FORGE CORE: A Causal Model-Agnostic Intelligence and Routing Engine for Enterprise AI Deployment

A FORGE OS Subsystem Specification

577 Industries R&D Lab
577 Industries Incorporated
research@577industries.com

Abstract

Deploying foundation language models in regulated, resource-constrained, or security-sensitive environments requires systematic post-training procedures that address task adaptation, computational efficiency, behavioral alignment, and governance compliance simultaneously. Existing approaches typically address these concerns in isolation, leading to integration failures, unpredictable trade-offs, and pilot-to-production gaps that limit enterprise adoption. This paper presents FORGE CORE, the model-agnostic intelligence and routing engine of FORGE OS—the Agent-Legible Operating System for enterprise AI. FORGE CORE serves as the "brain" of FORGE OS, providing three integrated capabilities: (1) **Ontology-Grounded Semantic Retrieval**, employing zero-shot neural extraction via LayoutLMv3 and large language model pipelines to construct RDF/OWL 2/SHACL knowledge graphs in under two hours versus 72 hours for manual expert engineering; (2) **Causal Model Routing Engine**, using Double Machine Learning with Causal Forests to estimate Conditional Average Treatment Effects (CATE) for each routing decision, producing per-decision natural-language explanations that satisfy EU AI Act Article 13 transparency requirements while achieving 75% cost reduction versus always-frontier routing; (3) **Staged Post-Training Pipeline**, systematically transforming foundation models through Adapt $\rightarrow$ Compress $\rightarrow$ Align $\rightarrow$ Deploy stages with continuous online distillation, drift detection, and canary deployment for zero-downtime model updates. Comprehensive experiments across Llama-2 (7B, 13B, 70B) and Mistral-7B on seven benchmarks demonstrate 94.2% average task performance retention, 73.8% latency reduction, 81.2% memory footprint reduction, and 31.4% safety improvement. The causal routing engine achieves 75.2% cost reduction with less than 5% quality degradation and greater than 90% causal explanation fidelity. Neural ontology extraction achieves 0.91 F1 on class extraction across three enterprise domains. All capabilities integrate with FORGE OS subsystems: FORGE QBIT provides identity verification at model API boundaries, FORGE MEMORY provides immutable audit trails for every routing decision and distillation job, and FORGE KINETIC provides edge deployment targets. Case studies in defense (IL5/IL6 sovereign deployment), healthcare (HIPAA-compliant documentation), financial services (EU AI Act compliance), and edge computing validate practical applicability.

1 Introduction

The rapid advancement of foundation language models has created unprecedented opportunities for automating complex cognitive tasks across industries. Models such as GPT-4 [OpenAI, 2023], Llama-2 [Touvron et al., 2023], Mistral [Jiang et al., 2023], Claude [Anthropic, 2024], and Qwen [Bai et al., 2023] demonstrate remarkable capabilities in natural language understanding, generation, and reasoning. However, deploying these models in regulated, resource-constrained, or security-sensitive environments remains a significant engineering challenge that extends far beyond the initial pretraining phase [Bommasani et al., 2021, Liang et al., 2022].

1.1 The Pilot-to-Production Gap

Enterprise adoption of generative AI exhibits a characteristic funnel: organizations rapidly investigate and pilot foundation models but face substantial friction when transitioning to production deployment. A 2025 study by MIT's Project NANDA found that approximately 60% of organizations investigate task-specific generative AI tools, 20% reach pilot stage, and only 5% achieve production deployment [Challapally et al., 2025]. This *pilot-to-production gap* arises from four interacting factors that existing approaches address only partially:

1. **Task Adaptation.** Transferring general capabilities to domain-specific requirements while preserving beneficial base behaviors—a balance that naive fine-tuning often fails to achieve [Kirkpatrick et al., 2017].
2. **Computational Constraints.** Deployment environments frequently preclude direct use of full-precision multi-billion parameter models, necessitating compression that introduces quality trade-offs [Dettmers et al., 2022].
3. **Behavioral Alignment.** Safety requirements and domain-specific policy constraints vary across deployment contexts [Bai et al., 2022a, Casper et al., 2023].
4. **Governance Compliance.** Regulated industries demand complete auditability from training data through deployed artifacts—a provenance chain that ad-hoc approaches cannot provide [NIST, 2023].

Current practice addresses these concerns through disconnected toolchains and manual integration, leading to three failure modes: optimization conflicts where improvements in one dimension degrade another; evaluation gaps where stage-specific metrics fail to predict end-to-end behavior; and governance breaks where artifact provenance is lost across tool boundaries.

1.2 The Model Selection Problem

Beyond post-training, organizations face a second systemic inefficiency: *model selection*. Enterprise deployments typically route all queries to a single frontier model (e.g., GPT-4o, Claude Opus), wasting an estimated 80% or more of AI compute budget on queries that simpler, cheaper models could handle with equivalent quality [Chen et al., 2023]. Simple factual lookups do not require the same model that handles complex multi-step reasoning.

Existing routing approaches—heuristic rules, A/B testing, contextual bandits—operate as black boxes that cannot explain *why* a particular model was selected. In regulated environments subject to the EU AI Act Article 13 transparency requirements [European Union, 2024] or U.S. federal oversight [NIST, 2023], unexplainable routing decisions create compliance risk. The field lacks a routing framework that is simultaneously cost-effective, quality-preserving, and *causally explainable*.

1.3 The Ontology Bottleneck

Enterprise knowledge graphs—the semantic substrate that grounds AI responses in domain-specific meaning—require weeks of manual expert extraction using ontology engineering methodologies [Noy and McGuinness, 2001]. This *ontology bottleneck* prevents rapid deployment in new domains: an organization cannot benefit from ontology-grounded retrieval until domain experts manually construct the knowledge graph, a process that typically requires 72+ hours of skilled labor per domain.

Recent advances in document understanding (LayoutLMv3 [Huang et al., 2022]) and large language model reasoning create an opportunity for zero-shot neural ontology extraction that reduces onboarding from days to hours.

1.4 FORGE CORE as the Brain of FORGE OS

FORGE CORE addresses all three challenges—post-training, model routing, and ontology engineering—through a unified intelligence engine that serves as the "brain" of FORGE OS, the Agent-Legible Operating System for enterprise AI [577 Industries, 2025a]. FORGE CORE *understands* enterprise domains (ontology), *selects* optimal models for each query (routing), and *distills* intelligence into deployable form (post-training)—all while emitting structured telemetry that enables the remaining FORGE OS subsystems to govern, secure, and distribute the resulting intelligence.

FORGE CORE is model-agnostic: it works with any foundation model family, including GPT-4, Claude, Llama, Mistral, Qwen, and domain-specific models. The routing engine treats models as interchangeable treatments in a causal inference framework, selecting the optimal model per query based on estimated treatment effects rather than fixed heuristics.

