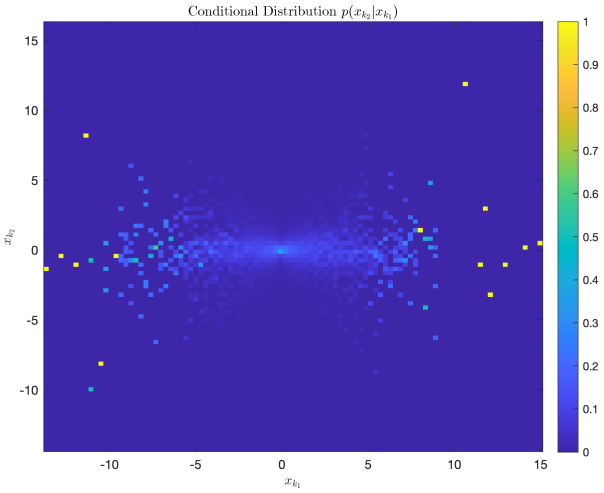


Figure 13: Conditional distribution of x_{51} given x_{50}.

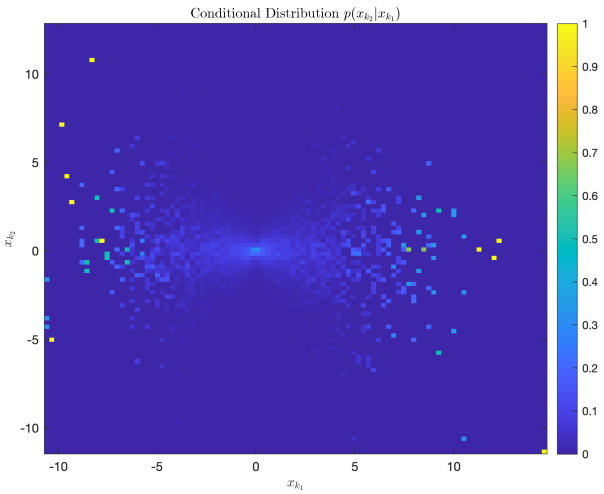


Figure 14: Conditional distribution of x_{109} given x_{108}.

Inspection of these figures reveals that as the absolute value of x_{k_1} increases, the standard deviation of x_{k_2} also increases. This indicates that x_{k_1} modulates the uncertainty of x_{k_2}, implying that the two components are not independent. Notably, the pair consisting of components 1 and 51 shows the least modulation—possibly because component 1 is nearly constant (appearing completely black) and carries very little information—suggesting a higher degree of independence relative to the other pairs.

0.2. Conditional Variance Model and Parameter Estimation

To characterize the residual dependencies among latent variables, we assume that the conditional probability is given by

$$p(x_k \mid \mathbf{x}_{\neq k}, \theta) = \frac{1}{\sqrt{2\pi\,\sigma_k^2(\mathbf{x}_{\neq k};\theta)}} \exp\!\left(-\frac{x_k^2}{2\,\sigma_k^2(\mathbf{x}_{\neq k};\theta)}\right), \tag{27}$$

with

$$\sigma_k^2(\mathbf{x}_{\neq k};\theta) = \sum_{j\neq k} a_{k,j}\, x_j^2 + b_k, \quad a_{k,j} > 0, \quad b_k > 0. \tag{28}$$

Here, $\mathbf{x}_{n,\neq k}$ denotes all components for the n-th sample except $x_{n,k}$, and θ comprises all parameters $\{a_{k,j}, b_k\}$. The conditional log-likelihood of the parameters, given a dataset $\mathbf{X}$ with N samples and K latent components, is

$$\begin{aligned}\mathcal{L}(\mathbf{X};\theta) &= \sum_{n=1}^{N}\sum_{k=1}^{K} \log p\Big(x_{n,k} \mid \mathbf{x}_{n,\neq k}, \theta\Big) \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} \left[-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log\sigma_{n,k}^2 - \frac{x_{n,k}^2}{2\,\sigma_{n,k}^2}\right],\end{aligned} \tag{29}$$

where we denote $\sigma_{n,k}^2 = \sigma_k^2(\mathbf{x}_{n,\neq k};\theta)$.

