Architecting Next-Generation Adaptive Learning Ecosystems: A Technical Synthesis of Deep Knowledge Tracing, Contextual Bandits, and Privacy-Preserving Microservices

Executive Summary

The convergence of advanced machine learning architectures, real-time distributed systems, and cognitive science is precipitating a fundamental architectural shift in educational technology. We are moving from static, content-centric repositories to dynamic, learner-centric control systems capable of observing, modeling, and actuating pedagogical interventions with millisecond latency. This transformation is driven by four technical pillars: **Deep Knowledge Tracing (DKT)**, which leverages sequence modeling to infer latent knowledge states; **Contextual Multi-Armed Bandits (CMAB)**, which optimize instructional policies under uncertainty; **Microservices Architectures**, which provide the scalable, event-driven backbone for real-time inference; and **Privacy-Preserving Machine Learning (PPML)**, which ensures regulatory compliance through decentralized training paradigms like Federated Learning.

This report provides an exhaustive technical analysis of these domains. It dissects the evolution of knowledge tracing from Bayesian frameworks to state-of-the-art Transformer architectures like FlatFormer, evaluating them not merely on predictive accuracy but on the critical trade-offs between cognitive fidelity and inference latency. It explores the integration of psychometric frameworks—such as Item Response Theory (IRT) and Evidence-Centered Design (ECD)—into deep learning pipelines to create "stealth assessments" that infer competency from implicit behavioral signals. Furthermore, it details the engineering patterns required to deploy these models at scale, utilizing Feature Stores and stream processing to bridge the gap between offline training and online serving, all while navigating the rigorous data protection landscapes of FERPA and GDPR.

1. The Cognitive Engine: Evolution of Deep Knowledge Tracing Architectures

Knowledge Tracing (KT) constitutes the core predictive engine of any adaptive learning system. It is the task of modeling a learner's changing knowledge state over time based on their sequence of interactions with educational content. The field has undergone a

discontinuity, shifting from probabilistic graphical models to deep neural networks capable of capturing complex, non-linear, and long-term dependencies in learning trajectories.

1.1 From Bayesian Roots to Recurrent Neural Networks

For decades, **Bayesian Knowledge Tracing (BKT)** served as the industry standard. BKT models knowledge as a binary latent variable (Learned/Not Learned) updated via a Hidden Markov Model (HMM). It typically relies on four parameters per skill: initial knowledge probability ($P(L_0)$), the probability of learning the skill at each step ($P(T)$), the probability of guessing correctly despite not knowing the skill ($P(G)$), and the probability of slipping (making a mistake) despite knowing the skill ($P(S)$).[1] While BKT offers interpretability—allowing educators to see explicit probabilities of mastery—it suffers from rigid assumptions. It treats skills as independent entities, failing to capture the complex inter-skill correlations (e.g., how mastering algebra aids in physics) and struggles to model long-term temporal patterns or "learning transfer".[2]

The introduction of **Deep Knowledge Tracing (DKT)** by Piech et al. marked a paradigm shift. DKT treats knowledge tracing as a sequence prediction problem, utilizing **Recurrent Neural Networks (RNNs)** and **Long Short-Term Memory (LSTM)** networks to model the student's knowledge state as a high-dimensional, continuous latent vector.[3]

- **Input Representation:** Interactions are encoded as tuples (q_t, a_t), where q_t represents the exercise tag (skill ID) and a_t represents the binary correctness. These are typically processed via one-hot encoding or learned embeddings, creating a feature space size of $2M$ (where M is the number of unique skills).[4]
- **Temporal Dynamics:** The LSTM architecture addresses the "vanishing gradient" problem of standard RNNs, allowing the model to propagate information over long sequences. This enables DKT to capture historical context—such as a struggle with fractions three weeks ago—that might influence current performance in a way BKT's Markovian property cannot.[5]
- **Performance:** Early benchmarks indicated that DKT significantly outperformed BKT, achieving Area Under the Curve (AUC) scores in the range of 0.82–0.85 on datasets like ASSISTments.[1]