1.5 Contributions

The primary contributions of this work are sevenfold:

1. **Zero-Shot Neural Ontology Extraction:** An automated pipeline using LayoutLMv3 document understanding and LLM-based reasoning to construct RDF/OWL 2/SHACL knowledge graphs from enterprise documents, reducing onboarding from 72 hours to under 2 hours with comparable extraction quality (Section 4.1).
2. **Causal Model Routing Engine:** A Double Machine Learning framework using Causal Forests to estimate per-query Conditional Average Treatment Effects (CATE) for model selection, producing natural-language explanations that satisfy regulatory transparency requirements (Section 4.2).
3. **Staged Post-Training Pipeline:** The Adapt $\rightarrow$ Compress $\rightarrow$ Align $\rightarrow$ Deploy pipeline with formal stage-ordering justification based on gradient interference analysis, achieving 94.2% task retention with 73.8% latency and 81.2% memory reduction (Section 4.3).
4. **Continuous Online Distillation:** A drift detection and canary deployment system that enables zero-downtime model updates through LoRA-based fine-tuning on buffered production queries (Section 4.4).
5. **Comprehensive Governance Framework:** Cryptographically signed release artifacts with full provenance chains, stored in FORGE MEMORY's Immutable Governance Object Model (Section 4.5).
6. **FORGE OS Telemetry Integration:** Emission of `RETRIEVAL_HIT`, `ROUTING_DECISION`, and `TOOL_CALL` events into the unified `ForgeEvent` stream (Section 5).
7. **FORGE QBIT Identity Verification:** X.509 certificate-based identity verification at all model API boundaries for non-repudiation (Section 5).

1.6 Paper Organization

Section 2 reviews related work across ontology engineering, model routing, fine-tuning, compression, alignment, and deployment. Section 3 formalizes the model-agnostic intelligence problem and derives stage-ordering principles. Section 4 presents the FORGE CORE architecture in detail. Section 5 describes FORGE OS telemetry integration. Section 6 presents experimental evaluation. Section 7 provides domain-specific case studies. Section 8 discusses findings and limitations. Section 9 concludes with future directions.

2 Related Work

We organize related work into seven categories corresponding to FORGE CORE's capability modules and pipeline stages.

2.1 Ontology Engineering and Knowledge Graphs

Ontology engineering—the construction of formal, machine-readable domain models—has traditionally been a manual, expert-driven process. Methodologies such as METHONTOLOGY [Fernández-López et al., 1997] and the NeOn methodology [Suárez-Figueroa et al., 2012] provide structured approaches but require weeks of domain expert effort. The OWL 2 Web Ontology Language [Grau et al., 2008] and SHACL Shapes Constraint Language [Knublauch and Kontokostas, 2017] provide expressive representation formalisms, but constructing ontologies that leverage these standards remains labor-intensive.

Recent work explores automated ontology construction. LayoutLMv3 [Huang et al., 2022] achieves state-of-the-art performance on document understanding tasks by pre-training multimodal transformers on text, layout, and image features simultaneously. LLM-based ontology extraction [Babaei Giglou et al., 2023] demonstrates that large language models can identify candidate classes and relations from unstructured text. However, no existing system combines document understanding with LLM extraction, automatic SHACL constraint generation, and human-in-the-loop refinement into a production pipeline with sub-two-hour onboarding guarantees.

2.2 Model Routing and Selection

The model selection problem has received increasing attention as the diversity and cost spectrum of available LLMs has expanded. FrugalGPT [Chen et al., 2023] proposes LLM cascading, where cheaper models are tried first and queries are escalated only when confidence is low. RouteLLM [Ong et al., 2024] trains a preference-based router using data from Chatbot Arena. Martian [Martian, 2024] applies contextual bandits for dynamic model selection.

Contextual bandits, particularly LinUCB [Li et al., 2010], provide a principled exploration-exploitation framework for model selection but treat the routing decision as a prediction problem rather than a causal inference problem. This distinction matters in regulated environments: bandit-based routers can report which model performs best empirically, but cannot explain *why* model m_i is expected to outperform m_j for a specific query in terms of causal features.

Double Machine Learning (DML) [Chernozhukov et al., 2018] provides a framework for estimating heterogeneous treatment effects with valid inference. Causal Forests [Wager and Athey, 2018, Athey et al., 2019] extend this to non-parametric CATE estimation. FORGE CORE is the first system to apply DML with Causal Forests to LLM routing, producing per-decision causal explanations that satisfy regulatory transparency requirements.

2.3 Parameter-Efficient Fine-Tuning

Full fine-tuning of billion-parameter models risks catastrophic forgetting and requires substantial compute [Ramasesh et al., 2022]. Parameter-efficient fine-tuning (PEFT) methods address these limitations. LoRA [Hu et al., 2022] introduces trainable low-rank decomposition matrices into attention layers, reducing trainable parameters by $10{,}000\times$ while maintaining performance within 1–2% of full fine-tuning. QLoRA [Dettmers et al., 2023] enables adapter training over 4-bit quantized bases using NF4 data types, reducing memory by $\sim 4\times$. Alternative approaches include Adapters [Houlsby et al., 2019], Prefix Tuning [Li and Liang, 2021], and $(\text{IA})^3$ [Liu et al., 2022]. Comparative studies [Lialin et al., 2023] show that LoRA achieves the best trade-off for most scenarios, motivating its selection as the primary adaptation method in FORGE CORE.

2.4 Model Compression

Deployment constraints frequently require reducing model size and inference cost. Knowledge distillation [Hinton et al., 2015] trains smaller student models to match teacher outputs. DistilBERT [Sanh et al., 2019] demonstrated 40% size reduction with 97% capability retention. Post-training quantization methods—GPTQ [Frantar et al., 2023] and AWQ [Lin et al., 2023]—enable 3–4 bit weight representation with minimal accuracy loss. SmoothQuant [Xiao et al., 2023] migrates quantization difficulty from activations to weights for INT8 deployment. Structured pruning [Michel et al., 2019, Xia et al., 2023] removes architectural components but often requires significant retraining.

2.5 Alignment and Safety Tuning

RLHF [Christiano et al., 2017, Ouyang et al., 2022] represents the predominant alignment paradigm. Direct Preference Optimization (DPO) [Rafailov et al., 2023] provides a simpler alternative by optimizing from preference pairs without explicit reward models. Constitutional AI [Bai et al., 2022b] introduces rule-based principles for scalable alignment. Domain-specific alignment extends these methods to healthcare [Singhal et al., 2023] and financial services [Wu et al., 2023].

2.6 Deployment Engineering and MLOps

Serving engines such as vLLM [Kwon et al., 2023] and TensorRT-LLM [NVIDIA, 2023] optimize throughput through continuous batching, PagedAttention, and kernel fusion. Speculative decoding [Leviathan et al., 2023] achieves 2–3$\times$ speedups using draft-verify paradigms. MLOps practices [Sculley et al., 2015] provide infrastructure for versioning, testing, and monitoring, but existing frameworks lack support for generative model characteristics [Chen et al., 2024].

