International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Multi-Version Infrastructure for Privacy-Preserving AI/ML Inference at Scale

Jay Bankimchandra Desai

Submitted:03/02/2026 **Revised**: 16/03/2026 **Accepted:** 24/03/2026

Abstract: As the number of regulatory regimes, multi-stakeholder data relationships, and compliance requirements grows, privacy becomes an increasing architectural concern for large-scale AI/ML systems for data inference. Inference pipelines that apply a single, globally cast restrictive data policy to every inference context incur a measurable decrease in model performance. To avoid degrading model performance through globally restrictive policies while also avoiding potential policy violations introduced by dynamically modifying data usage per request, our multi-version architecture explicitly maintains multiple versions of user and participant information at the feature and embedding levels. In conjunction, context-aware version selection mechanisms deterministically map the metadata describing an incoming request to the appropriate data usage policy at runtime. In turn, versioned feature vectors are generated from superset representations of available signals, with the appropriate version selected based on the incoming request context and its corresponding data usage policy. Model-specific embeddings are derived from their privacy-compliant feature vectors to ensure end-to-end compliance. Rule-based selection schemes, implemented as abstractions decoupled from inference execution code, allow rapid regulatory adaptation without requiring service redeployment. Continuous monitoring helps validate selection quality and detect performance regressions in production environments. The computational overhead introduced by generating and maintaining multiple feature and embedding versions can be reduced through centralized build-once orchestration, shared feature storage schemas, and hybrid offline–online embedding generation within internet-scale latency budgets. Beyond privacy, this architectural pattern generalizes to fairness-aware inference, multi-tenant data isolation, and auditable policy enforcement, enabling versioned features and embedding representations as a foundational primitive for developing trustworthy, policy-compliant AI/ML systems.

Keywords: *Privacy-Preserving Inference, Multi-Version Feature Representations, Embedding Compliance, Differential Privacy, Regulatory-Aware Machine Learning*

1: Privacy Challenges in Large-scale AI/ML Inference Systems

Modern AI/ML inference systems today power a growing number of real-time decision-making applications, from advertisement ranking to content recommendation to e-commerce personalization. From a high-level view, such systems typically require feature extraction from structured signals — including but not limited to user activities, context, and interaction, and historical data — and require an additional step of feature embedding to encode those features into dense representations for downstream models [1]. The computational performance of these systems is directly tied to the expressiveness and fidelity of these representations.

San Jose State University, USA

The scale of such server-side inference pipelines is non-trivial: production systems must serve hundreds of thousands of concurrent requests per second while enforcing data usage policies deterministically across every inference path [1].

However, as inference pipelines are rolled out across different geographies and regulatory jurisdictions, it is increasingly untenable to assume that all available user data can be used uniformly across every inference context [8]. Privacy laws vary by geography and use case. Data that is permissible for targeting in one geographic region may be prohibited in another, by law or by contract, depending on the kind of service delivered and the types of customer segments targeted. The problem of indiscriminate data usage is well understood: even coarse-grained demographic attributes, when combined at the

feature level, are sufficient to uniquely identify a substantial portion of users in large-scale datasets, making it critical that the inference pipeline enforces strict, context-aware data usage constraints [7].

The risk of re-identification through inference pipelines is not hypothetical. Research has demonstrated that even sparse, seemingly anonymized datasets can be de-anonymized when an adversary correlates a small number of known data points against a large dataset — a threat directly relevant to inference systems that aggregate multiple user signals into feature vectors [2]. When an inference pipeline passes a rich set of user features through multiple downstream models without enforcing strict, version-aware signal suppression, it creates the same conditions that enable re-identification attacks: a broad feature space that, even when partially observed, can uniquely distinguish individual users. This underscores why suppressing non-permitted signals at the feature level — rather than relying on post-hoc anonymization — is a foundational requirement of the multi-version architecture.

Privacy problems are further compounded when restrictions on inference are not just placed on end users but also on other participants on the platform, such as advertisers, sellers, or content creators. In this case, the signals of one participant must be blocked from competing participants at the level of individual features. In the absence of such a scheme, both regulatory non-compliance and a loss of community trust threaten the viability of the infrastructure, motivating a multi-version infrastructure that can dynamically enforce context-sensitive privacy without compromising inference quality when scaled [1].

