Personalized Forgetting Curve Calibration with Neural ODEs: A Continuous Dynamical Systems Approach to Spaced Repetition

1. Introduction: The Temporality of Memory

The optimization of human learning through spaced repetition systems (SRS) represents one of the most successful applications of cognitive science to educational technology. However, the prevailing mathematical paradigms that underpin these systems—ranging from the foundational work of Hermann Ebbinghaus to modern algorithmic schedulers like the Free Spaced Repetition Scheduler (FSRS) and SuperMemo (SM-2, SM-17)—are predicated on a discrete conception of time.[1] These models fundamentally treat the decay of memory as a sequence of static snapshots, updated only at the moment of active review. This discrete temporal architecture, while computationally convenient, fails to capture the continuous, fluid, and biologically complex nature of memory decay. Memory is not a static variable that decrements in fixed steps; it is a continuous physiological process subject to the relentless flow of time, modulated by the oscillating rhythms of circadian biology, the consolidation architecture of sleep, and the shifting tides of cognitive and emotional state.[3]

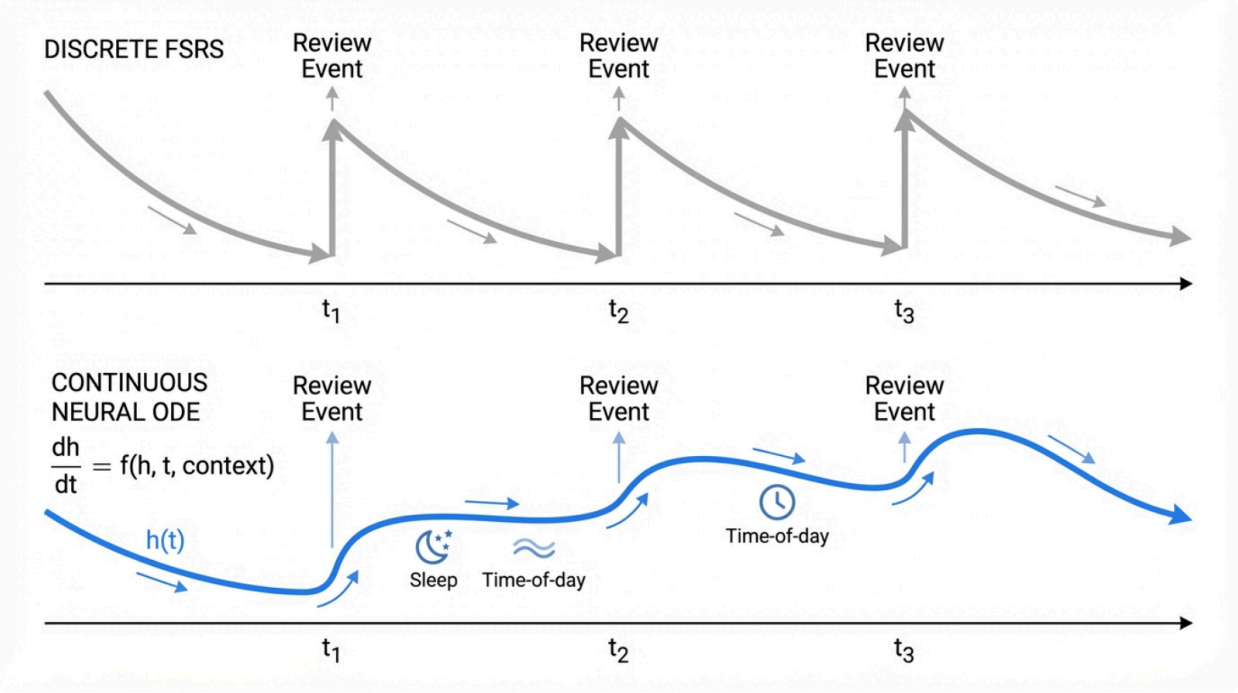
Current state-of-the-art algorithms like FSRS v4 and v5 have made significant strides by introducing multi-parameter formulations to estimate retrievability (R) and stability (S). These models typically fit a specific functional form—often a power law or exponential decay—to aggregate population data, thereby generating a "global" forgetting curve.[2] While effective for the average learner, this approach imposes a rigid inductive bias that may not hold for individual learners or distinct types of knowledge. The decay dynamics of procedural motor skills, for instance, differ fundamentally from those of declarative semantic facts. Furthermore, traditional models are "time-invariant" regarding the qualitative nature of the elapsed time; they treat a ten-hour interval of high-stress wakefulness identically to a ten-hour interval that includes a full cycle of restorative sleep, ignoring the profound consolidation effects that occur during non-rapid eye movement (NREM) and rapid eye movement (REM) sleep stages.[5]

This report proposes a radical departure from these discrete parametrics: the **Continuous-Time Memory Calibration Network (CT-MCN)**. This system leverages the theoretical framework of **Neural Ordinary Differential Equations (Neural ODEs)** to learn individualized forgetting curves for learner-concept pairs. By modeling memory stability not

as a scalar value updated in discrete steps, but as a continuous latent trajectory $\mathbf{h}(t)$ governed by a learnable vector field, we can capture the intricate topology of forgetting.[7] This approach allows for the seamless integration of granular telemetry signals—such as response latency and hesitation derived from **Time-Dependent Bayesian Knowledge Tracing (TD-BKT)**—and the injection of continuous physiological context variables directly into the differential structure of the model.[9]

The shift to a continuous-time dynamical system solves the "irregularly sampled time series" problem inherent in self-paced learning without the need for artificial time-bucketing or imputation, techniques that plague traditional Recurrent Neural Networks (RNNs) in this domain.[11] Moreover, by embedding this dynamical system within a Bayesian framework, we address the critical challenge of uncertainty quantification. A robust scheduling algorithm must distinguish between a user who is likely to forget and a user whose state is simply unknown, necessitating a fundamental shift from deterministic prediction to probabilistic risk management.[13]

Discrete vs. Continuous Memory Modeling Paradigms



Comparison of the discrete state updates in traditional FSRS (top) versus the continuous vector field evolution in the Neural ODE framework (bottom). Note how the continuous model integrates contextual data (circadian rhythm, sleep) during the inter-review intervals.