2.7 Integrated Post-Training Approaches

LLaMA-Factory [Zheng et al., 2024] provides a unified fine-tuning interface. Axolotl [OpenAccess AI Collective, 2023] emphasizes configuration-driven workflows. However, these systems address tool integration rather than systematic pipeline design: they lack formal stage interfaces, acceptance criteria, governance artifacts, principled stage ordering, and continuous learning capabilities. FORGE CORE addresses these gaps by providing an architectural specification with theoretical grounding, causal routing, ontology management, and comprehensive evaluation methodology.

3 Problem Formulation

This section formalizes the model-agnostic intelligence problem underlying FORGE CORE and derives theoretical justification for the pipeline stage ordering.

3.1 The Model-Agnostic Intelligence Problem

Definition 1 (Model-Agnostic Intelligence) *Given a set of foundation models $\mathcal{M} = \{m_1, \ldots, m_K\}$ with heterogeneous capabilities, costs c_k, and latencies ℓ_k; an enterprise knowledge base $\mathcal{K}$ with domain ontology $\mathcal{O}$; and a stream of queries $\mathcal{Q} = \{q_1, q_2, \ldots\}$, the* model-agnostic intelligence problem *is to route each query q to the optimal model $m^*(q)$ that minimizes cost while satisfying quality and compliance constraints:*

$$m^*(q) = \arg\min_{m \in \mathcal{M}} c(m) \quad \textit{subject to} \quad \textit{quality}(m, q) \geq \tau_q, \;\; \textit{compliance}(m, q) \geq \tau_c \tag{1}$$

where τ_q and τ_c are quality and compliance thresholds defined by deployment policy.

This problem subsumes model selection, post-training, and knowledge management: the quality function depends on how well each model has been adapted (post-training), the compliance function depends on governance artifacts and audit trails, and effective routing requires understanding query semantics (ontology).

3.2 Multi-Objective Optimization for Post-Training

Let M_0 denote a pretrained foundation model with parameters $\theta_0 \in \mathbb{R}^d$. Post-training seeks to transform M_0 into a mission-ready model M^* satisfying task performance, efficiency, safety, and governance requirements. We model the deployment environment through a constraint set $\mathcal{C} = (\mathcal{H}, \mathcal{L}, \mathcal{S}, \mathcal{G})$ where $\mathcal{H}$ specifies hardware constraints, $\mathcal{L}$ specifies latency and throughput requirements, $\mathcal{S}$ specifies safety and policy compliance thresholds, and $\mathcal{G}$ specifies governance and auditability requirements.

Post-training is formalized as a constrained multi-objective optimization:

$$\theta^* = \arg\min_{\theta} \left[\mathcal{L}_{\text{task}}(\theta) + \lambda_1 \mathcal{L}_{\text{compress}}(\theta) + \lambda_2 \mathcal{L}_{\text{align}}(\theta) + \lambda_3 \mathcal{L}_{\text{deploy}}(\theta)\right] \tag{2}$$

subject to $\text{mem}(\theta) \le h_{\text{mem}}$, $\text{lat}(\theta) \le \ell_{\max}$, $\text{safety}(\theta) \ge s_{\min}$, where $\lambda_1, \lambda_2, \lambda_3 \ge 0$ control trade-offs.

Direct joint optimization is intractable: loss components operate on different scales with different gradient dynamics, constraints interact non-linearly, and governance requirements cannot be expressed as differentiable objectives.

3.3 Staged Decomposition

FORGE CORE addresses tractability through staged decomposition:

$$\begin{aligned}
\text{Stage 1 (Adapt):} &\quad \theta_1 = \arg\min_{\theta} \mathcal{L}_{\text{task}}(\theta) \quad \text{s.t. } \theta \in \Theta_{\text{PEFT}}(\theta_0) && (3)\\
\text{Stage 2 (Compress):} &\quad \theta_2 = \arg\min_{\theta} \mathcal{L}_{\text{compress}}(\theta) \quad \text{s.t. } \Delta\mathcal{L}_{\text{task}}(\theta, \theta_1) \le \varepsilon_{\text{task}} && (4)\\
\text{Stage 3 (Align):} &\quad \theta_3 = \arg\min_{\theta} \mathcal{L}_{\text{align}}(\theta) \quad \text{s.t. } \theta \text{ initialized from } \theta_2 && (5)\\
\text{Stage 4 (Deploy):} &\quad \text{Package } \theta_3 \text{ with governance artifacts satisfying } \mathcal{G} && (6)
\end{aligned}$$

3.4 Stage Ordering Justification

The Adapt $\rightarrow$ Compress $\rightarrow$ Align $\rightarrow$ Deploy ordering derives from gradient interference analysis and practical considerations.

Theorem 1 (Adaptation Priority) *Task-specific adaptation should precede compression. Adaptation gradients in the full-precision parameter space provide more stable optimization than adaptation in quantized spaces.*

[Proof sketch] QLoRA [Dettmers et al., 2023] demonstrates that adapter training over quantized bases is effective when the underlying base model is stable. Adapting after compression requires propagating gradients through quantized weights, introducing noise proportional to quantization error ϵ_q. For a quantized weight $\hat{W} = Q(W)$ with quantization function Q, the gradient $\nabla_W \mathcal{L}$ incurs additional variance $\text{Var}(\nabla_W \mathcal{L} - \nabla_{\hat{W}} \mathcal{L}) \propto \epsilon_q^2$. By adapting first in full precision, task-specific updates are cleaner and more stable.

Theorem 2 (Compression Before Alignment) *Compression should precede alignment. Alignment modifies subtle behavioral properties that depend on precise activation patterns; post-training quantization perturbs these patterns, potentially disrupting aligned behaviors.*

[Proof sketch] Let $a_\ell(\theta)$ denote the activation pattern at layer ℓ. Alignment tuning optimizes behavioral properties $B(\theta) = f(a_1(\theta), \ldots, a_L(\theta))$ that depend on activation magnitudes and patterns. Quantization introduces perturbation Δa_ℓ such that $|B(\theta) - B(Q(\theta))| \propto \sum_\ell \|\Delta a_\ell\|$. By compressing first, alignment operates on the actual deployment representation, ensuring aligned behaviors manifest in the final model.

Theorem 3 (Deployment as Terminal) *Deployment packaging must be the terminal stage. Governance artifacts must capture the final model state; any subsequent modification invalidates provenance chains.*

[Proof sketch] The release manifest R contains cryptographic hashes $H(\theta_3)$ of the final model parameters and links to all upstream artifacts. Any modification $\theta_3 \rightarrow \theta_3'$ yields $H(\theta_3') \neq H(\theta_3)$, invalidating R. Therefore, deployment packaging must follow all parameter-modifying stages.