Derivatives with Respect to the Parameters First, note that

$$\frac{\partial}{\partial\sigma_{n,k}^2}\left[-\frac{1}{2}\log\sigma_{n,k}^2 - \frac{x_{n,k}^2}{2\,\sigma_{n,k}^2}\right] = -\frac{1}{2\sigma_{n,k}^2} + \frac{x_{n,k}^2}{2\,(\sigma_{n,k}^2)^2}. \tag{30}$$

For $j \neq k$, we have

$$\frac{\partial\sigma_{n,k}^2}{\partial a_{k,j}} = x_{n,j}^2. \tag{31}$$

Thus, by the chain rule,

$$\frac{\partial\mathcal{L}}{\partial a_{k,j}} = \sum_{n=1}^{N} \frac{x_{n,j}^2}{2}\left(\frac{x_{n,k}^2}{(\sigma_{n,k}^2)^2} - \frac{1}{\sigma_{n,k}^2}\right), \quad j \neq k. \tag{32}$$

Similarly, since

$$\frac{\partial\sigma_{n,k}^2}{\partial b_k} = 1, \tag{33}$$

we obtain

$$\frac{\partial\mathcal{L}}{\partial b_k} = \sum_{n=1}^{N} \frac{1}{2}\left(\frac{x_{n,k}^2}{(\sigma_{n,k}^2)^2} - \frac{1}{\sigma_{n,k}^2}\right). \tag{34}$$

It is convenient to optimize the log-parameters. By the chain rule,

$$\frac{\partial \mathcal{L}}{\partial \log a_{k,j}} = a_{k,j} \frac{\partial \mathcal{L}}{\partial a_{k,j}}, \tag{35}$$

$$\frac{\partial \mathcal{L}}{\partial \log b_k} = b_k \frac{\partial \mathcal{L}}{\partial b_k}. \tag{36}$$

Vectorized Implementation For computational efficiency, we represent the parameters in matrix form. Define the matrix

$$(\mathbf{A})_{k,j} = \begin{cases} a_{k,j}, & k \neq j, \\ 0, & k = j, \end{cases}$$

with $\mathbf{A} \in \mathbb{R}^{K \times K}$, and the vector $\mathbf{b} \in \mathbb{R}^K$ with entries b_k.
Then the conditional variance for all samples and components can be written as

$$\mathbf{\Sigma} = \mathbf{X}^{\odot 2} \mathbf{A}^\top + \mathbf{1}_N \, \mathbf{b}^\top, \tag{37}$$

where $\mathbf{X}^{\odot 2}$ denotes the element-wise square of $\mathbf{X}$ and $\mathbf{1}_N$ is an N-dimensional vector of ones.
Define the auxiliary matrix

$$\mathbf{U} = \frac{1}{2}\Big(\mathbf{X}^{\odot 2} - \mathbf{\Sigma}\Big) \odot \mathbf{\Sigma}^{\odot -2}, \tag{38}$$

where all operations are performed element-wise.
The negative log-likelihood loss is then given by

$$\text{Loss} = \frac{NK}{2} \log(2\pi) + \frac{1}{2}\Big(\mathbf{1}_N^\top \log \mathbf{\Sigma} \, \mathbf{1}_K\Big) + \frac{1}{2} \sum_{n,k} \frac{x_{n,k}^2}{\sigma_{n,k}^2}. \tag{39}$$

The gradients with respect to the log-parameters are vectorized as

$$\nabla_{\log(\mathbf{A})} \text{Loss} = -\Big(\mathbf{U}^\top \mathbf{X}^{\odot 2}\Big) \odot \mathbf{A}, \tag{40}$$

$$\nabla_{\log(\mathbf{b})} \text{Loss} = -\Big(\mathbf{U}^\top \mathbf{1}_N\Big) \odot \mathbf{b}. \tag{41}$$

Verification and Convergence The gradients were verified using the `checkgrad` function on a small subset of the data (e.g., $X_{\text{subset}} = X(1{:}10, 1{:}50)$), yielding a relative error of approximately 2.63×10^{-7}.
Subsequently, the model was trained on 80% of the dataset (25,600 images). Figure 15 shows the loss function value returned by the `minimize` function over iterations, verifying convergence of the parameter estimation.