However, standard DKT is not without flaws. It suffers from two primary issues:

1. **Reconstruction Failure:** The model sometimes fails to predict correct performance on a skill immediately after the student has demonstrated mastery of that same skill.
2. Waviness: The predicted probability of mastery can fluctuate erratically over time steps, violating the pedagogical expectation that knowledge generally accumulates monotonically.
   DKT+ was introduced to address these specific shortcomings. It incorporates regularization terms into the loss function to penalize large, unjustified changes in prediction for unvisited concepts (smoothing the "waviness") and to enforce

reconstruction consistency.6 Despite these improvements, DKT+ sometimes exhibits lower generalization performance in certain heterogeneous datasets compared to the raw DKT baseline, highlighting the delicate balance between regularization and model capacity.7

1.2 The Transformer Revolution: Attention Mechanisms in KT

The sequential nature of RNNs presents a computational bottleneck: training cannot be parallelized, and inference latency grows linearly with sequence length. This limitation, combined with the desire to capture more explicit dependencies between specific past interactions and current problems, led to the adoption of **Transformer** architectures.

1.2.1 Self-Attentive Knowledge Tracing (SAKT)

SAKT adapts the Transformer's self-attention mechanism to the educational domain. Instead of compressing the entire history into a single hidden state, SAKT allows the model to "attend" to specific past interactions that are most relevant to the current prediction.[5]

- **Mechanism:** The model computes attention weights based on the similarity between the current exercise (Query) and past exercises (Keys). These weights are then applied to the past responses (Values) to generate a context-aware representation of the student's knowledge.
- **Relevance:** This is particularly powerful for identifying sparse dependencies. For instance, if a student fails a geometry problem, SAKT can specifically attend to their performance on algebra problems solved weeks prior, ignoring irrelevant intervening history in history or literature.
- **Performance:** Empirical evaluations on the College of Engineering (COE) dataset show SAKT achieving an AUC of **0.6692** and Accuracy of **0.7903**, notably outperforming the LSTM-based DKT (AUC 0.6474).[6] This suggests that the ability to explicitly reference past events provides a superior inductive bias for learning trajectories than pure recurrence.

1.2.2 Separated Self-Attentive Interaction Network (SAINT)

SAINT refines the Transformer approach by architecturally separating the processing of exercises and responses. It employs a deep encoder-decoder structure:

- **Encoder:** Processes the stream of exercise embeddings, capturing the structural relationships between questions (e.g., prerequisite hierarchies) independent of student performance.
- Decoder: Processes the stream of student responses (correctness) and integrates the encoder's output.
  This separation allows the model to learn complex content representations and student performance patterns distinctively before fusing them, leading to state-of-the-art (SOTA) performance on large-scale benchmarks like EdNet.5

1.2.3 Context-Aware Attentive Knowledge Tracing (AKT)

AKT introduces the concept of **monotonic attention**. Standard self-attention treats all past tokens as potentially equally relevant, which contradicts the learning science principle of memory decay—recent interactions are generally more indicative of current state than distant ones. AKT constrains the attention mechanism to prioritize recent history while still allowing "jumps" to highly relevant distant concepts. It also incorporates a Rasch-model-based embedding layer, explicitly modeling item difficulty (d) and discrimination (α) parameters within the neural network, effectively bridging classical psychometrics with deep learning.[8]

1.3 The Performance-Complexity Trap and FlatFormer

A critical trend identified in recent literature is the "Performance-Complexity Trap." As researchers chased marginal gains in AUC, architectures became increasingly baroque. Models like **HiTSKT** (Hierarchical Temporal-Session Knowledge Tracing) and **GKT** (Graph Knowledge Tracing) introduced deep hierarchical encoders and complex graph neural networks to explicitly model session structures and knowledge topology.[5] While these models capture nuance, they incur prohibitive computational costs, rendering them unsuitable for real-time deployment in systems serving millions of concurrent users.