2: Architecture of the multi-version feature representations

The core principle of multi-version inference systems is the construction of multiple compliant versions of user and participant data as needed. Whereas the customary approach to modeling privacy enforcement is to assume the presence of a single privacy enforcer that applies the same privacy requirement across all requests, privacy requirements are modeled here as a finite set of versions, each corresponding to a distinct combination of data usage constraints [3]. Versions are created based on request context metadata that consists of user geography, regulatory jurisdiction, ranking event type, model category, and client classification. The combination of these metadata elements uniquely determines the set of versions to be constructed for the inference request.

To generate the feature vectors, a superset of all available features is assembled from the relevant data sources. A version-specific feature vector is then obtained by applying constraints that zero out or remove features not permitted by the applicable policy for that version. Each version thus represents a valid, policy-compliant projection of the full feature space, ensuring that inference at any version boundary never exposes signals that violate the corresponding data usage policy [3].

The importance of controlling signal exposure at inference time is further supported by research into privacy-preserving deep learning, which has shown that even partial sharing of model parameters or intermediate representations can leak sensitive information about the underlying data [4]. In the context of a multi-version inference pipeline, this means that it is insufficient to simply restrict features at the input layer if intermediate representations — such as activations or embeddings — are shared or reused across version boundaries. Each version's signal constraints must propagate consistently through every layer of the inference stack, from the raw feature vector through to the final model output, to ensure that restricted signals cannot be recovered or inferred from intermediate states.

In practice, this is most commonly achieved by maintaining separate feature representations per version, disabling access to restricted features within each, so that architecture continuity is preserved while only policy-permitted signals are exposed. In-memory representations of multi-version feature vectors are separated into their own virtual containers to prevent signal pollution across versions. This ensures that signals in one version do not leak into other inference paths. When inference stages span microservices, transport-safe representations carry versioned vectors across service boundaries while retaining their version identity and preventing serialization overhead [3].

3: Creating versioned embeddings for model compliance

Although versioned feature vectors define the valid signal space for a given inference context, model

embeddings must also be produced to be compliant with end-to-end privacy enforcement. Model embeddings are dense numerical representations derived from a feature vector, and their values are specific to each AI/ML model that consumes that feature vector as input. The embedding space of one model is not transferable to another without loss of compliance or performance [5]. This is particularly relevant at the scale of modern embedding-based architectures: transformer-based language models such as BERT BASE and BERT LARGE — with 110M and 340M parameters respectively — maintain incompatible representation spaces, making it essential that each model's embeddings are derived from its own version-specific, policy-compliant feature vector [5].

Allowing a model to use embeddings derived from an unconstrained feature vector would violate the privacy guarantee established at the feature layer. Each model embedding must therefore be traceable back to the specific versioned feature vector from which it was generated, ensuring that privacy constraints propagate end-to-end through the inference pipeline. Breaks in this chain — such as reusing embeddings across versions or deriving embeddings from a superset feature vector — directly undermine the compliance guarantees of the multi-version architecture [5].

There are several approaches to managing the computational cost of embedding generation. In less latency-sensitive applications, embeddings can be pre-generated offline and retrieved at inference time, provided they are validated against the correct feature vector version. In high-stakes, latency-critical applications, embeddings may be computed on the fly from the versioned feature vector at request time [6]. In a distributed inference pipeline, when the first service builds all feature vector versions, it can simultaneously generate all corresponding model embeddings and propagate only the version-validated embeddings downstream. This build-once approach prevents downstream services from needing to re-derive embeddings independently, reducing both latency and the risk of version mismatches [5].

Component	Description	Compliance Requirement
BERT BASE	Transformer-based language model	Version-specific embedding space
BERT LARGE	Larger transformer-based language model	Incompatible with BERT BASE embedding space
Offline embeddings	Pre-generated for non-critical applications	Must be validated against the feature vector
On-the-fly embeddings	Generated during high-stakes inference	Must reflect constrained feature version
Training corpora	Large text datasets for embedding training	Must be processed without modification

Table 1: Versioned Embedding Compliance Requirements and Approaches [5, 6]

4: Version selection mechanisms and rule-based frameworks

Once feature vectors and embeddings have been built for each version, a deterministic mechanism is required to select the correct version for a given inference request. Version selection maps the context attributes of the inference request — user geographic location, regulatory jurisdiction, ranking event type, model family, and client classification — to a single version of feature vectors and embeddings, which are then propagated to all downstream AI/ML models participating in that inference request [7].

One of the most important properties of this selection process is its determinism and consistency. Since inference requests carry contextual attributes that can be highly specific — including geographic region, platform segment, and content category — version selection logic must ensure that every combination of such attributes maps unambiguously to a compliant data version. Failure to do so risks applying overly permissive data policies to certain request contexts, potentially exposing restricted signals that should have been suppressed [7].