0.3. Normalization and Comparison

After training, we define normalized latent variables as

$$c_k = \frac{x_k}{\sigma_k(x_{\setminus k}; \theta)}, \tag{42}$$

where $\sigma_k(x_{\setminus k}; \theta)$ is the conditional standard deviation estimated from the model.
Excess kurtosis is defined as

$$\text{Excess Kurtosis} = \kappa - 3, \tag{43}$$

where κ is the kurtosis of the distribution. Since a normal distribution has a kurtosis of 3, a positive excess kurtosis indicates a distribution with heavier tails and a sharper peak (leptokurtic) compared to a normal distribution, while a negative excess kurtosis indicates a distribution with lighter tails and a flatter peak (platykurtic).

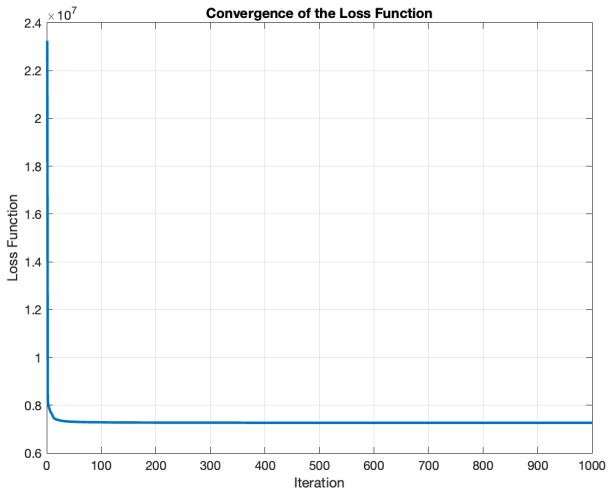


Figure 15: Convergence of the loss function during parameter estimation.

Figures 16–23 display the empirical histograms of $p(c_k)$ for latent variables $k = 1$, 3, 30, and 227 on both linear and logarithmic scales. Figure 24 shows the excess kurtosis for $p(c_k)$ is much closer to 0 than for $p(x_k)$. The results indicate that after normalization the distributions are closer to Gaussian (i.e., less sparse), demonstrating that the parameter estimation has partially corrected for residual dependencies.

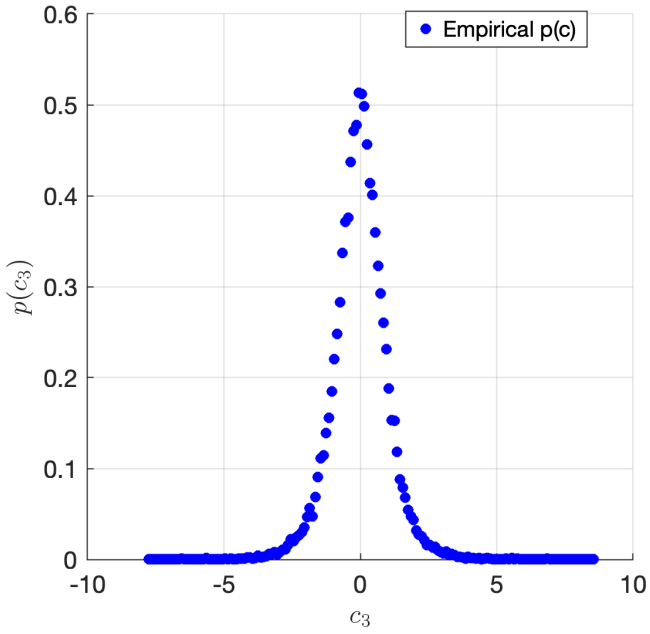


Figure 17: Normalized marginal distribution $p(c_3)$ on a linear scale.