FlatFormer represents a strategic pivot back to efficiency. It challenges the assumption that high cognitive fidelity requires deep structural stacking.

- **Information Injection:** Instead of complex hierarchies, FlatFormer uses a standard, lightweight "flat" Transformer but injects rich cognitive biases directly into the input embeddings and attention layers.
- **Hybrid Session Encoding:** It combines learnable session identifiers with fixed sinusoidal encodings, capturing the periodic nature of learning sessions without the overhead of hierarchical recurrent layers.[9]
- **Attention Logit Decay:** Crucially, it integrates a pre-computed **forgetting curve bias** directly into the attention logits. This explicitly models the Ebbinghaus forgetting curve, forcing the model to down-weight distant interactions unless they are highly semantically relevant.
- **Operational Superiority:** On the massive EdNet dataset, FlatFormer achieved an absolute AUC increase of **8.3%** over the hierarchical HiTSKT baseline while using **less than 15% of the parameters** and achieving **3x faster inference speeds**.[5] This makes FlatFormer a premier candidate for production environments where latency budgets are tight.

1.4 Inference Optimization for Production

In a live adaptive system, the DKT model must provide predictions (e.g., "what is the probability the student solves this next problem?") in real-time to guide the curriculum

navigation. A latency target of **sub-100ms** is standard to maintain user flow.[11] Large Transformers often exceed this budget.

- **Quantization:** Converting model weights from 32-bit floating-point (FP32) to 8-bit integers (INT8) reduces memory bandwidth consumption and accelerates matrix multiplications. Post-training quantization can often be applied with negligible accuracy loss (<1%).[12]
- **Knowledge Distillation:** For ultra-low latency scenarios (e.g., on-device inference), a large "Teacher" model (like an ensemble of SAINT models) can be used to train a compact "Student" model (like a shallow SAKT). The student learns to mimic the teacher's soft probability targets, retaining much of the teacher's predictive power while being orders of magnitude faster.[13]
- **Graph Optimization:** Tools like NVIDIA TensorRT or ONNX Runtime optimize the computational graph, fusing layers (e.g., fusing Conv2D and ReLU) and selecting optimal kernel implementations for the specific target hardware (GPU/TPU).[14]

2. The Decision Engine: Adaptive Pathways with Contextual Bandits

While DKT answers the question "What does the student know?", it does not answer "What should we teach next?". The latter is a sequential decision-making problem. While Reinforcement Learning (RL) via Markov Decision Processes (MDPs) is the theoretical ideal, the data requirements for training stable RL agents are often prohibitive. **Contextual Multi-Armed Bandits (CMAB)** offer a robust, intermediate solution that continuously balances exploration (gathering data on content effectiveness) and exploitation (delivering the best-known content).

2.1 The Contextual Bandit Formulation

In the educational CMAB setting:

- **Context (x_t):** A high-dimensional vector representing the learner's current state. This includes the knowledge state vector from the DKT model, recent engagement metrics (dwell time, click rate), and static demographics.[15]
- **Arms (A):** The set of available pedagogical actions. This could be selecting a specific problem, a video lecture, or an instructional strategy (e.g., "show a worked example" vs. "give a practice problem").
- **Reward (r_t):** The observed outcome, which acts as the supervisory signal. Defining this reward is the most critical engineering challenge in the system.[17]

2.2 Algorithm Selection: Thompson Sampling vs. LinUCB

Two primary algorithms dominate the landscape, each with distinct advantages for education.

2.2.1 Thompson Sampling (TS)

Thompson Sampling is a probabilistic algorithm that maintains a posterior distribution for the expected reward of each arm. In each step, it samples a value from each distribution and selects the arm with the highest sample.