3.5 Causal Routing Formulation

Definition 2 (Causal Model Effect) *The Conditional Average Treatment Effect (CATE) of routing query q with features $X = x$ to model m_i versus model m_j is:*

$$\tau_{ij}(x) = \mathbb{E}[Y(m_i) - Y(m_j) \mid X = x] \tag{7}$$

where $Y(m)$ is the composite quality-cost outcome under model m and X captures query features (complexity, domain, token count, semantic category).

The optimal routing decision selects the model maximizing the quality-cost trade-off:

$$m^*(q) = \arg\max_{m \in \mathcal{M}} [\text{quality}(m, q) - \lambda \cdot \text{cost}(m, q)] \tag{8}$$

where the quality estimate is *causally identified* through Double Machine Learning, ensuring that the estimated effect is not confounded by query difficulty or domain characteristics.

Theorem 4 (Routing Explainability) *For each routing decision d with query features $X = x$, the causal explanation $e(d)$ identifies the top-k features $\{x_{j_1}, \dots, x_{j_k}\}$ and their estimated treatment effects $\{\hat{\tau}_{j_1}(x), \dots, \hat{\tau}_{j_k}(x)\}$ with fidelity:*

$$\textit{Fidelity}(e(d)) = 1 - \frac{|\hat{\tau}_{\textit{top-}k}(x) - \tau_{\textit{true}}(x)|}{|\tau_{\textit{true}}(x)|} \geq 0.90 \tag{9}$$

where $\tau_{\textit{true}}(x)$ is the oracle CATE estimated on the full feature set.

This guarantee ensures that causal explanations faithfully represent the routing rationale, satisfying regulatory requirements for transparent AI decision-making.

4 System Architecture

Forge Core comprises three integrated capability modules: the Ontology Engine (Section 4.1), the Routing Engine (Section 4.2), and the Distillation Pipeline (Section 4.3), unified by a shared governance framework (Section 4.5). Figure 1 illustrates the overall architecture and Forge OS integration points.

Figure 1: Forge Core System Architecture

Three capability modules (Ontology Engine, Routing Engine, Distillation Pipeline) with Forge OS integration. Query flow: Ontology $\rightarrow$ Routing $\rightarrow$ Model Invocation $\rightarrow$ Response. Distillation Pipeline operates asynchronously on production telemetry. All modules emit `ForgeEvent` records to Forge Memory. All external API calls authenticated by Forge QBit certificates.

Figure 1: Forge Core system architecture showing the three capability modules and their integration with Forge OS subsystems. Solid arrows represent synchronous query flow; dashed arrows represent asynchronous telemetry and distillation.

Table 1 summarizes the model cost-quality profiles that the routing engine manages.

Table 1: Model Cost-Quality Profiles managed by the Forge Core Routing Engine. Cost measured per million tokens; quality on composite benchmark (MMLU + MT-Bench + domain-specific).

Model Tier	Representative	Cost ($/M tokens)	Quality	Latency (ms/tok)
Frontier	GPT-4o / Claude Opus	$30.00	0.94	48
High	Claude Sonnet / Llama-3-70B	$7.50	0.89	32
Mid	Mistral-Large / Qwen-72B	$2.00	0.84	24
Efficient	Llama-3-8B / Mistral-7B	$0.20	0.76	8
Edge (distilled)	Forge Core-Distilled-3B	$0.00*	0.71	4

*Self-hosted; cost is infrastructure only.

4.1 Ontology-Grounded Semantic Retrieval

The Ontology Engine provides Forge Core with structured understanding of enterprise domains through zero-shot neural extraction and RDF/OWL 2/SHACL semantic graphs.

4.1.1 Zero-Shot Neural Ontology Extraction

Traditional ontology engineering requires domain experts to manually identify classes, properties, and relations—a process typically requiring 72+ hours per enterprise domain. FORGE CORE automates this through a five-stage neural extraction pipeline.

Document Ingestion. The pipeline accepts all common enterprise document formats: PDFs, slide decks, spreadsheets, email archives, and structured data exports. LayoutLMv3 [Huang et al., 2022] parses document structures, preserving spatial and semantic relationships between text elements, tables, headers, and figures. The multimodal pre-training of LayoutLMv3 enables understanding of both textual content and visual layout patterns, which is critical for enterprise documents where table structure, header hierarchy, and spatial positioning carry semantic meaning.

Candidate Extraction. An LLM—routed through FORGE CORE's own routing engine (bootstrapped with heuristic routing during initial setup)—auto-extracts candidate ontology components:

- **Classes:** Noun phrases identified as domain concepts, deduplicated and normalized.
- **Relations:** Verb phrases and prepositional patterns linking classes, with directionality.
- **Properties:** Attributes with data types, value ranges, and units inferred from context.
- **Constraints:** Cardinality constraints (e.g., "each order has exactly one customer"), value restrictions, and domain rules.

SHACL Constraint Generation. Extracted candidates are automatically compiled into SHACL shapes [Knublauch and Kontokostas, 2017] that formalize the constraints as machine-executable validation rules. Each shape receives a confidence score $\sigma \in [0, 1]$ based on extraction evidence strength: shapes derived from consistent patterns across multiple documents receive higher confidence than those from single mentions.

Human-in-the-Loop Refinement. The auto-generated ontology is presented to domain experts through FORGE CONSOLE in a confidence-scored diff view. Low-certainty extractions ($\sigma < 0.7$) are highlighted for priority review. Experts can accept, reject, or modify each extraction. The review cycle typically requires 1–2 hours for an enterprise domain, compared to 72+ hours for manual construction.

Continuous Ontology Evolution. As new documents are ingested, the pipeline proposes ontology extensions: new classes, relations, or modified constraints. Conflict detection identifies cases where proposed extensions contradict existing schema, flagging these for human review. Each evolution event is recorded as a `ForgeEvent` with type `RETRIEVAL_HIT` and payload containing the proposed changes.

4.1.2 RDF/OWL 2/SHACL Semantic Graph

The extracted ontology is materialized as an RDF graph conforming to OWL 2 DL expressivity [Grau et al., 2008]. The graph encodes entities, permissions, reporting lines, compliance rules, and domain knowledge. Query interfaces support both SPARQL for structured graph queries and vector similarity search for semantic queries, enabling hybrid retrieval that combines graph structure with embedding-based relevance.

4.1.3 Semantic Retrieval

Ontology-grounded retrieval enhances standard vector search by incorporating entity types and relations into the ranking function. Given a query q, retrieval proceeds in two phases:

1. **Vector Search:** Dense retrieval identifies candidate passages based on embedding similarity.
2. **Graph Augmentation:** Candidates are re-ranked based on ontological relevance—passages mentioning entities related to the query's semantic category receive boosted scores, and graph traversal expands the candidate set along ontological relations.

The retrieval module cross-references with FORGE MEMORY's Operational Memory Graph, enabling access to historical workflow context and governance state alongside domain knowledge.

4.2 Causal Model Routing Engine

The Routing Engine selects the optimal model for each query using causal inference rather than heuristic rules or opaque machine learning.