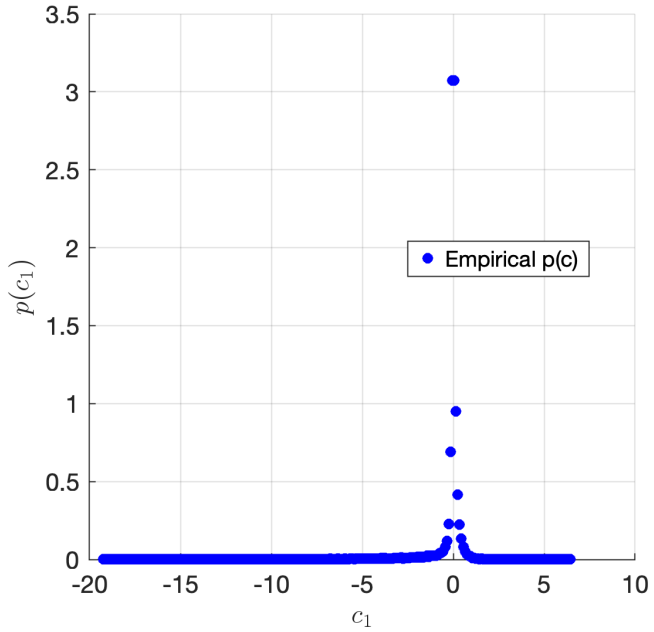


Figure 16: Normalized marginal distribution $p(c_1)$ on a linear scale.

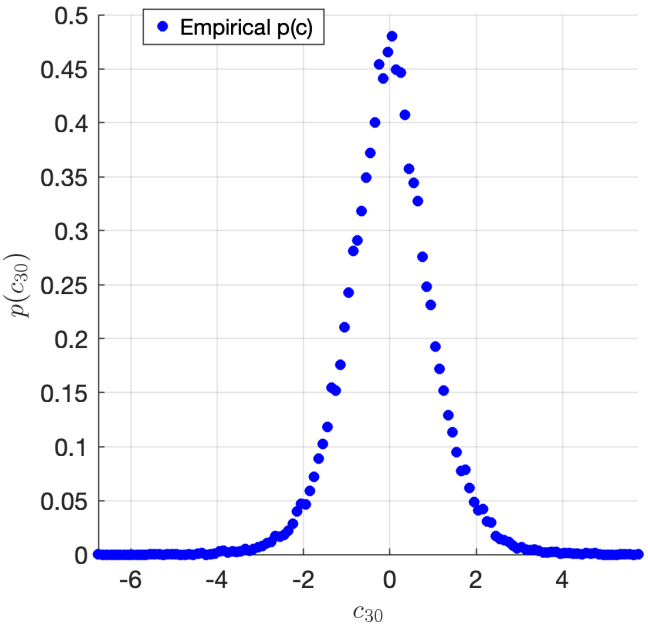


Figure 18: Normalized marginal distribution $p(c_{30})$ on a linear scale.

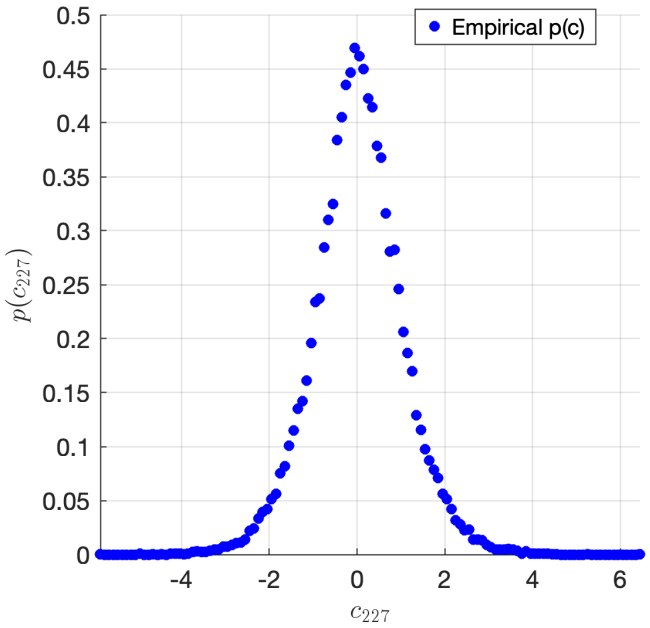


Figure 19: Normalized marginal distribution $p(c_{227})$ on a linear scale.