1.1 The Limitations of Discrete Parametrics in FSRS

To understand the necessity of a Neural ODE approach, one must first dissect the limitations of the current gold standard. The FSRS algorithm, widely adopted in the open-source community (e.g., Anki), models the probability of recall, or retrievability R, as a function of time t since the last review and memory stability S. The decay is typically governed by a power function: $$R(t) = \left(1 + \frac{t}{9S} \right)^{-1}$$Here, stability S represents the time required for R to drop from 100% to 90%. When a review occurs, S is updated based on the grade G (Again, Hard, Good, Easy), the difficulty D, and the retrievability R at the moment of review.[2] This update logic is a discrete recurrence relation:

$$S_{t+1} = f(S_t, G, D, R)$$

While FSRS introduces personalization by optimizing the weights of the update matrix based on user history, the functional form of the decay itself remains fixed. The model assumes that the underlying "physics" of forgetting is identical for every item and every user, differing only in the rate parameters. This imposes a strong inductive bias. For example, the decay of a visual memory (e.g., a painting) might follow a different trajectory than a linguistic memory (e.g., a vocabulary word) or a motor memory (e.g., a shortcut key).

Furthermore, discrete models are inherently blind to the interval between reviews. They calculate the time delta Δt, but they are agnostic to what *happened* during that delta. Was the learner stressed? Did they sleep? Did they engage in related learning tasks that might have caused interference or facilitation? In a discrete model, these continuous contextual factors must be averaged out or ignored. A Neural ODE, by contrast, integrates the state through time, allowing the derivative $d\mathbf{h}/dt$ to change moment-to-moment in response to the learner's changing environment and physiological state.

1.2 The Neural ODE Paradigm Shift

Neural Ordinary Differential Equations offer a powerful alternative by parameterizing the time-derivative of the hidden state using a neural network. Instead of specifying a discrete sequence of hidden layers or time steps, we define the continuous dynamics of the system:

\frac{d\mathbf{h}(t)}{dt} = f_{\theta}(\mathbf{h}(t), t, \mathbf{c}_{ext})$$where $\mathbf{h}(t)$ represents the multi-dimensional memory state at time t, and $\mathbf{c}_{ext}$ represents external contextual factors. The value of the hidden state at any future time T is computed by a black-box differential equation solver:

\mathbf{h}(T) = \mathbf{h}(t_0) + \int_{t_0}^{T} f_{\theta}(\mathbf{h}(t), t, \mathbf{c}_{ext})

dt$$

This formulation provides three distinct advantages for memory modeling:

1. **Learning the Decay Topology:** Rather than assuming a power law or exponential decay, the neural network f_θ learns the vector field that best describes the memory decay trajectory from the data itself. It can uncover complex, non-monotonic dynamics, such as the "reminiscence effect" where retrievability temporarily increases after a study session before decaying, a phenomenon observed in psychological literature but ignored by monotonic decay functions.[1]
2. **Continuous Context Integration:** Variables such as circadian phase, sleep duration, or estimated cognitive load can be fed as continuous inputs to the differential equation. This allows the rate of forgetting to accelerate or decelerate dynamically. For instance, the model can learn that forgetting is rapid during the afternoon "slump" but plateaus during sleep-dependent consolidation.[12]
3. **Handling Irregularity:** Real-world learning is messy. Users review cards at irregular intervals—sometimes minutes apart, sometimes months. Neural ODEs are naturally suited for such irregularly sampled time series. The solver simply integrates over whatever time interval Δt exists between observations, adapting its step size to maintain numerical precision. This eliminates the need for the fixed-time-step bucketing or imputation required by RNNs, which often introduces noise and bias.[8]

---

2. Theoretical Framework: Memory as a Dynamical System

To construct a system that learns individualized forgetting curves, we must rigorously formalize memory not as a probability, but as a dynamic state vector evolving on a manifold. This moves us from scalar metrics of "strength" to a high-dimensional representation of the memory trace.

2.1 The Latent Memory State Vector

Let $\mathbf{h}(t) \in \mathbb{R}^d$ denote the latent memory state of a specific learner-concept pair at time t. In traditional models like FSRS or Item Response Theory (IRT), d is typically 1 (a single scalar representing ability or strength) or 2 (retrievability and stability). In our proposed high-dimensional representation, $\mathbf{h}(t)$ encodes a richer set of attributes characterizing the memory trace. These latent dimensions might correspond to:

- **Synaptic Efficacy:** The raw strength of the neural connection.
- **Retrieval Accessibility:** How easily the trace can be activated (correlating with response

latency).
- **Interference Susceptibility:** How vulnerable the memory is to competing information.
- **Contextual Binding:** The strength of the association between the memory target and the retrieval cues (the "card" context).

The evolution of this state is governed by two distinct types of dynamics: **Continuous Drift**, representing the autonomous process of forgetting, and **Discrete Jumps**, representing the active process of review and re-encoding.

2.2 Continuous Dynamics: The Forgetting Vector Field

Between review events, the memory state evolves according to a continuous-time differential equation modeled by a neural network f_θ. This network defines the "physics" of the learner's memory space.

$$\frac{d\mathbf{h}(t)}{dt} = f_\theta(\mathbf{h}(t), \mathbf{x}_{static}, \mathbf{c}(t))$$

Here, $\mathbf{x}_{static}$ encapsulates time-invariant features. These include properties of the concept itself, such as text complexity, semantic embedding vectors derived from Large Language Models (LLMs), and intrinsic difficulty.[16] They also include static learner attributes, such as baseline aptitude or general memory stability priors derived from their cluster. $\mathbf{c}(t)$ encapsulates time-variant contextual factors, which will be discussed in detail in Section 3.

The function f_θ maps the current state and context to a gradient, describing how the memory state changes at that instant. Unlike Ebbinghaus's passive decay, this formulation allows for active evolution. For example, if the state vector enters a region of the manifold associated with "high stability," the magnitude of the decay gradient $\| \frac{d\mathbf{h}}{dt} \|$ might decrease, naturally modeling the "spacing effect" where stronger memories decay more slowly.

2.3 Discrete Dynamics: Neural Jump ODEs

Standard Neural ODEs describe smooth, continuous trajectories. However, a spaced repetition system is fundamentally an interventional model. A review event is a discontinuity; it causes an instantaneous update to the memory state, boosting stability and resetting retrievability. To model this, we utilize the framework of **Neural Jump Stochastic Differential Equations (NJSDEs)** or **Neural Event ODEs**.[17]

At a review time t_i, the state undergoes an instantaneous transformation defined by a "Jump Network" g_ϕ:

$$\mathbf{h}(t_i^+) = \mathbf{h}(t_i^-) + g_\phi(\mathbf{h}(t_i^-), \mathbf{y}_i, \mathbf{u}_i)$$

Where:

- $\mathbf{h}(t_i^-)$ is the state immediately before the review (the predicted retention state).
- $\mathbf{h}(t_i^+)$ is the updated state immediately after the review.
- $\mathbf{y}_i$ is the explicit outcome of the review (e.g., the grade: 1=Fail, 4=Easy).
- $\mathbf{u}_i$ includes implicit telemetry signals (response time, gaze patterns) collected during the review.