- **Bayesian Nature:** It naturally models uncertainty. If a new educational video has few views, its reward distribution will be wide (high variance). The algorithm is likely to sample a high value from the "tail" of this distribution, leading to exploration. As data accumulates, the distribution narrows around the true mean (exploitation).[15]
- **Contextual TS:** In the linear payoff assumption ($r \approx x^T\theta$), the algorithm samples the weight vector θ from the posterior $P(\theta | \text{history})$. This allows for personalized exploration based on the specific learner's context features.[18]
- **Pedagogical Fit:** Empirical studies suggest TS often outperforms deterministic algorithms in education because its randomized nature prevents the system from getting stuck in repetitive loops—a common failure mode where a deterministic agent repeatedly recommends the same "optimal" content to a struggling student.[19]

2.2.2 Linear Upper Confidence Bound (LinUCB)

LinUCB calculates a deterministic score for each arm: $\mu_a + \alpha \cdot \sigma_a$, where μ_a is the predicted reward and σ_a represents the uncertainty (standard deviation).

- **Optimism in the Face of Uncertainty:** The algorithm optimistically assumes the arm is as good as its upper confidence bound. This explicitly directs exploration toward arms where the system has the least information (high σ) or high promise (high μ).[15]
- **Computational Efficiency:** LinUCB is computationally efficient and handles non-stationary environments well, making it suitable for systems where the content library changes frequently.[18]

2.3 Reward Function Engineering: The "Ghost in the Machine"

A naive reward function (e.g., $r=1$ if correct, $r=0$ if incorrect) is disastrous. It incentivizes the bandit to recommend trivially easy content to maximize the "correctness" rate, halting learning. The reward function must be a composite signal that proxies for **learning gain**.

- ZPD Targeting: The reward should be non-monotonic with respect to predicted correctness. Operational research on Vygotsky’s Zone of Proximal Development (ZPD) suggests the optimal learning zone lies between 35% and 70% success probability.1 A shaped reward function might look like:

  $$R = \begin{cases} 1 & \text{if } 0.35 \le P(\text{correct}) \le 0.70 \\ 0.5 & \text{if } P(\text{correct}) > 0.70 \text{ (Too Easy)} \\ 0.2 & \text{if } P(\text{correct}) < 0.35 \text{ (Too Hard)} \end{cases}$$

- **Learning Gain Proxy:** Ideally, the reward should reflect ΔKS (change in knowledge state). This can be estimated by querying the DKT model *before* and *after* the interaction. If the DKT model's confidence in a concept increases significantly after the student engages with a specific video, that video receives a high reward.[20]
- **Engagement Signals:** Incorporating normalized dwell time ("valid reads") or "Click > Skip" patterns (where a user skips multiple items to click one, indicating strong preference) ensures the content is not just pedagogically sound but also engaging.[21]

2.4 Cold-Start Mitigation and Hybrid Systems

Bandits struggle when N is small (new users or new content).

- **Content-Based Bootstrapping:** For new content items, we can initialize the bandit's parameters (θ) not randomly, but based on content metadata (e.g., text embeddings from BERT, topic tags). This allows the bandit to make educated guesses about a new video's effectiveness based on its similarity to effective videos.[2]
- **Cohort Clustering:** New students can be assigned to a "cluster" of similar learners (e.g., based on initial assessment or demographics). The bandit initially shares the interaction history (policy) of the cluster with the new user, accelerating convergence.[22]

3. The Behavioral Sensor: Stealth Assessment and Feature Engineering

Adaptive systems are only as good as their data. Relying solely on explicit quiz answers ignores the vast majority of the user experience. **Stealth Assessment** aims to infer competency continuously from implicit behaviors.

3.1 Evidence-Centered Design (ECD)

ECD provides the theoretical framework for stealth assessment. It consists of three models:

1. **Competency Model:** The set of variables we wish to measure (e.g., "Physics Understanding," "Persistence," "Creativity").
2. **Evidence Model:** The probabilistic relationship between observable behaviors and competency variables.
3. **Task Model:** The design of the environment to elicit these behaviors.