The granularity of versioning is adaptable. Different platform contexts can have vastly different compliance requirements: some may need only a coarse-grained separation between geographic regions governed by different data use legislation, while others may require finely separated event types, model usage categories, and client classifications to satisfy regulatory requirements. This allows the infrastructure to scale over time as new regulatory requirements are established.

Versioning is governed by a rule-based approach that decouples data usage policies from the inference execution code, enabling scalability and maintainability [8]. This separation is also required under major regulatory frameworks such as the GDPR, which imposes fines of up to €20 million or 4% of global annual turnover for violations that are not correctly auditable [8]. Selection rules consist of conditions, priorities, and fallbacks that map request contexts to version identifiers. They are stored externally to the inference service code so that compliance teams can add or deprecate selection rules without deploying a new version of the inference service. Breach notification requirements — including the obligation to notify supervisory authorities within 72 hours — further reinforce the need for version selection systems to be continuously monitored and reliably correct [8].

In practice, monitoring systems track the distribution of version selections across requests, the rate of unexpected version assignment changes, and inference performance regressions. These signals can be used to validate assumptions in the selection rules and to tune them over time without disrupting production inference [7].

5: Optimizing Multi-Version Pipelines for Scalability

Building a multi-version inference infrastructure at internet scale introduces meaningful computational, memory, and networking overhead that must be tightly controlled to meet the latency budgets and cost profiles of production AI/ML systems. A set of complementary optimizations can minimize this overhead without sacrificing the correctness or completeness of privacy enforcement.

The first and most important optimization is to build once and reuse many times. Large-scale inference pipelines are typically decomposed into microservices that handle distinct stages of ranking and prediction. Without centralized version construction, every feature vector and embedding version would be independently recomputed at each stage of the pipeline. Instead, the first service in the pipeline constructs all versions at once and propagates the complete, pre-validated set downstream to all subsequent services. This is analogous to how Apache Spark persists intermediate RDDs in memory rather than recomputing them across stages, achieving speedups of over 100× in iterative workloads [9]. Centralized construction ensures that versions are built only once per request, keeping latency overhead constant as the number of downstream services grows.

The second optimization exploits the shared structure of feature data to reduce memory usage and serialization cost. Feature storage schemas can be optimized to factor out shared, non-restricted features into a common "safe" base representation, while storing only the incremental, version-specific features separately [10]. This approach avoids redundant storage of common signals across versions and allows any version-specific feature vector to be reconstructed at inference time by combining the shared base with its corresponding version-specific delta. The resulting reduction in total memory footprint and serialization overhead is especially significant when the number of active versions is large relative to the size of the version-specific feature deltas.

The third optimization involves externalizing version configuration from the inference service itself. Storing version definitions and selection rules in externalized, runtime-readable configuration allows compliance teams to introduce new versions or modify existing ones without requiring a full service redeployment. This decoupling of policy from code accelerates the response to new regulatory requirements and enables controlled performance comparisons across different privacy policy configurations in production [9].

Optimization Strategy	Primary Benefit	Inspired By
Build once, reuse many times	Reduces redundant computation	Spark RDD persistence

Shared base feature storage	Reduces memory redundancy	MSR code structure
Externalized version configuration	Enables compliance without redeployment	Spark library composition

Table 2: Key Optimization Strategies in Multi-Version Inference Pipelines [9, 10]

6: Interpretation and Future Directions in Policy-aware Inference

The multi-version infrastructure model for privacy-preserving AI/ML inference demonstrates that complex and dynamic privacy constraints can be enforced at the feature and embedding levels without reducing inference performance to the lowest common denominator of the most restrictive policy baseline. However, the implications of this architecture extend well beyond regulatory compliance. The same pattern of versioned data representation, deterministic selection, and decoupled policy management is directly applicable to fairness-aware inference and data isolation in multi-tenant AI/ML platforms [11].

Version selection in production inference pipelines operates over a bounded, low-dimensional context space — defined by attributes such as geography, jurisdiction, model family, and event type. This structure is well-suited to optimization techniques from Bayesian optimization, which are most effective over continuous domains of fewer than 20 dimensions [11]. Acquisition functions such as expected improvement and the knowledge gradient could be applied to automatically learn and refine version selection policies based on observed inference outcomes, subject to the constraint that per-request overhead must remain negligible. Given that the knowledge gradient acquisition function performs within 98% of the multi-step optimal algorithm across a wide range of problem settings [11], near-optimal policy updates are achievable without manual tuning.