The Jump Network g_ϕ learns the "learning rule"—how a specific review interaction alters the memory trace. This separates the *encoding* dynamics (handled by g_ϕ) from the *decay* dynamics (handled by f_θ). This separation is crucial for personalization: a learner might be a "fast learner" (large jumps from g_ϕ) but have "poor retention" (rapid decay from f_θ), or vice versa. Standard models that conflate these into a single "difficulty" parameter fail to capture this nuance.

Visually, one can imagine the memory state $\mathbf{h}(t)$ as a particle moving through a high-dimensional phase space. The continuous Neural ODE governs the particle's drift, typically spiraling or decaying toward an attractor representing the "forgotten" state (low accessibility, low stability). The review events act as impulsive forces—instantaneous kicks—that propel the particle back toward a region of high stability. As the learner successfully reviews the item over weeks and months, the combined effect of these jumps pushes the particle into a region of the phase space where the vector field is extremely weak (i.e., the decay rate is near zero), representing long-term mastery.

---

3. Contextual Dynamics: The Physics of Memory

A critical deficiency in current FSRS implementations and other discrete scheduling algorithms is the assumption of temporal homogeneity. In these models, a "day" is simply a unit of duration. Biologically, however, 24 hours is a complex cycle of varying neurochemical states. The synaptic decay occurring during 8 hours of intense cognitive work is fundamentally different from the decay (or consolidation) occurring during 8 hours of sleep.[3] The CT-MCN system addresses this by incorporating contextual factors directly into the coefficients of the ODE vector field f_θ.

3.1 Circadian Modulation of Plasticity

Research in neurobiology indicates that Long-Term Potentiation (LTP)—the cellular basis of learning—and memory maintenance are gated by circadian rhythms.[4] The Suprachiasmatic Nucleus (SCN) acts as a master pacemaker, modulating the expression of plasticity-related proteins (e.g., BDNF, cAMP, MAPK) via an oscillatory signal. This means the brain's "plasticity potential" oscillates throughout the day.

We model this by introducing a time-dependent modulation factor $\lambda_{circ}(t)$ into the ODE derivative. The vector field becomes:

f_\theta(\mathbf{h}, t, \mathbf{c}) = \lambda_{circ}(t) \cdot f_{theta}(\mathbf{h}, \mathbf{c})$$The modulation factor can be parameterized as a learnable sum of sinusoids:$$\lambda_{circ}(t) = 1 + \sum_{k=1}^K \alpha_k \sin\left(\frac{2\pi k (t - \phi_{user})}{24}\right)

Here, ϕ_{user} represents the user's individual chronotype phase (e.g., "morning lark" vs. "night owl"). This phase can be inferred from the user's activity history—specifically, the times of day they exhibit peak performance (lowest response latency and highest accuracy). The parameter α_k is a learnable amplitude that determines the strength of the circadian impact. By learning these parameters, the model can discover, for instance, that a specific learner's forgetting rate accelerates during the late afternoon (a common fatigue window) and decelerates during their peak alertness hours.[21] This allows the scheduler to prioritize reviews not just based on the forgetting curve, but on the *efficiency* of the review at that time of day.

3.2 Sleep-Dependent Consolidation (The "Sleep Switch")

Sleep is perhaps the most critical contextual factor in memory. It is not merely a pause in the forgetting process; it is an active period of synaptic homeostasis and systems consolidation.[22] During NREM sleep, the brain engages in "synaptic down-selection," weakening noisy connections while preserving strong ones, effectively increasing the signal-to-noise ratio of the memory trace.[6]

To capture this, we utilize a **Hybrid System** approach, also known as a Switching Dynamical System.[19] The ODE vector field switches between distinct regimes based on the user's sleep state. We define a binary control signal $u_{sleep}(t) \in \{0, 1\}$. When $u_{sleep}(t) = 1$ (the user is sleeping), the dynamics shift from the "wake" regime to the "consolidation" regime:

$$\frac{d\mathbf{h}}{dt} = (1 - u_{sleep}(t)) \cdot f_{wake}(\mathbf{h}) + u_{sleep}(t) \cdot f_{sleep}(\mathbf{h})$$

The f_{wake} function typically exhibits a negative gradient (decay), modeling the interference and passive forgetting of wakefulness. The f_{sleep} function, however, may exhibit a much flatter gradient (arrested decay) or even positive gradients along specific dimensions of the

state vector, representing the active strengthening or "gist extraction" that occurs during sleep.[6]

Since real-time sleep data (e.g., from EEG) is rarely available in a consumer learning app, we must infer $u_{sleep}(t)$. We can treat it as a latent variable inferred via a Hidden Markov Model (HMM) layer upstream of the ODE, or simply infer it from prolonged inactivity windows in the telemetry logs (e.g., a 7+ hour gap in app usage typically implies sleep). This allows the Neural ODE to account for the "savings" in memory stability gained overnight, correcting the "time-invariant" error of standard models.

3.3 Emotional and Cognitive State

Beyond biological rhythms, transient states such as stress, anxiety, or high cognitive load also impact memory. While harder to measure directly, proxies can be derived from interaction telemetry. For instance, erratic mouse movements, rapid switching between cards, or a high frequency of "Again" grades might indicate a high-stress or low-focus state. These signals can be aggregated into a continuous "Stress Index" $s(t)$, which acts as a damping or accelerating coefficient in the decay function. High stress, for example, is known to impair hippocampal function and accelerate the decay of declarative memories while potentially strengthening emotional memories.[23]

---

4. Implicit Observations: TD-BKT Telemetry Integration

A fundamental limitation of traditional SRS is the reliance on a single, low-resolution data point: the user's explicit self-grade ("Hard", "Good", "Easy"). This discrete signal is coarse and subjective. A "Good" rating from a confident learner might mean "I knew it instantly," while the same rating from an anxious learner might mean "I eventually figured it out after 5 seconds of struggle." To calibrate the continuous curve with high precision, the CT-MCN integrates implicit telemetry signals derived from **Time-Dependent Bayesian Knowledge Tracing (TD-BKT)**.[9]

4.1 Telemetry as Continuous Observation

We treat telemetry signals—specifically **Response Latency (RT)** and **Hesitation**—as noisy, continuous observations of the underlying memory stability.