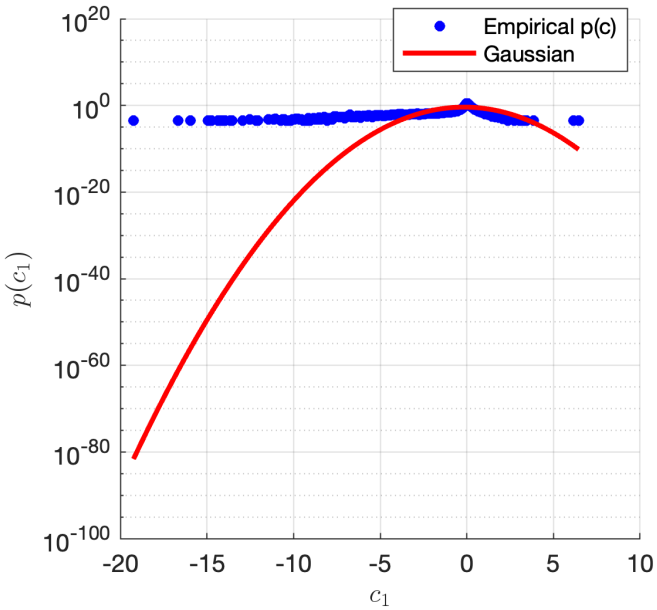


Figure 20: Normalized marginal distribution $p(c_1)$ on a logarithmic scale.

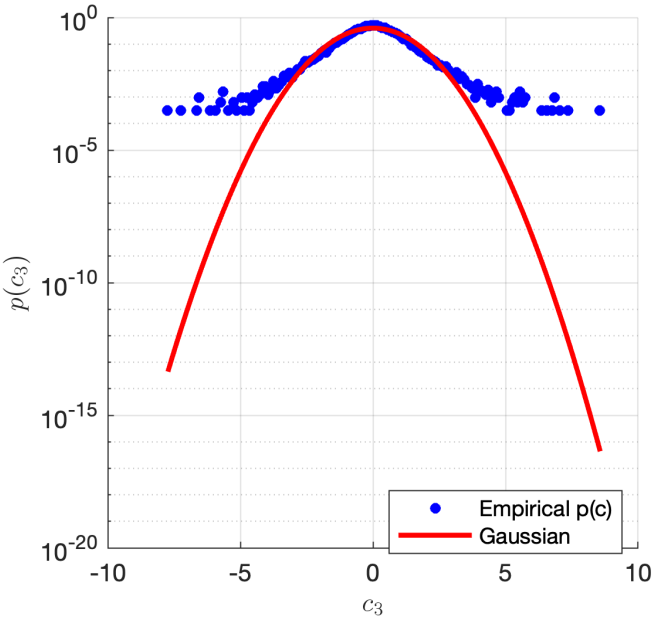


Figure 21: Normalized marginal distribution $p(c_3)$ on a logarithmic scale.

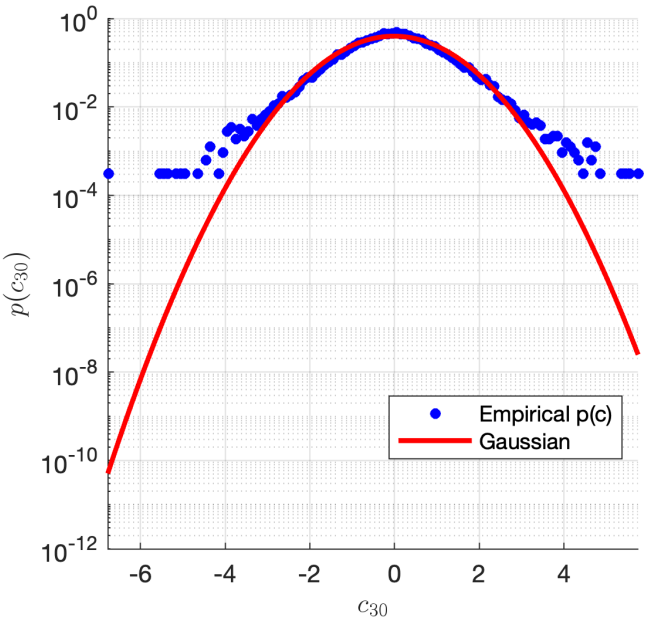


Figure 22: Normalized marginal distribution $p(c_{30})$ on a logarithmic scale.