In games like *Physics Playground*, ECD is implemented using **Bayesian Networks**. If a student draws a lever to solve a puzzle, this observation propagates through the network, increasing the probability of the "Understanding Torque" node.[23] This allows for the assessment of "soft skills" like **persistence** (e.g., time spent on unsolved problems, number of restarts) and **creativity** (e.g., rarity of the solution method) alongside domain knowledge.[24]

3.2 Log Data Feature Engineering

Raw log data (clickstreams) is noisy and requires sophisticated engineering to become a useful signal for DKT or Bandit models.

- **Dwell Time Normalization:** Raw "time on page" is often meaningless (users leave tabs open). It must be normalized against the content length and complexity (expected reading speed). A "Valid Read" is a binary feature derived from this normalized dwell time, filtering out bounces and idle sessions.[21]
- **Implicit Feedback Signals:**
  - **Click > Skip Above:** If a user is presented with a list of recommendations and clicks the 3rd item, this is a positive signal for item 3 and a *negative* signal for items 1 and 2. Research shows this relative preference signal is more robust than absolute click counts.[1]
  - **Video Interaction:** Scrubbing backwards in a video often indicates confusion (high intrinsic load), while speeding up playback suggests the content is too easy (low load). These are critical inputs for determining ZPD alignment.[25]
- **Mouse and Eye Dynamics:** In the absence of eye-tracking hardware, mouse movement can serve as a proxy. Metrics like "trajectory deviation" (how straight the mouse moves to a target) and "hesitation time" correlate with cognitive load and uncertainty. High deviation often indicates high cognitive load.[26]

4. The Infrastructure: Microservices for Real-Time Adaptation

Deploying these complex models to serve millions of users requires a robust **Microservices Architecture**. The system must handle high-throughput event ingestion and provide low-latency inference.

4.1 Event-Driven Backbone (Kafka & Flink)

A synchronous Request/Response architecture (REST) is often too tightly coupled for this domain. An **Event-Driven Architecture** is preferred.

- **Apache Kafka:** Acts as the central nervous system. Every learner interaction (click, answer, pause) is published as an event to a Kafka topic. This decouples the "Producers" (learning apps) from the "Consumers" (analytics, DKT models, reporting), ensuring no data loss during traffic spikes.[28]
- **Apache Flink:** A stream processing engine that consumes from Kafka. Flink is essential for **stateful stream processing**. It can compute real-time features like "rolling average correctness over the last 10 minutes" or "current session duration" and update the Feature Store immediately. It manages the "state" of the learner in real-time.[28]

4.2 The Feature Store: Resolving Training-Serving Skew

A major cause of failure in production ML systems is **training-serving skew**—when the

feature calculation logic used during training (offline) differs from that used during inference (online). A **Feature Store** (e.g., Tecton, Feast) resolves this.[30]

- **Offline Store:** (e.g., Snowflake, S3) Stores years of historical data for batch training of complex DKT models. It is optimized for high-throughput scans.[32]
- **Online Store:** (e.g., Redis, DynamoDB) Stores only the *current values* of features (e.g., "Student X's current knowledge state vector") for low-latency retrieval.
- **Unified Logic:** The Feature Store ensures that the definition of a feature (e.g., "average score") is defined once in code and applied consistently to both offline batch generation and online stream processing.[33]

4.3 Optimizing for Sub-100ms Latency

To maintain a state of "flow," the system's response to a student's action (e.g., grading and recommending the next step) should effectively be instantaneous (<100ms).[11]