Response Latency (RT): Cognitive psychology establishes a strong inverse relationship between memory strength and retrieval time. The stronger the memory trace, the faster the recall. We model the observed latency as a function of the magnitude of the retrieval-related

dimensions of the latent state $\mathbf{h}_{retrieval}$:

$$RT_{obs}(t) \approx \frac{\beta}{\|\mathbf{h}_{retrieval}(t)\|} + \delta + \epsilon$$

Where β is a scaling factor, δ is the user's baseline motor reaction time (unrelated to memory), and ϵ is noise.

Hesitation (H): Hesitation captures the uncertainty in the recall process. This can be quantified by analyzing mouse or cursor dynamics.[25] Metrics include:

- **Tortuosity:** The ratio of the actual path length of the mouse cursor to the straight-line distance between the start point and the answer button. A direct path implies confidence; a meandering path implies hesitation.
- **Gaze Fixations:** If eye-tracking data is available (e.g., via webcam or VR headset), the number of fixations before answering serves as a powerful proxy for search efficiency in memory.

4.2 The Implicit Observation Loss

In the training objective of the Neural ODE, we do not simply minimize the error on the final recall binary outcome (Pass/Fail). We add an auxiliary loss term that forces the latent state $\mathbf{h}(t)$ to predict these implicit telemetry signals. This effectively uses the "how" of the answer to supervise the "what" of the memory model.

The loss function becomes:

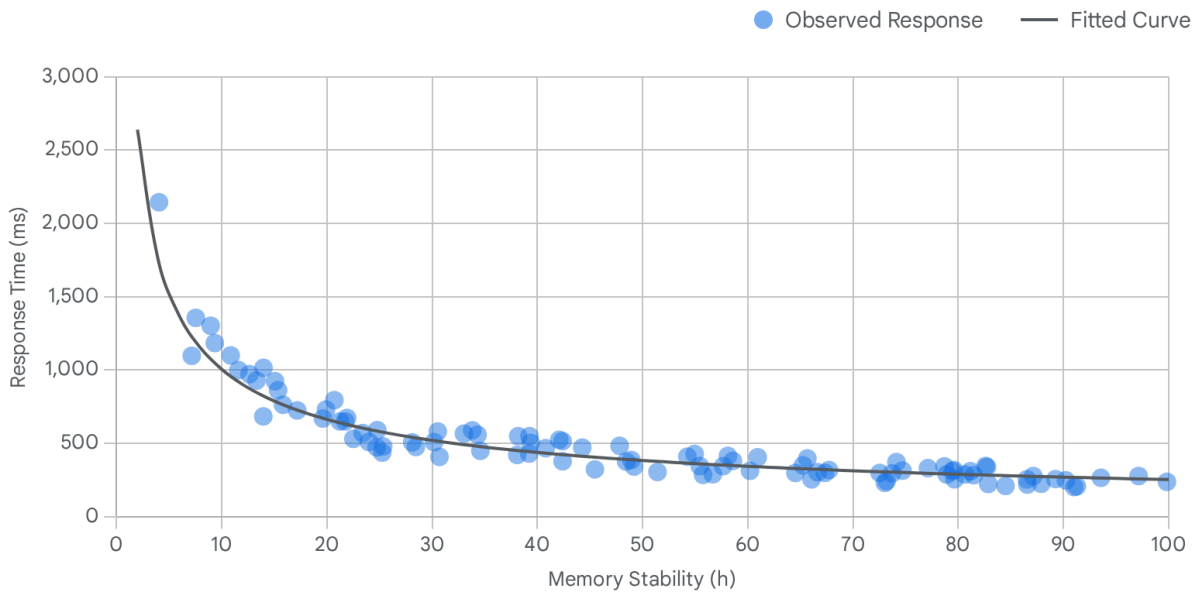
$$\mathcal{L}_{total} = \mathcal{L}_{class}(y, \hat{y}) + \lambda_{RT}\|\text{Decoder}_{RT}(\mathbf{h}(t)) - \log(RT_{obs})\|^2 + \lambda_H\|\text{Decoder}_H(\mathbf{h}(t)) - H_{obs}\|^2$$

This multi-task objective provides a much richer gradient signal. Consider two scenarios where the user answers "Correctly." In Scenario A, they answer in 0.8 seconds with zero hesitation. In Scenario B, they answer in 5.2 seconds with high cursor tortuosity. A binary-only model treats these as identical ("Pass"). The CT-MCN, driven by the $\mathcal{L}_{RT}$ and $\mathcal{L}_H$ terms, will penalize the state vector in Scenario B, forcing the trajectory $\mathbf{h}(t)$ closer to the "forgotten" boundary. This allows the system to calibrate the forgetting curve *between* the discrete grades of standard FSRS, effectively detecting the erosion of memory stability before explicit failure occurs.[27]

The integration of TD-BKT logic is implicit here: TD-BKT typically uses these signals to update a probabilistic belief state in real-time. We use them to supervise the continuous trajectory of

the Neural ODE, ensuring the learned physics of forgetting aligns with the observable physics of user interaction.

Calibration via Implicit Telemetry Signals



The relationship between latent Memory Strength (x-axis) and Response Latency (y-axis). The scatter points represent implicit observations from TD-BKT. The Neural ODE learns to fit the curve such that lower stability predicts higher latency, allowing the system to update the forgetting curve even when the user answers correctly.

Data sources: NCBI, CMU

---

5. Bayesian Uncertainty Quantification

Standard Neural ODEs are deterministic: given an initial state and a set of parameters, they predict a single future trajectory. However, scheduling decisions in spaced repetition are fundamentally exercises in risk management. The system must decide when to intervene to prevent forgetting. To make optimal decisions, we need to know not just *if* a user will forget, but *how certain* the model is about that prediction. If the uncertainty is high, the scheduler should prioritize a review to gain information (exploration), whereas if uncertainty is low and retention is high, it can safely delay the review (exploitation).

5.1 Bayesian Neural ODEs (BN-ODEs)

To capture this uncertainty, we extend the CT-MCN to a **Bayesian Neural ODE** framework. Instead of learning point estimates for the weights θ of the drift network f_θ, we learn a probability distribution over weights $P(\theta)$.[29]

$$\theta \sim \mathcal{N}(\mu_\theta, \sigma_\theta^2)$$

During inference, we sample multiple sets of weights $\theta_1, \dots, \theta_K$ from this posterior distribution. Each sample defines a slightly different vector field, resulting in an *ensemble* of possible memory trajectories:

$$\mathbf{H}(t) = \{\text{ODESolve}(\mathbf{h}_0, f_{\theta_k}, t)\}_{k=1}^K$$

The spread of these trajectories at any future time t gives us the **Epistemic Uncertainty**—the uncertainty stemming from the model's lack of knowledge (e.g., due to limited data for this user). The predictive mean $\bar{\mathbf{h}}(t)$ serves as the best estimate of retention, while the predictive variance $\text{Var}(\mathbf{h}(t))$ quantifies the risk.