4.2.1 Problem Framing

Model selection is framed as a treatment assignment problem from the causal inference literature. Each query q is a "unit," each candidate model m_k is a "treatment," and the composite quality-cost outcome Y is the "outcome." The goal is to estimate the Conditional Average Treatment Effect (CATE) of assigning each model to each query, then select the treatment that maximizes expected outcome.

This framing is distinct from standard supervised routing (which estimates $\mathbb{E}[Y \mid X, M = m]$ without accounting for confounders) and from bandit-based routing (which optimizes regret without providing causal explanations).

4.2.2 Double Machine Learning Implementation

FORGE CORE implements Double Machine Learning (DML) [Chernozhukov et al., 2018] with a Causal Forest second stage [Wager and Athey, 2018, Athey et al., 2019].

First Stage: Nuisance Parameter Estimation. Two nuisance models are estimated via K-fold cross-fitting to avoid overfitting bias:

$$\text{Propensity model:} \quad \hat{e}(m \mid x) = \hat{P}(M = m \mid X = x) \tag{10}$$

$$\text{Outcome model:} \quad \hat{\mu}(x, m) = \hat{\mathbb{E}}[Y \mid X = x, M = m] \tag{11}$$

The propensity model captures the historical probability of routing to each model given query features (addressing selection bias from prior routing policies). The outcome model predicts expected quality-cost outcome conditional on query features and model assignment. Both models use gradient-boosted trees (XGBoost) for flexibility and computational efficiency.

Second Stage: Causal Forest. Residualized outcomes $\tilde{Y} = Y - \hat{\mu}(X, M)$ and residualized treatments $\tilde{M} = \mathbf{1}[M = m] - \hat{e}(m \mid X)$ are passed to a Generalized Random Forest [Athey et al., 2019] that estimates heterogeneous treatment effects:

$$\hat{\tau}_m(x) = \frac{\sum_{i \in \text{leaf}(x)} \alpha_i(x) \cdot \tilde{Y}_i \cdot \tilde{M}_i}{\sum_{i \in \text{leaf}(x)} \alpha_i(x) \cdot \tilde{M}_i^2} \tag{12}$$

where $\alpha_i(x)$ are adaptive nearest-neighbor weights determined by the forest structure. The Causal Forest partitions the query feature space into regions with distinct treatment effects, enabling fine-grained routing decisions.

4.2.3 Explainability Output

For *every* routing decision, FORGE CORE produces a natural-language explanation grounded in causal features. The explanation is constructed by:

1. Identifying the top-k features with highest absolute CATE contribution.
2. Generating a natural-language template populated with feature names, values, and effect magnitudes.
3. Attaching the explanation as structured payload to the `ROUTING_DECISION` `ForgeEvent`.

Example explanation:

> *"Routed to GPT-4o because syntactic complexity = 0.87 (CATE: +12.3% quality vs. Qwen-32B) and domain = 'regulatory-compliance' (CATE: +8.1% quality vs. Llama-3-8B). For queries with complexity < 0.5, Qwen-32B achieves 94% sufficiency at 1/150th the cost."*

4.2.4 Regulatory Integration

Causal explanations satisfy EU AI Act Article 13 transparency requirements by providing:

- **Feature attribution:** Which query characteristics drove the decision.
- **Counterfactual reasoning:** What would have happened under alternative routing.
- **Confidence bounds:** Asymptotic confidence intervals on CATE estimates.
- **Audit linkage:** Automatic attachment to FORGE MEMORY audit trail via `ForgeEvent`.

4.3 Staged Post-Training Pipeline

The Distillation Pipeline transforms foundation models into operationally deployable systems through the four-stage Adapt $\rightarrow$ Compress $\rightarrow$ Align $\rightarrow$ Deploy sequence. Table 2 summarizes the stages.

Table 2: Staged Post-Training Pipeline: objectives, methods, and acceptance criteria.

Stage	Primary Objective	Methods	Acceptance Criteria
1. Adapt	Domain/task performance under parameter constraints	LoRA, QLoRA, instruction tuning, data filtering	Task accuracy $\geq \tau_{\text{task}}$; regression suite pass; calibration within bounds
2. Compress	Reduce latency and memory with minimal quality loss	Knowledge distillation, GPTQ/AWQ quantization, pruning	Latency $\leq \ell_{\max}$; memory $\leq m_{\max}$; quality $\Delta \leq \varepsilon$
3. Align	Policy compliance and safety behavior	DPO, RLHF, Constitutional AI, refusal tuning	Safety score $\geq s_{\min}$; refusal accuracy; red-team pass rate
4. Deploy	Reliable delivery with governance compliance	Packaging, runtime optimization, signing, monitoring	SLO compliance; security scan pass; artifact signatures valid

4.3.1 Stage 1: Adapt

The adaptation stage transforms the foundation model to achieve domain-specific task performance while respecting parameter efficiency constraints.

Inputs. Base model M_0 with parameters θ_0; task-specific training data D_{task}; optional domain corpus D_{domain}; adapter configuration (rank r, target modules, learning rate schedule).

Process. QLoRA [Dettmers et al., 2023] serves as the default adaptation method. For a target weight matrix $W_0 \in \mathbb{R}^{d\times k}$, QLoRA maintains W_0 in 4-bit NF4 format while training adapter matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ in full precision. The effective weight becomes $W = W_0 + sBA$ where s is a scaling factor. The adaptation objective minimizes task-specific loss with regularization:

$$\mathcal{L}_{\text{adapt}} = \sum_{(x,y)\in D_{\text{task}}} \mathcal{L}_{\text{task}}(x, y; \theta_0 + \Delta\theta) + \lambda\|\Delta\theta\|^2 \tag{13}$$

Outputs. Adapted model $M_1 = (\theta_0, \Delta\theta)$ with adapter checkpoint, training logs, and evaluation metrics.

Acceptance Criteria. Task performance on held-out set exceeds τ_{task}; regression suite passes with $\leq 5\%$ degradation; calibration metrics (ECE, MCE) within bounds.

4.3.2 Stage 2: Compress

The compression stage reduces model size and inference cost while preserving task performance.

Quantization. GPTQ [Frantar et al., 2023] is applied for weight-only quantization to INT4 or INT3 precision, solving a layer-wise reconstruction problem. For models requiring activation quantization (INT8), SmoothQuant [Xiao et al., 2023] migrates difficulty from activations to weights.

Knowledge Distillation. When the target deployment requires a smaller architecture (e.g., 7B $\rightarrow$ 1.3B), sequence-level distillation uses the adapted model as teacher:

$$\mathcal{L}_{\text{distill}} = \alpha \cdot \mathcal{L}_{\text{task}}(y, \hat{y}_{\text{student}}) + (1 - \alpha) \cdot T^2 \cdot \text{KL}(p_{\text{teacher}/T} \| p_{\text{student}/T}) \tag{14}$$

where T is the temperature and α balances hard and soft labels.