- **API Gateway:** Manages **latency budgets**. It enforces timeouts and circuit breaking (using patterns like Resilience4j). If the "Advanced Recommendation Service" takes too long (>50ms), the Gateway can failover to a "Simple Heuristic Service" to ensure the UI remains responsive.[11]
- **Edge Inference:** Moving the DKT inference closer to the user—to CDN edge nodes or even the user's device (via TensorFlow Lite)—eliminates network round-trip latency. This is particularly effective for interaction-heavy features like real-time handwriting recognition or mouse dynamics analysis.[34]
- **In-Memory Caching:** Redis is critical for caching DKT state vectors and Bandit policy parameters. A round-trip to a disk-based database is too slow for the inference loop.[35]

5. The Trust Layer: Privacy-Preserving Machine Learning

The aggregation of granular behavioral data (eye tracking, location, cognitive state) creates a "honey pot" of sensitive information. Compliance with **FERPA** (USA) and **GDPR** (EU) is not just a legal box-checking exercise but a fundamental architectural constraint.

5.1 Federated Learning (FL)

Federated Learning fundamentally changes the data paradigm. Instead of bringing the data to the model (centralized training), we bring the model to the data.[36]

- **Protocol:**
  1. The central server sends the current global DKT model weights to the student's device.
  2. The device trains the model locally using the student's private log data.
  3. Only the **model updates** (gradients) are sent back to the server; the raw log data

never leaves the device.
4. The server aggregates these updates (e.g., using Federated Averaging) to improve the global model.[37]

- **Data Minimization:** This architecture aligns perfectly with the GDPR principle of data minimization. Since the raw data is never centralized, the risk of a massive data breach is significantly mitigated.[38]

5.2 Differential Privacy (DP)

While FL keeps data local, it is theoretically possible to reconstruct training examples from the gradients sent to the server ("model inversion attacks"). **Differential Privacy (DP)** provides a mathematical guarantee against this.[39]

- **Noise Injection:** DP algorithms inject calibrated noise (e.g., Laplacian or Gaussian) into the gradients before they are shared (Local DP) or during aggregation (Central DP).
- **Privacy-Utility Trade-off:** There is an inherent tension. Increasing the noise (lower privacy budget ϵ) increases privacy but degrades model accuracy. However, research shows that fine-tuning pre-trained models with DP can achieve high accuracy while maintaining strong privacy guarantees.[39]

5.3 Regulatory Compliance

- **FERPA:** Systems must ensure that any data sharing (even for educational improvement) falls under specific exceptions (e.g., the "school official" exception). Federated Learning simplifies this by minimizing what is "shared" (gradients vs. records).[40]
- **GDPR:** The "Right to Explanation" poses a challenge for "black box" deep learning models. Integrating **Explainable AI (XAI)** techniques (like SHAP or LIME) into the pipeline allows the system to generate human-readable justifications for its recommendations ("We recommended this video because you struggled with the previous concept of 'Torque'"), satisfying transparency requirements.[1]

Conclusion

The architecture of next-generation adaptive learning systems is a synthesis of advanced AI and rigorous systems engineering. By replacing static rules with **Deep Knowledge Tracing** (specifically efficient models like **FlatFormer**), we gain a nuanced understanding of learner capability. By employing **Contextual Bandits** with carefully engineered reward functions targeting the **Zone of Proximal Development**, we optimize the learning path dynamically. **Microservices** and **Feature Stores** provide the necessary velocity for these decisions to happen in real-time, while **Federated Learning** ensures that this intelligence does not come at the cost of user privacy. This convergence lays the technical foundation for a future where education is not just accessible, but deeply, securely, and effectively personalized.