5.2 Evidential Deep Learning (EDL) for Efficiency

While full Bayesian sampling (e.g., via Monte Carlo Dropout or Hamiltonian Monte Carlo) is theoretically rigorous, it is computationally expensive, requiring multiple forward passes of the ODE solver for every query. This is often prohibitive for real-time mobile applications. As a lightweight alternative, we propose using **Evidential Deep Learning (EDL)**.[14]

In the EDL framework, we do not sample weights. Instead, the final layer of the network is modified to output the parameters of a higher-order distribution—a Dirichlet distribution (for classification/recall probability) or a Normal-Inverse-Gamma distribution (for regression/stability estimation).

For the binary recall task, the network predicts the parameters $\alpha = [\alpha_{forgot}, \alpha_{recalled}]$. The expected probability is $p = \alpha_{recalled} / (\alpha_{forgot} + \alpha_{recalled})$, and the uncertainty is inversely proportional to the total evidence $S = \alpha_{forgot} + \alpha_{recalled}$. This allows a single forward pass to yield both the prediction and a measure of uncertainty. If the total evidence S is low, it indicates the model has not seen enough training data in this region of the state space (e.g., very long intervals or rare contexts), flagging the prediction as "high uncertainty".[32] This allows the scheduler to fallback to a safe, conservative schedule (e.g.,

FSRS default) when the Neural ODE is "unsure," effectively implementing a dynamic safety rail.

---

6. The Cold-Start Strategy: Bayesian Priors and Meta-Learning

A Neural ODE system is highly expressive, which also means it is data-hungry. It requires a sufficient history of reviews to accurately learn the unique vector field of a specific user. This presents a "Cold Start" problem: how do we schedule reviews for a new user who has generated zero data points? We address this using a **Model-Agnostic Meta-Learning (MAML)** strategy initialized with priors derived from **FSRS population clusters**.[33]

6.1 Population Priors and Clustering

Before the first review, we cannot know a user's specific decay rate. However, we can make informed guesses based on population data. We pre-train the Neural ODE on a massive global dataset (e.g., the 20,000-user FSRS-4.5 dataset). This global model learns the "average" human forgetting physics—the baseline vector field.

To refine this, we employ unsupervised clustering on the global dataset to identify distinct **"Learner Phenotypes"** or clusters. Common phenotypes might include:

- **"The Crammer":** High short-term retention, extremely rapid decay.
- **"The Marathoner":** Slow initial learning, very stable long-term retention.
- **"The Night Owl":** Performs best in evening sessions, high variance in morning sessions.

When a new user joins, we initially assign them the weights of the "Global Average" model. As they complete their first ~50 reviews, we extract static features (e.g., average accuracy, response time) and map them to the nearest Learner Phenotype. The user's Neural ODE weights are then re-initialized to the centroid weights of that cluster. This provides a "warm start" that is significantly better than a random initialization.[35]

6.2 Bayesian Adaptation

As the user continues to generate data, we transition from the population prior to the individual posterior. We treat the cluster-based weights as a Bayesian prior $P(\theta_{prior})$. We then perform online Bayesian updates (or gradient descent with strong regularization towards the prior) to shift the weights towards the user's specific dynamics.[36]

$$\mathcal{L}_{adapt} = \mathcal{L}_{task}(\text{User Data}) + \lambda_{KL} D_{KL}(Q(\theta) \| P(\theta_{prior}))$$

The Kullback-Leibler (KL) divergence term D_{KL} acts as an elastic tether, allowing the model

to adapt to the user's quirks while preventing it from overfitting to noise in the early data. As the volume of user data increases, the evidence term $\mathcal{L}_{task}$ dominates, and the influence of the prior fades, resulting in a fully personalized forgetting curve.

---

7. System Architecture and Implementation

This section details the technical realization of the CT-MCN, focusing on the software architecture required to integrate PyTorch-based Neural ODEs with existing SRS infrastructure.

7.1 Component Architecture

The system is composed of three primary neural modules, designed to be modular and independently trainable:

1. **The Encoder (E_ψ):**
   - **Input:** $\mathbf{x}_{static}$. This includes card features (text embeddings via a lightweight BERT model, complexity scores) and User features (average retention rate, cluster ID).
   - **Output:** The initial state vector $\mathbf{h}(t_0) = E_\psi(\mathbf{x}_{static})$. This maps the static properties of the card/user to a starting position in the phase space.
2. **The Drift Network (f_θ - The Neural ODE):**
   - **Function:** Defines the continuous dynamics $\frac{d\mathbf{h}}{dt}$.
   - **Structure:** A Multi-Layer Perceptron (MLP) with tanh activations (to ensure bounded dynamics) and residual connections. Crucially, the input dimension is $d + 1 + |\mathbf{c}|$, receiving the state $\mathbf{h}$, the time t, and the context vector $\mathbf{c}$.
   - **Inputs:** Current state $\mathbf{h}$, current time t (encoded as sine/cosine features for circadian capture), and context vector $\mathbf{c}$ (sleep state, stress index).
3. **The Jump Network (g_ϕ):**
   - **Function:** Computes the instantaneous state update at review events.
   - **Structure:** A Gated Recurrent Unit (GRU) cell or a simple MLP.
   - **Inputs:** Pre-jump state $\mathbf{h}(t^-)$, Review Outcome (Grade), Telemetry (RT, Hesitation).