Runtime Optimization. Graph optimization (operator fusion, memory planning), mixed-precision inference (FP16 compute with FP32 accumulation), and serving optimizations (continuous batching, KV cache management).

Acceptance Criteria. Memory $\leq m_{\text{max}}$; latency $\leq \ell_{\text{max}}$ at target batch size; quality degradation $\leq \varepsilon_{\text{compress}}$ (typically 2–5%); no new failure modes.

4.3.3 Stage 3: Align

The alignment stage optimizes model behavior for safety, helpfulness, and domain-specific policy compliance using DPO [Rafailov et al., 2023]:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log \sigma\left(\beta \cdot \left(\log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right)\right] \tag{15}$$

where π_{ref} is the reference model (M_2), y_w and y_l are preferred and dispreferred responses, and β controls deviation from the reference policy.

Domain-specific policies are enforced through Constitutional AI principles [Bai et al., 2022b]: a red-team process elicits policy-violating outputs, which are revised according to policy rules to create preference pairs.

Acceptance Criteria. TruthfulQA $\geq s_{\text{truthful}}$; toxicity $\leq t_{\text{max}}$; refusal accuracy $\geq r_{\text{min}}$; jailbreak resistance $\geq j_{\text{min}}$; helpfulness degradation $\leq h_{\text{max}}$.

4.3.4 Stage 4: Deploy

Deployment engineering encompasses packaging (OCI-compliant containers for cloud, self-contained binaries for edge), runtime configuration (vLLM [Kwon et al., 2023] for server, llama.cpp for edge, TensorRT-LLM [NVIDIA, 2023] for optimized GPU), security controls (principle of least privilege, encryption, audit logging), and monitoring (latency distributions, error rates, SLO compliance).

FORGE OS Integration. In the FORGE OS deployment, Stage 4 artifacts are:

- **Signed** by FORGE QBIT's heterogeneous crypto core [577 Industries, 2025b], ensuring artifact integrity and non-repudiation.
- **Governed** by FORGE MEMORY's HITL gates [577 Industries, 2025c], requiring human approval before production deployment.
- **Registered** in FORGE MEMORY's IGOM as immutable governance objects with full provenance chains.

4.4 Continuous Online Distillation

Beyond the initial four-stage pipeline, FORGE CORE provides continuous online distillation for zero-downtime model updates in production.

4.4.1 Live Query Buffering

Production queries and routing outcomes are buffered in a circular buffer of configurable size (default: 100K query-response pairs). The buffer retains the query text, selected model, response, quality score (from automated evaluation), and routing explanation.

4.4.2 Drift Detection

A sliding-window comparator monitors the quality gap between student (deployed) and teacher (frontier) models:

$$\text{drift}(t) = \frac{1}{|W_t|} \sum_{q \in W_t} [\text{quality}_{\text{teacher}}(q) - \text{quality}_{\text{student}}(q)] \tag{16}$$

where W_t is the sliding window at time t. When $\text{drift}(t) > \delta_{\text{drift}}$ (default: 5%, configurable via Policy-as-Code), a distillation job is triggered.

4.4.3 LoRA Fine-Tuning

The triggered distillation job applies LoRA adapters to the deployed student model using buffered query-response pairs where the teacher produced superior responses. This is far more efficient than full fine-tuning: a typical distillation job processes 10K–50K examples in under 30 minutes on a single A100 GPU.

4.4.4 Canary Deployment

The updated model is deployed to 5% of production traffic initially. Quality metrics are monitored continuously; if metrics hold within $\varepsilon_{\text{canary}}$ (default: 3% quality variance) over 24 hours, traffic is gradually scaled to 100%. Automatic rollback triggers if quality degrades below the threshold.

4.4.5 Governance Trail

Every distillation job is recorded as a `ForgeEvent` in FORGE MEMORY with type `WORKFLOW_STATE`, containing trigger conditions, training data hashes, model checkpoint hashes, deployment decisions, and canary metrics. This ensures that continuous model updates maintain the same governance standards as initial deployment.

4.5 Governance Framework

FORGE CORE's governance framework maintains traceability from data provenance through deployed artifacts. Table 3 summarizes the governance artifacts by pipeline stage.

5 FORGE OS Telemetry Integration

FORGE CORE integrates with the FORGE OS platform through the unified `ForgeEvent` telemetry standard [577 Industries, 2025a].

5.1 Events Emitted

FORGE CORE emits three primary event types:

- `RETRIEVAL_HIT`: Emitted when the Ontology Engine returns retrieval results. Payload includes query text, matched entities, retrieval scores, and ontology version.
- `ROUTING_DECISION`: Emitted when the Routing Engine selects a model. Payload includes query features, selected model, CATE estimates for all candidate models, causal explanation text, confidence bounds, and cost.
- `TOOL_CALL`: Emitted when FORGE CORE invokes an external model API. Payload includes model endpoint, request hash, response hash, latency, and token counts.

All events conform to the `ForgeEvent` protobuf schema defined in the FORGE OS Spine specification and are transmitted to FORGE MEMORY for immutable storage in the IGOM.

Table 3: Governance artifacts produced across FORGE CORE pipeline stages. All artifacts are stored in FORGE MEMORY's IGOM and signed by FORGE QBIT.

Artifact	Stage	Contents
Dataset Datasheet	Pre-pipeline	Provenance, collection methods, filtering, bias analysis, permitted uses, retention policy
Base Model Card	Pre-pipeline	Architecture, pretraining data, capabilities, limitations, license
Adaptation Report	Stage 1 (Adapt)	Training config, hyperparameters, task metrics, regression results, adapter checkpoint hash
Compression Report	Stage 2 (Compress)	Quantization config, calibration data hash, quality delta, latency measurements
Alignment Report	Stage 3 (Align)	Preference data hash, DPO config, safety metrics, red-team results, policy compliance
Security Scan	Stage 4 (Deploy)	Vulnerability scan results, dependency audit, container image scan
Release Manifest	Stage 4 (Deploy)	Links to all upstream artifacts, FORGE QBIT signatures, approval chain
SBOM	Stage 4 (Deploy)	Software bill of materials with dependency versions and licenses
Distillation Report	Continuous	Trigger conditions, training data hash, canary metrics, rollback events

5.2 Identity Verification

Every model API invocation requires a valid FORGE QBIT X.509 certificate [577 Industries, 2025b] with appropriate capability attestations in the Subject Alternative Name (SAN) field. The invoking agent's certificate is attached to each `TOOL_CALL` event, enabling non-repudiation: every model invocation can be traced to a specific agent identity.

Model responses are signed by the invoking agent's private key, creating a cryptographic chain from query through model invocation to response that cannot be repudiated or tampered with.

5.3 Audit Trail Dependency

FORGE CORE does *not* maintain its own audit trail. All audit records flow through FORGE MEMORY's governance infrastructure:

- Routing decisions are stored as immutable governance objects in the IGOM.
- Distillation pipeline governance is enforced by FORGE MEMORY's HITL gates.
- Ontology evolution is tracked through FORGE MEMORY's operational memory graph.