Works cited

1. Building Adaptive Educational Tools_ A Comprehensive Framework for Behavioral Learning Assessment.pdf
2. How Deep is Knowledge Tracing? - Computer Science, accessed January 5, 2026, https://home.cs.colorado.edu/~mozer/Research/Selected%20Publications/reprints/KhajahLindseyMozer2016.pdf
3. [1506.05908] Deep Knowledge Tracing - arXiv, accessed January 5, 2026, https://arxiv.org/abs/1506.05908
4. Incorporating Rich Features into Deep Knowledge Tracing, accessed January 5, 2026, https://d-miller.github.io/DRK12/topic1/447.pdf
5. FlatFormer: A Flat Transformer Knowledge Tracing Model Based on Cognitive Bias Injection, accessed January 5, 2026, https://arxiv.org/html/2512.06629v1
6. Deep Knowledge Tracing for Personalized Adaptive Learning at Historically Black Colleges and Universities - arXiv, accessed January 5, 2026, https://arxiv.org/pdf/2410.13876
7. Deep Knowledge Tracing for Personalized Adaptive Learning at Historically Black Colleges and Universities - arXiv, accessed January 5, 2026, https://arxiv.org/html/2410.13876v1
8. A Survey of Knowledge Tracing: Models, Variants, and Applications - arXiv, accessed January 5, 2026, https://arxiv.org/html/2105.15106v4
9. [2512.06629] FlatFormer: A Flat Transformer Knowledge Tracing Model Based on Cognitive Bias Injection - arXiv, accessed January 5, 2026, https://www.arxiv.org/abs/2512.06629
10. FlatFormer: A Flat Transformer Knowledge Tracing Model Based on Cognitive Bias Injection, accessed January 5, 2026, https://www.researchgate.net/publication/398476001_FlatFormer_A_Flat_Transformer_Knowledge_Tracing_Model_Based_on_Cognitive_Bias_Injection
11. Engineering Speed at Scale — Architectural Lessons from Sub-100-ms APIs - InfoQ, accessed January 5, 2026, https://www.infoq.com/articles/engineering-speed-scale/
12. MLOps for Low-Latency Applications: A Practical Guide - CloudFactory, accessed January 5, 2026, https://www.cloudfactory.com/blog/mlops-for-low-latency
13. [2509.17894] Optimizing Inference in Transformer-Based Models: A Multi-Method Benchmark - arXiv, accessed January 5, 2026, https://arxiv.org/abs/2509.17894
14. Best Practices — NVIDIA TensorRT Documentation, accessed January 5, 2026, https://docs.nvidia.com/deeplearning/tensorrt/latest/performance/best-practices.html
15. Thompson Sampling for Contextual Bandits with Linear Payoffs - Microsoft, accessed January 5, 2026, https://www.microsoft.com/en-us/research/wp-content/uploads/2013/06/report-contextual.pdf
16. Understanding contextual bandits: a guide to dynamic decision-making - Kameleoon, accessed January 5, 2026, https://www.kameleoon.com/blog/contextual-bandits