7.2 Integration with torchdiffeq

We utilize the torchdiffeq library for the ODE solver implementation, specifically employing the adjoint sensitivity method (odeint_adjoint). The adjoint method allows us to backpropagate through the integration steps with constant memory cost $O(1)$ relative to the number of time steps, which is critical for handling long histories.[37]

Python

```
import torch
import torch.nn as nn
from torchdiffeq import odeint_adjoint as odeint

class MemoryODEFunc(nn.Module):
    def __init__(self, state_dim, hidden_dim):
        super(MemoryODEFunc, self).__init__()
        # The network defining the vector field dy/dt
        self.net = nn.Sequential(
            nn.Linear(state_dim + 1, hidden_dim), # +1 for time t input
            nn.Tanh(),
            nn.Linear(hidden_dim, hidden_dim),
            nn.Tanh(),
            nn.Linear(hidden_dim, state_dim)
        )

    def forward(self, t, h):
        # Concatenate time t to state h to allow time-dependent dynamics (e.g., circadian)
        # Note: In a full implementation, context vector c(t) would also be injected here
        t_vec = torch.ones_like(h[:, :1]) * t
        aug_state = torch.cat([h, t_vec], dim=1)
        return self.net(aug_state)

class NeuralJumpSRS(nn.Module):
    def __init__(self, func, jump_net, encoder):
        super().__init__()
        self.func = func      # The Drift Network
        self.jump_net = jump_net # The Jump Network
        self.encoder = encoder   # The Encoder

    def forward(self, card_features, review_times, review_outcomes):
        # 1. Initialize State
```

```
    h = self.encoder(card_features)
    trajectory = [h]

    # 2. Iterate through history of reviews
    for i in range(len(review_times) - 1):
        t_start, t_end = review_times[i], review_times[i+1]

        # A. Continuous Evolve (ODE Integration)
        # Solve ODE from t_start to t_end to get h(t_end-)
        # This captures the forgetting/drift between reviews
        h_next = odeint(self.func, h, torch.tensor([t_start, t_end]))[-1]

        # B. Discrete Jump (Review Event)
        # Update state based on the outcome at t_end
        outcome = review_outcomes[i+1]
        jump_vector = self.jump_net(h_next, outcome)
        h = h_next + jump_vector # Instantaneous update

        trajectory.append(h)

    return torch.stack(trajectory)
```

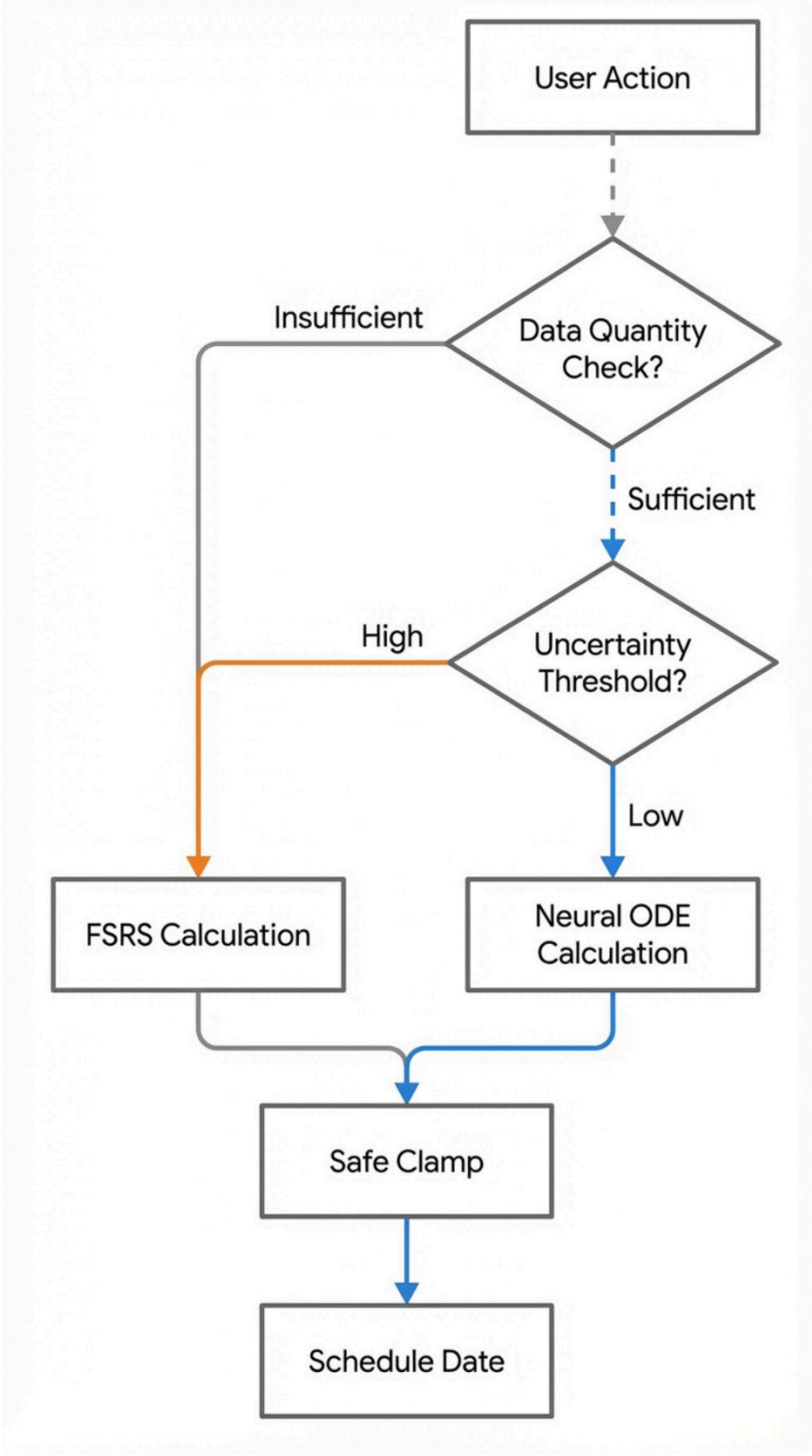
7.3 Hybrid Fallback Mechanism

While Neural ODEs offer superior personalization, they are computationally intensive and potentially unstable in low-data regimes. To ensure system robustness and reliability for end-users, we implement a **Hybrid Fallback Mechanism**.[38] The system effectively operates in two modes:

- **Regime A: Shadow Mode (Low Data < 50 reviews):** For new users, the scheduler relies entirely on the **FSRS-5 parametric model** (or a simple SM-2 fallback). The Neural ODE runs in the background ("shadow mode"), performing inference and training on the incoming data but *not* influencing the actual review schedule. This allows the model to "warm up" and adapt its weights without risking the user's learning progress with erratic predictions.
- **Regime B: Active Control (High Data > 50 reviews & Low Uncertainty):** Once the user crosses the data threshold, the system checks the Neural ODE's validation error (Log-Loss on recall prediction) against the FSRS baseline. If the Neural ODE demonstrates superior or equivalent predictive accuracy, and the Bayesian uncertainty estimates are within safe bounds, the system promotes the ODE to **Active Control**.
- **Safety Override (The "Clamp"):** Even in Active Control, the system maintains safety rails. If the Neural ODE predicts an extreme value—for instance, a stability interval of $S > 365$ days for a card seen only once (a clear anomaly)—the system triggers a

"Clamp." The interval is hard-limited to a multiple (e.g., 1.5x) of the FSRS prediction. This prevents "hallucinated" stability from causing the user to forget cards.