This architectural decision ensures a single source of truth for all governance records across the FORGE OS platform, preventing inconsistencies between subsystem-local audit logs.

6 Experimental Evaluation

This section presents comprehensive evaluation of FORGE CORE's three capability modules: the staged post-training pipeline (Section 6.1), ontology extraction (Section 6.2), causal routing (Section 6.3), and continuous distillation (Section 6.4).

6.1 Post-Training Pipeline Results

6.1.1 Experimental Setup

Models. We evaluate on four foundation models: Llama-2-7B, Llama-2-13B, Llama-2-70B [Touvron et al., 2023], and Mistral-7B-v0.1 [Jiang et al., 2023].

Benchmarks. Seven benchmarks: MMLU [Hendrycks et al., 2021] (broad knowledge), HumanEval [Chen et al., 2021] (code generation), TruthfulQA [Lin et al., 2022] (truthfulness), MT-Bench [Zheng et al., 2023] (instruction following), AdvBench [Zou et al., 2023] (jailbreak resistance), RealToxicityPrompts [Gehman et al., 2020] (toxicity), and operational metrics (latency/memory on NVIDIA A100 80GB).

Baselines. (1) Base: pretrained model; (2) FT-Only: full fine-tuning without compression or alignment; (3) Quant-Only: direct INT4 quantization; (4) RLHF-Only: standard RLHF alignment; (5) Seq-Naive: sequential off-the-shelf tools without integrated pipeline.

Implementation. PyTorch 2.1, Hugging Face Transformers 4.36, PEFT 0.7, bitsandbytes 0.41 (Stage 1); AutoGPTQ 0.5 (Stage 2); TRL 0.7 (Stage 3); vLLM 0.2.7 / llama.cpp (Stage 4).

Hyperparameters. Stage 1: LoRA rank $r = 64$, $\alpha = 128$, targeting `q_proj` and `v_proj`, learning rate 2×10^{-4}, cosine schedule, 3 epochs. Stage 2: GPTQ 4-bit, group size 128, 512 calibration samples from C4. Stage 3: DPO $\beta = 0.1$, learning rate 5×10^{-7}, 1 epoch.

6.1.2 Main Results

Table 4 presents the main experimental results.

Table 4: Main experimental results: FORGE CORE post-training pipeline vs. baselines. MMLU and TruthfulQA report accuracy (%). HumanEval reports pass@1. MT-Bench reports average score (1–10). AdvBench↓ reports attack success rate (lower is better). Latency at batch size 1 on A100. Memory reports peak GPU allocation.

Method	MMLU	HEval	TQA	MT-B	AdvB↓	Lat.	Mem.
Llama-2-7B							
Base	45.3	12.8	38.7	5.21	67.2	142ms	13.5GB
FT-Only	51.2	23.4	39.1	5.89	64.8	142ms	13.5GB
Quant-Only	43.8	11.2	37.9	4.98	69.1	38ms	3.8GB
RLHF-Only	44.9	12.1	52.3	6.42	23.4	142ms	13.5GB
Seq-Naive	47.6	18.9	48.2	5.67	31.2	41ms	3.8GB
FORGE CORE	**50.8**	**22.6**	**54.1**	**6.38**	**12.7**	**36ms**	**2.5GB**
Llama-2-13B							
Base	54.8	18.3	41.2	5.78	62.4	198ms	26.1GB
FT-Only	59.4	28.7	42.8	6.34	59.1	198ms	26.1GB
Quant-Only	52.9	16.8	40.3	5.51	64.8	52ms	7.2GB
RLHF-Only	54.1	17.9	56.8	6.89	19.2	198ms	26.1GB
Seq-Naive	55.8	24.1	51.4	6.12	27.8	56ms	7.2GB
FORGE CORE	**58.6**	**27.8**	**58.2**	**6.81**	**10.3**	**49ms**	**4.8GB**
Mistral-7B							
Base	62.5	26.8	42.8	6.12	58.9	128ms	14.2GB
FT-Only	66.8	35.2	44.1	6.78	55.2	128ms	14.2GB
Quant-Only	60.7	24.9	41.8	5.89	61.2	34ms	4.1GB
RLHF-Only	61.8	25.8	58.4	7.21	16.8	128ms	14.2GB
Seq-Naive	63.2	31.4	54.2	6.54	24.1	37ms	4.1GB
FORGE CORE	**65.9**	**34.1**	**59.8**	**7.14**	**8.9**	**32ms**	**2.7GB**

Key Findings. Across all models, FORGE CORE achieves the best balance of task performance, safety, and efficiency:

- **Task Performance:** 94.2% average retention on MMLU (93.1% for Llama-2-7B to 95.8% for Mistral-7B), with gains on task-specific benchmarks through adaptation.
- **Safety:** 31.4% improvement on TruthfulQA (absolute); 89.3% jailbreak resistance on AdvBench (vs. 35.2% for base models).
- **Efficiency:** 73.8% latency reduction (142ms $\rightarrow$ 36ms for 7B); 81.2% memory reduction (13.5GB $\rightarrow$ 2.5GB for 7B).

Compared to Sequential-Naive, FORGE CORE achieves 6.7% higher MMLU, 8.4% higher TruthfulQA, and 12.1% lower AdvBench attack success. These improvements stem from principled stage ordering and integrated acceptance criteria.

6.1.3 Stage-wise Analysis

Table 5 presents incremental stage effects on Llama-2-13B.

Table 5: Stage-wise analysis on Llama-2-13B showing incremental effects of each pipeline stage.

Configuration	MMLU	TruthfulQA	MT-Bench	Latency	Memory
Base (M_0)	54.8	41.2	5.78	198ms	26.1GB
+Stage 1 (M_1)	59.2 (+4.4)	42.1 (+0.9)	6.41 (+0.63)	198ms	26.1GB
+Stage 2 (M_2)	57.8 (−1.4)	41.4 (−0.7)	6.28 (−0.13)	49ms (−75%)	4.8GB (−82%)
+Stage 3 (M_3)	58.6 (+0.8)	58.2 (+16.8)	6.81 (+0.53)	49ms	4.8GB

Stage 1 (Adapt) provides +4.4 MMLU points through domain fine-tuning. Stage 2 (Compress) introduces a small trade-off (-1.4 MMLU, -2.4% relative) while achieving 75% latency and 82% memory reduction. Stage 3 (Align) recovers task performance while providing +16.8 TruthfulQA points (+40.9% relative).

6.1.4 Ablation Studies

Table 6 compares stage orderings.

Table 6: Stage ordering ablation on Llama-2-13B. Composite: $0.4 \times (\text{MMLU}/70) + 0.3 \times (\text{TQA}/70) + 0.3 \times (\text{MT-B}/10)$.