Regulatory convergence also reinforces the need for this type of architecture. Recent regulatory frameworks, including European Union requirements that algorithms producing decisions which "significantly affect" individuals must be explainable [12], create strong incentives for auditable, policy-aware inference systems. Versioned feature representations directly support post-hoc compliance auditing: every inference output can be associated with the specific feature version and data usage policy that governed it. With interpretability and machine learning now the subject of over 20,000 papers indexed on Google Scholar in the past five years [12], the demand for transparent, auditable inference systems will only intensify.

Future work includes the automatic generation of policy-to-version mappings and the integration of differential privacy and federated learning methods within the multi-version framework. The intersection of privacy engineering, regulatory technology, and high-performance inference infrastructure represents one of the most consequential frontiers in trustworthy AI deployment [12].

Conclusion

As deployments of AI/ML inference systems continue to scale across geographical, regulatory, and multi-stakeholder platform boundaries, context-aware dynamic privacy enforcement can no longer be relegated to an afterthought or implemented as a static global policy applied uniformly across all inference workloads. The multi-version architecture provides a principled framework for scalable enforcement of privacy-inference trade-offs by modeling privacy-differentiated data usage in terms of versioned representations at the feature and embedding levels. These versions are selected at runtime in a deterministic manner using rule-based approaches that are decoupled from inference execution code, and their construction is coordinated through centralized orchestration with optimized shared feature storage. This architecture extends naturally to fairness-aware inference, multi-tenant data isolation, and regulatory auditability, with versioned feature and embedding representations acting as a general-purpose primitive enabling responsible AI/ML deployment across a wide range of applications. These needs will only grow as privacy regulation expands internationally and the demand for explainable, auditable inference systems intensifies. Bridging privacy engineering, policy-aware architecture, and high-performance inference infrastructure is a critically important technical frontier for scaling trustworthy AI systems.

References

[1] H. Brendan McMahan et al., "Communication-Efficient Learning of Deep Networks from Decentralized Data," Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017. [Online]. Available: https://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf

[2] Arvind Narayanan and Vitaly Shmatikov, "Robust De-anonymization of Large Sparse Datasets," 2007. [Online]. Available: https://www.stat.cmu.edu/~brian/303-2012-full/303-2011/303-2010/0-from%20the%20world/2010-03-12-de-anonymizing%20netflix.pdf

[3] Martín Abadi et al., "Deep Learning with Differential Privacy," ACM Digital Library, 2016. [Online]. Available: https://dl.acm.org/doi/epdf/10.1145/2976749.2978318

[4] Reza Shokri, Vitaly Shmatikov, "Privacy-Preserving Deep Learning," ACM Digital Library, 2015. [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/2810103.2813687

[5] Jacob Devlin, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2019. [Online]. Available: https://aclanthology.org/N19-1423.pdf

[6] Christian Janiesch et al., "Machine Learning and Deep Learning," Electronic Markets, 2021. [Online]. Available: https://link.springer.com/content/pdf/10.1007/s12525-021-00475-2.pdf

[7] Latanya Sweeney, "k-Anonymity: A Model for Protecting Privacy," International Journal on Uncertainty, 2002. [Online]. Available: https://homepage.divms.uiowa.edu/~sriram/5980/spring16/k-anonymity1.pdf

[8] Foot Anstey, "The General Data Protection Regulation: A Practical Guide to the Changes Ahead," 2018. [Online]. Available: https://www.faintranet.co.uk/wp-content/uploads/FOOT-ANSTEY-GDPR-_Digital-Version.pdf

[9] Matei Zaharia et al., "Apache Spark: A Unified Engine for Big Data Processing," ACM Digital Library, 2016. https://dl.acm.org/doi/pdf/10.1145/2934664

[10] Alexandros G. Dimakis et al., "A Survey on Network Codes for Distributed Storage," Proceedings of the IEEE, Vol. 99, No. 3, March 2011.

[11] https://www.academia.edu/10344819/I_N_V_I_A_Survey_on_Network_Codes_for_Distributed_Storage

[12] Peter I. Frazier. "A Tutorial on Bayesian Optimization." arXiv, 2018. https://arxiv.org/pdf/1807.02811

[13] Finale Doshi-Velez and Been Kim, "Towards a Rigorous Science of Interpretable Machine Learning," arXiv, 2017. https://arxiv.org/pdf/1702.08608