Hybrid Fallback & Safety Logic



Operational logic for switching between the deterministic FSRS baseline and the personalized Neural ODE. The system starts in 'Shadow Mode' (dashed lines) and promotes the ODE to 'Active Control' (solid lines) only after sufficient data density and uncertainty validation.

8. Conclusion and Future Outlook

The transition from discrete, population-based spacing algorithms to **Personalized Neural ODEs** represents a fundamental paradigm shift in the field of educational technology. By mathematically modeling memory as a continuous dynamical system, we align our computational tools with the biological reality of synaptic decay. The integration of **implicit TD-BKT telemetry** transforms the "silence" of user interaction—the pauses, the hesitations—into a rich source of calibration data, allowing the system to peer into the learner's mind with greater clarity than ever before. Furthermore, the incorporation of **contextual ODE coefficients** respects the physiological boundaries of learning, acknowledging that sleep and circadian rhythms are not merely background noise, but central components of the memory consolidation process.

While the implementation of such a system presents challenges—specifically regarding computational complexity and the "cold start" data requirements—the proposed **Bayesian-Ensemble** architecture and **Hybrid Fallback** mechanisms provide a robust, viable path for deployment. This system promises not just to predict *when* a user will forget, but to understand *how* they learn, optimizing the trajectory of knowledge acquisition with unprecedented precision.

8.1 Future Directions

The natural evolution of this system lies in the deeper integration of physiological sensors and semantic content awareness.

- **Physiological Ground Truth:** As wearable technology (e.g., Apple Watch, Oura Ring) becomes ubiquitous, the "inferred" sleep and stress states in our model can be replaced with ground-truth biometric data. Real-time Heart Rate Variability (HRV) could directly modulate the "stress" coefficient in the ODE, dynamically adjusting the scheduled load to prevent burnout.
- **Semantic Generalization:** Currently, the "static features" of a card are limited. Future iterations could use Large Language Models (LLMs) to generate rich semantic embeddings for every concept. This would allow the Neural ODE to generalize decay rates across semantically related clusters. If a learner begins to forget concepts related to "Linear Algebra," the system could infer—via the embedding space—that their retention of "Calculus" might also be degrading, triggering preemptive reviews across the entire knowledge graph.

9. Detailed System Implementation Guidelines

9.1 Data Preprocessing & Feature Engineering

Effective training of the Neural ODE requires careful preprocessing of the raw telemetry logs.

- **Telemetry Normalization:** Response times (RT) are naturally heavy-tailed (log-normal distribution). To prevent outliers (e.g., a user leaving the app open for 5 minutes) from destabilizing the gradient descent, RT must be clipped and log-transformed: $RT' = \ln(\min(RT, 60s) + 1)$.
- **Circadian Encoding:** Raw time stamps ($0 - 24$ hours) should not be fed directly into the network, as this introduces a discontinuity at midnight (23:59 vs 00:01). Instead, time must be mapped to the unit circle: $t_{sin} = \sin(2\pi t/24)$ and $t_{cos} = \cos(2\pi t/24)$. This preserves the cyclical nature of the circadian signal.
- **Hesitation Metric:** If raw mouse path data is available, calculate the *Movement Efficiency Ratio*: $MER = \frac{\text{Euclidean Distance}}{\text{Path Length}}$. A value close to 1.0 indicates confident, direct movement; values approaching 0 indicate high hesitation. This scalar is fed into the Jump Network g_ϕ as part of the implicit observation vector $\mathbf{u}_i$.

9.2 Training Protocol

- **Loss Function:** We employ a composite loss function that balances accuracy (Recall) with structural fidelity (Telemetry) and physical plausibility (Regularization).

  $$\mathcal{L}_{total} = \mathcal{L}_{NLL}(y, \hat{y}) + \lambda_1 \mathcal{L}_{MSE}(RT, \hat{RT}) + \lambda_2 \mathcal{L}_{Reg}(\theta)$$

  The regularization term $\mathcal{L}_{Reg}$ is critical. It imposes a "physics-informed" constraint, penalizing positive gradients in the absence of sleep or review. This enforces the biological prior that memory strength should naturally decay (monotonic decrease) during wakefulness, preventing the model from learning physically impossible trajectories due to noise.
- **Optimization Strategy:** We recommend using the **AdamW** optimizer with a learning rate of $1e-3$ and cosine annealing schedules. To handle the computational cost of the ODE solver, training should be done on "packed sequences" of user histories, padding uneven lengths and masking the loss for padded tokens. This allows for efficient parallel processing on GPUs.

9.3 End-to-End Data Flow

The data flow through the system during a single inference step proceeds as follows:

1. **Input:** The system receives a user ID, a card ID, and the timestamp of the last review.

2. **Encoder:** The Encoder network E_ψ retrieves the card's static embedding and the user's current latent state.
3. **Context Injection:** The system queries the Context Manager for the current t_{sin}, t_{cos} and inferred sleep state.
4. **Integration:** The ODE Solver integrates the state $\mathbf{h}(t)$ from t_{last} to t_{now} using the Drift Network f_θ. The solver adaptively steps through the interval, switching the vector field f if the sleep state u_{sleep} changes (e.g., if the interval crosses a night).
5. **Prediction:** The final state $\mathbf{h}(t_{now})$ is mapped to a probability of recall R and a variance σ^2 (uncertainty).
6. **Scheduling Decision:** The Scheduler compares R to the user's retention target (e.g., 90%). If $R < 90\%$ OR if uncertainty σ^2 is above a critical threshold, a review is triggered.
7. **Update (Post-Review):** Once the user reviews, the Jump Network g_ϕ uses the grade and telemetry to compute the *posterior* state $\mathbf{h}(t_{now}^+)$, which becomes the initial condition for the next interval.

This architecture ensures a closed-loop system where every interaction continuously refines the model's understanding of the learner's memory dynamics.