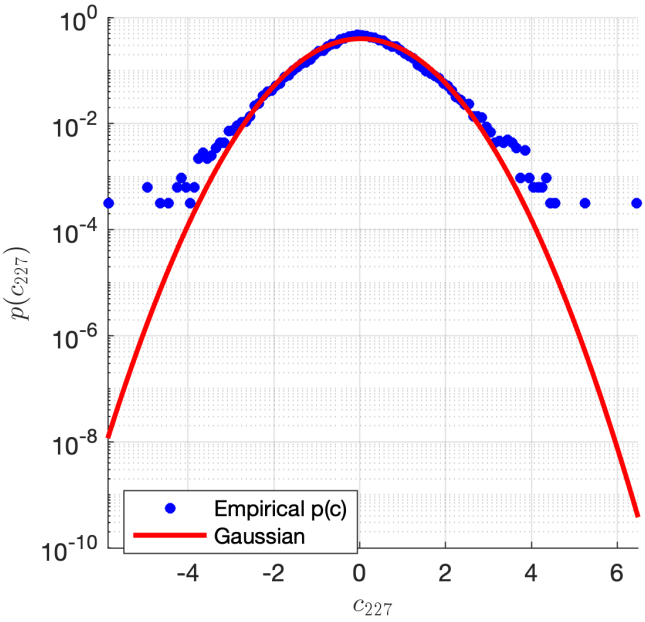


Figure 23: Normalized marginal distribution $p(c_{227})$ on a logarithmic scale.

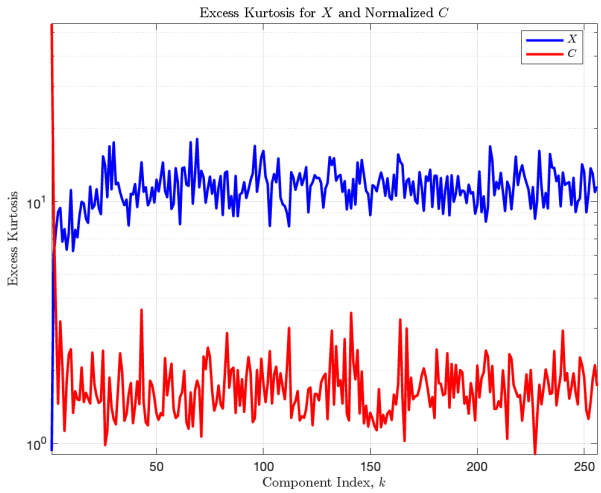


Figure 24: Comparison of excess kurtosis between $p(x_k)$ and $p(c_k)$ across latent variables.

Furthermore, we compare the conditional distributions $p(c_{k_2} \mid c_{k_1})$ with $p(x_{k_2} \mid x_{k_1})$ as shown in Figure 25 and Figure 26. It is evident that the normalized variables c_k are more independent, which is consistent with the generative model assumption $c_k \sim \mathcal{N}(0, 1)$.

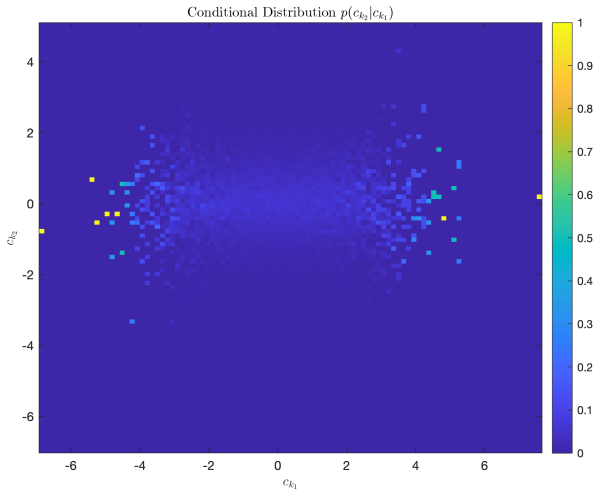


Figure 25: Conditional distribution $p(c_{51} \mid c_{50})$.