17. Contextual Bandits with Cross-Learning, accessed January 5, 2026, https://proceedings.neurips.cc/paper/2019/file/6aadca7bd86c4743e6724f9607256126-Paper.pdf
18. Noise-Adaptive Thompson Sampling for Linear Contextual Bandits - NeurIPS, accessed January 5, 2026, https://proceedings.neurips.cc/paper_files/paper/2023/file/4a6824f8f137e78f18e73d9cfc1d22ed-Paper-Conference.pdf
19. Feel-Good Thompson Sampling for Contextual Bandits: a Markov Chain Monte Carlo Showdown - arXiv, accessed January 5, 2026, https://arxiv.org/html/2507.15290v3
20. Gamification of Behavior Change: Mathematical Principle and Proof-of-Concept Study - NIH, accessed January 5, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC10998180/
21. Reweighting Clicks with Dwell Time in Recommendation - arXiv, accessed January 5, 2026, https://arxiv.org/pdf/2209.09000
22. Adaptive Learning Using Artificial Intelligence in e-Learning: A Literature Review - MDPI, accessed January 5, 2026, https://www.mdpi.com/2227-7102/13/12/1216
23. 1 Stealth Assessments' Technical Architecture Seyedahmad Rahimi1, Russell G. Almond2, Valerie J. Shute2 1University of Florida, accessed January 5, 2026, https://myweb.fsu.edu/vshute/pdf/architecture.pdf
24. Stealth Assessment - OAPEN Library, accessed January 5, 2026, https://library.oapen.org/bitstream/handle/20.500.12657/26058/1004027.pdf?sequence=1&isAllowed=y
25. What We Can Learn From Historic MOOC Data: Findings From Our Participation in the AIM Analytics Dropout Prediction Challenge - Center for Academic Innovation, accessed January 5, 2026, https://ai.umich.edu/blog/what-we-can-learn-from-historic-mooc-data-findings-from-our-participation-in-the-aim-analytics-dropout-prediction-challenge/
26. Use of mouse-tracking method to measure cognitive load - Arizona State University, accessed January 5, 2026, https://asu.elsevierpure.com/en/publications/use-of-mouse-tracking-method-to-measure-cognitive-load/
27. Using mouse cursor tracking to investigate online cognition: Preserving methodological ingenuity while moving toward reproducible science - PMC - PubMed Central, accessed January 5, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC8219569/
28. A Scalable Microservices Architecture for Real-Time Data Processing in Cloud-Based Applications - The Science and Information (SAI) Organization, accessed January 5, 2026, https://thesai.org/Downloads/Volume16No9/Paper_5-A_Scalable_Microservices_Architecture.pdf
29. A Containerized Microservices Architecture with Reinforcement Learning for Scalable, Adaptive Learning | Journal of Web Engineering - River Publishers, accessed January 5, 2026, https://journals.riverpublishers.com/index.php/JWE/article/view/30595

30. What is a Feature Store? A Complete Guide to ML Feature Engineering | Databricks Blog, accessed January 5, 2026, https://www.databricks.com/blog/what-feature-store-complete-guide-ml-feature-engineering
31. Feature Store for Machine Learning: Definition, Benefits - Snowflake, accessed January 5, 2026, https://www.snowflake.com/en/fundamentals/feature-store/
32. Accelerate and improve recommender system training and predictions using Amazon SageMaker Feature Store | Artificial Intelligence, accessed January 5, 2026, https://aws.amazon.com/blogs/machine-learning/accelerate-and-improve-recommender-system-training-and-predictions-using-amazon-sagemaker-feature-store/
33. Feature Stores for Machine Learning: The Backbone of Scalable ML Systems, accessed January 5, 2026, https://skphd.medium.com/feature-stores-for-machine-learning-the-backbone-of-scalable-ml-systems-0e76748f6306
34. Solving AI Foundational Model Latency with Telco Infrastructure - arXiv, accessed January 5, 2026, https://arxiv.org/html/2504.03708v1
35. How would you design an online feature store for machine learning systems (to serve features in real-time)? - Design Gurus, accessed January 5, 2026, https://www.designgurus.io/answers/detail/how-would-you-design-an-online-feature-store-for-machine-learning-systems-to-serve-features-in-real-time
36. Privacy Preserving Machine Learning With Federated Personalized Learning in Artificially Generated Environment | IEEE Journals & Magazine | IEEE Xplore, accessed January 5, 2026, https://ieeexplore.ieee.org/document/10691662/
37. Privacy-Preserving Robust Federated Learning with Distributed Differential Privacy, accessed January 5, 2026, https://ieeexplore.ieee.org/document/10063618/
38. [2503.13550] Towards Privacy-Preserving Data-Driven Education: The Potential of Federated Learning - arXiv, accessed January 5, 2026, https://arxiv.org/abs/2503.13550
39. Protecting Trained Models in Privacy-Preserving Federated Learning | NIST, accessed January 5, 2026, https://www.nist.gov/blogs/cybersecurity-insights/protecting-trained-models-privacy-preserving-federated-learning
40. Privacy and Data Sharing, accessed January 5, 2026, https://studentprivacy.ed.gov/privacy-and-data-sharing