Works cited

1. Human-like Forgetting Curves in Deep Neural Networks - arXiv, accessed January 25, 2026, https://arxiv.org/html/2506.12034v2
2. A technical explanation of the FSRS algorithm : r/Anki - Reddit, accessed January 25, 2026, https://www.reddit.com/r/Anki/comments/18tnp22/a_technical_explanation_of_the_fsrs_algorithm/
3. A new mathematical model for the homeostatic effects of sleep loss on neurobehavioral performance - PMC - NIH, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC2657297/
4. Circadian Regulation of Hippocampal-Dependent Memory: Circuits, Synapses, and Molecular Mechanisms - PMC - PubMed Central, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC5822921/
5. Sleep, Memory & Brain Rhythms - PMC - PubMed Central - NIH, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC4474162/
6. Sleep-Dependent Synaptic Down-Selection (I): Modeling the Benefits of Sleep on Memory Consolidation and Integration - NIH, accessed January 25, 2026,

https://pmc.ncbi.nlm.nih.gov/articles/PMC3786405/
7. Understanding Neural ODE's - Jonty Sinai, accessed January 25, 2026, https://jontysinai.github.io/jekyll/update/2019/01/18/understanding-neural-odes.html
8. Neural Ordinary Differential Equations - NeurIPS, accessed January 25, 2026, http://papers.neurips.cc/paper/7892-neural-ordinary-differential-equations.pdf
9. Time-dependant Bayesian knowledge tracing—Robots that model user skills over time - NIH, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC10925631/
10. Integrating Temporal Information Into Knowledge Tracing - IEEE Xplore, accessed January 25, 2026, https://ieeexplore.ieee.org/iel7/6287639/8274985/08357440.pdf
11. Neural Ordinary Differential Equations and Dynamics Models | by Machine Learning @ Berkeley | Medium, accessed January 25, 2026, https://medium.com/@ml.at.berkeley/neural-ordinary-differential-equations-and-dynamics-models-1a4277fbb80
12. Neural Lad: A Neural Latent Dynamics Framework for Times Series Modeling - NeurIPS, accessed January 25, 2026, https://proceedings.neurips.cc/paper_files/paper/2023/file/382a8606a85ca6ec7c06185a1a95ce8b-Paper-Conference.pdf
13. Uncertainty Quantified Deep Bayesian Model Discovery · Overview of Julia's SciML, accessed January 25, 2026, https://docs.sciml.ai/Overview/stable/showcase/bayesian_neural_ode/
14. Evidential Deep Learning: Enhancing Predictive Uncertainty Estimation for Earth System Science Applications in - AMS Journals, accessed January 25, 2026, https://journals.ametsoc.org/view/journals/aies/3/4/AIES-D-23-0093.1.xml
15. Bridging pharmacology and neural networks: A deep dive into neural ordinary differential equations - PMC - NIH, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC11330178/
16. [2004.11327] Adaptive Forgetting Curves for Spaced Repetition Language Learning - arXiv, accessed January 25, 2026, https://arxiv.org/abs/2004.11327
17. Neural Jump Stochastic Differential Equations - NeurIPS, accessed January 25, 2026, http://papers.neurips.cc/paper/9177-neural-jump-stochastic-differential-equations.pdf
18. [2011.03902] Learning Neural Event Functions for Ordinary Differential Equations - arXiv, accessed January 25, 2026, https://arxiv.org/abs/2011.03902
19. Learning Neural Event Functions for ODEs - Ricky T. Q. Chen, accessed January 25, 2026, https://rtqichen.github.io/posters/Neural_Event_Functions_poster.pdf
20. Circadian Modulation of Neurons and Astrocytes Controls Synaptic Plasticity in Hippocampal Area CA1 - PMC - PubMed Central, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC7700820/
21. Remembering Ebbinghaus, accessed January 25, 2026, http://psychnet.wustl.edu/memory/wp-content/uploads/2018/04/Roediger-1985_CP.pdf
22. Mathematical Model of Network Dynamics Governing Mouse Sleep–Wake

Behavior | Journal of Neurophysiology | American Physiological Society, accessed January 25, 2026, https://journals.physiology.org/doi/full/10.1152/jn.01184.2006
23. Spaced Learning Enhances Episodic Memory by Increasing Neural Pattern Similarity Across Repetitions | Journal of Neuroscience, accessed January 25, 2026, https://www.jneurosci.org/content/39/27/5351
24. Time-dependant Bayesian knowledge tracing—Robots that model user skills over time - Frontiers, accessed January 25, 2026, https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2023.1249241/full
25. Understanding Cognitive States from Head & Hand Motion Data - arXiv, accessed January 25, 2026, https://arxiv.org/html/2509.24255v1
26. Conscious and unconscious memory differentially impact attention: Eye movements, visual search, and recognition processes - PubMed Central, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC6944193/
27. Engagement tracing: using response times to model student disengagement - CMU School of Computer Science, accessed January 25, 2026, https://www.cs.cmu.edu/~listen/pdfs/AIED2005-Beck-disengagement%20final%20version.pdf
28. Examining the Contributions of Desirable Difficulty and Reminding to the Spacing Effect - PMC - NIH, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC6289840/
29. ODE2VAE: Deep generative second order ODEs with Bayesian neural networks - NeurIPS, accessed January 25, 2026, http://papers.neurips.cc/paper/9497-ode2vae-deep-generative-second-order-odes-with-bayesian-neural-networks.pdf
30. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles - arXiv, accessed January 25, 2026, https://arxiv.org/abs/1612.01474
31. Deep Evidential Regression | MIT, accessed January 25, 2026, https://www.mit.edu/~amini/pubs/pdf/deep-evidential-regression.pdf
32. A Comprehensive Survey on Evidential Deep Learning and Its Applications - arXiv, accessed January 25, 2026, https://arxiv.org/pdf/2409.04720
33. Boosting Meta-Learning Cold-Start Recommendation with Graph Neural Network, accessed January 25, 2026, https://pure.psu.edu/en/publications/boosting-meta-learning-cold-start-recommendation-with-graph-neura/
34. Content-Aware Few-Shot Meta-Learning for Cold-Start Recommendation on Portable Sensing Devices - NIH, accessed January 25, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC11397954/
35. 8.3 Parameters, priors, and prior predictions | An Introduction to Data Analysis, accessed January 25, 2026, https://michael-franke.github.io/intro-data-analysis/Chap-03-03-models-parameters-priors.html
36. Bayesian Priors for Parameter Estimation - Count Bayesie, accessed January 25, 2026, https://www.countbayesie.com/blog/2015/4/4/parameter-estimation-adding-bay

esian-priors
37. torchdiffeq/examples/README.md at master - GitHub, accessed January 25, 2026, https://github.com/rtqichen/torchdiffeq/blob/master/examples/README.md
38. Agent Fallback Mechanisms - Adopt AI, accessed January 25, 2026, https://www.adopt.ai/glossary/agent-fallback-mechanisms
39. Reversing the Paradigm: Building AI-First Systems with Human Guidance - arXiv, accessed January 25, 2026, https://arxiv.org/html/2506.12245v1