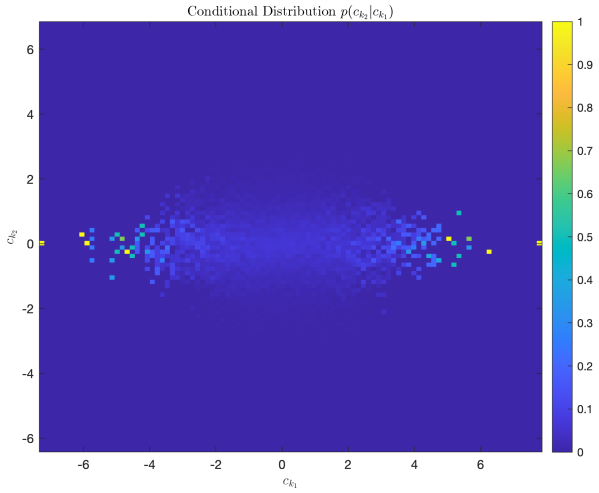


Figure 26: Conditional distribution $p(c_{109} \mid c_{108})$.

0.4. Generative Weights

The parameter $a_{k,j}$ represents the dependency of component k on component j. By plotting the generative weights using `plotIm(W(:,k))` for selected components in Figure 27 and Figure 28, we observed that those j components with larger $a_{k,j}$ values often exhibit similar orientations and spatial locations as the corresponding selected component k, albeit sometimes with the positions of the black and white edges reversed. Stronger correlations between components with similar orientations and positions indicate that they are more likely to appear together, implying that natural images exhibit local statistical dependencies: components co-occurring in the same region are coordinated in both position and orientation, and natural images tends to have continuous edges. This reflects the local translational invariance and directional consistency that is typical of natural images.

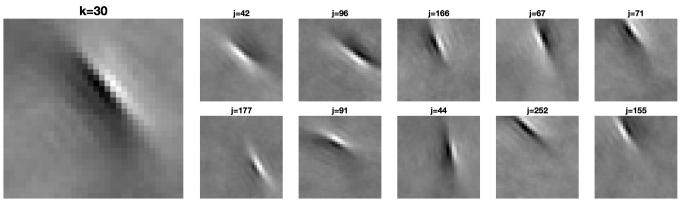


Figure 27: Combined generative weights (top 10) for component $k = 30$.

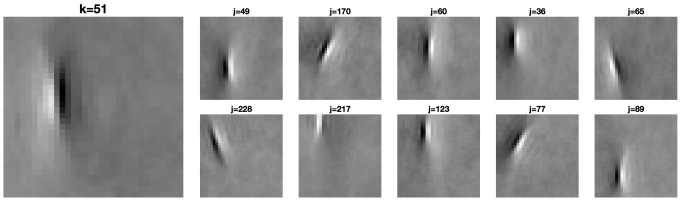


Figure 28: Combined generative weights (top 10) for component $k = 51$.

0.5. V1 Data Interpretation

In our computational framework, we identify the response of neuron k with the normalized variable

$$c_k = \frac{x_k}{\sigma_k}, \tag{44}$$

where x_k represents the signal strength of the kth component and σ_k is the normalization factor.
The computational model indicates that contrast primarily affects the magnitude of x_k. For neurons that are selective for a particular orientation and location (corresponding to component k), **the neural response is given by** c_k. When a surround masker is presented, neurons surrounding the k-selective neuron increase their activity. Although the raw signal x_k remains largely unaltered, the normalization factor σ_k increases because the response of nearby, highly correlated neurons (those with similar edge features) is enhanced. As a result, the response c_k decreases, leading to a diminished neural response for the k-selective neuron.
In contrast, for an *orthogonal mask* the coupling parameter $a_{k,j}$ is very small between k-selective neuron and neurons selective for orthogonal orientations. Thus, when an orthogonal mask is applied, even if surrounding neural responses increase due to enhanced contrast, both the normalization σ_k and raw siganl strength x_k are unchanged ,leaving the response c_k largely unchanged.
In summary, the model predicts that surround parallel masking decreases the neural response by increasing the normalization factor σ_k (while leaving x_k relatively unaltered), whereas orthogonal masking exerts minimal influence on the response because the corresponding coupling parameters $a_{k,j}$ are small. This computational interpretation is consistent with the divisive normalization observed in the neural processing of natural images.