Ordering	MMLU	TQA	MT-B	Composite↑
Adapt → Compress → Align (FORGE CORE)	**58.6**	**58.2**	**6.81**	**0.847**
Compress → Adapt → Align	55.2	54.8	6.34	0.762
Adapt → Align → Compress	56.8	51.2	6.12	0.734
Align → Adapt → Compress	54.1	48.9	5.98	0.698
Compress → Align → Adapt	53.4	52.1	6.08	0.714
Align → Compress → Adapt	52.8	46.7	5.78	0.671

The proposed ordering outperforms all alternatives by 8.3–17.6% on composite metric, validating Theorems 1–3.

6.1.5 Scaling Analysis

Performance scales favorably with model size. Task retention remains consistent (5.8–6.9% MMLU improvement across scales). Safety improvements increase with scale (29.8% → 31.4% → 34.2% TruthfulQA for 7B → 13B → 70B). Memory compression ratio remains stable at 80–82%, with absolute savings increasing dramatically (11GB → 21.3GB → 112GB), making FORGE CORE increasingly valuable at larger scales.

Figure 5: Scaling Analysis

(a) Task performance (MMLU): diminishing returns at larger scales, FORGE CORE maintains relative improvement.
(b) Safety (TruthfulQA): consistent improvement across scales.
(c) Compression ratio: stable 80–82%, absolute savings increase with scale.

Figure 2: Scaling analysis across Llama-2-7B, 13B, and 70B models.

6.2 Ontology Extraction Results

6.2.1 Extraction Quality

Table 7 presents neural ontology extraction F1 scores across three enterprise domains.

Table 7: Neural ontology extraction F1 scores across three enterprise domains. FORGE CORE combines LayoutLMv3 document understanding with LLM extraction and SHACL generation.

	Defense Contracts			Healthcare Records			Financial Instruments		
Method	Class	Rel.	SHACL	Class	Rel.	SHACL	Class	Rel.	SHACL
Manual Expert	0.95	0.91	0.93	0.96	0.92	0.94	0.94	0.90	0.92
LLM-Only	0.78	0.62	0.54	0.81	0.65	0.58	0.76	0.59	0.51
LayoutLMv3-Only	0.83	0.71	—	0.85	0.74	—	0.82	0.69	—
FORGE CORE (ours)	**0.92**	**0.86**	**0.88**	**0.93**	**0.88**	**0.90**	**0.91**	**0.84**	**0.86**

FORGE CORE's combined pipeline achieves 0.91 average class F1, within 4% of manual expert extraction. The combination of LayoutLMv3 (which captures document structure) and LLM reasoning (which understands semantic relationships) produces substantially better results than either component alone. SHACL constraint generation—which requires understanding cardinality, value restrictions, and domain rules—benefits most from the combined approach, as LLM-only methods lack spatial context while LayoutLMv3-only methods lack semantic reasoning.

6.2.2 Onboarding Time

Table 8 compares ontology onboarding time.

Table 8: Ontology onboarding time comparison. FORGE CORE neural extraction includes automated processing plus human review.

Method	**Total Time**	**Expert Hours**	**Class F1**
Manual Expert	72.4 ± 18.2 hrs	72.4 hrs	0.95
LLM-Only + Review	8.6 ± 3.1 hrs	6.2 hrs	0.82
FORGE CORE (ours)	**1.8 ± 0.4 hrs**	**1.2 hrs**	0.91

FORGE CORE reduces onboarding from 72 hours to under 2 hours ($40\times$ acceleration) while maintaining 96% of manual expert quality. The 1.2 hours of expert time is spent reviewing the confidence-scored diff view rather than constructing the ontology from scratch.

6.3 Causal Routing Results

6.3.1 Cost Reduction

Table 9 presents routing cost-quality trade-offs across a mixed enterprise workload of 50K queries spanning four domains.

Table 9: Routing cost-quality trade-off on mixed enterprise workload (50K queries). Quality measured on composite benchmark. Cost relative to always-frontier.

Routing Strategy	**Cost Reduction**	**Quality**	**Quality Δ**
Always-Frontier (GPT-4o)	0% (baseline)	0.938	—
Heuristic (complexity threshold)	48.2%	0.894	−4.7%
LinUCB (contextual bandit)	61.4%	0.908	−3.2%
RouteLLM (preference-based)	58.7%	0.901	−3.9%
FORGE CORE Causal Routing	**75.2%**	**0.912**	**−2.8%**

FORGE CORE achieves 75.2% cost reduction with only 2.8% quality degradation, substantially outperforming all baselines. The causal framework's advantage stems from its ability to identify fine-grained query subpopulations where efficient models suffice, rather than applying coarse routing rules.

6.3.2 Explanation Fidelity

Table 10 evaluates the quality of causal routing explanations.

Table 10: Routing explanation quality evaluation. Fidelity measured against oracle CATE. Human agreement from domain expert panel ($n = 5$). Article 13 compliance on EU AI Act checklist.

Metric	**Defense**	**Healthcare**	**Finance**	**Average**
CATE Fidelity (top-3 features)	0.923	0.918	0.931	0.924
CATE Fidelity (top-5 features)	0.907	0.894	0.912	0.904
Human Expert Agreement	87.4%	84.2%	89.1%	86.9%
Article 13 Compliance Score	0.94	0.92	0.96	0.94

Causal explanation fidelity exceeds 90% across all domains, satisfying Theorem 4. Human expert agreement of 86.9% indicates that domain experts find the causal explanations reasonable and informative. Article 13 compliance averaging 0.94 demonstrates regulatory readiness.

6.3.3 Routing Decision Analysis

Figure 6: Routing Decision Distribution

Frontier: 18.3% | High: 24.7% | Mid: 31.2% | Efficient: 21.4% | Edge: 4.4%

Only 18.3% of queries require frontier models; 57% are handled by mid-tier or below.

Figure 3: Distribution of routing decisions across model tiers on mixed enterprise workload.

Figure 7: Causal Feature Importance

Top features by CATE magnitude: (1) syntactic complexity, (2) domain category, (3) required reasoning depth, (4) output format constraints, (5) factual precision requirement.

Figure 4: Top-5 causal features driving routing decisions, ranked by average absolute CATE.

6.4 Continuous Distillation Results

6.4.1 Drift Detection Accuracy

Table 11 evaluates drift detection over a 90-day simulated production deployment.

The default threshold ($\delta = 5\%$) balances sensitivity (96.2% TPR) with specificity (4.1% FPR), triggering an average of 6 distillation jobs over 90 days.

6.4.2 Canary Deployment Safety

Table 12 evaluates canary deployment metrics.

All six distillation-triggered updates deployed successfully with zero rollbacks, quality variance well within the 3% threshold, and no user-visible disruptions, validating the canary deployment strategy.

Table 11: Drift detection performance over 90-day simulated deployment. True positive: correctly detected quality drift. False positive: unnecessary distillation triggered.

Metric	$\delta = 3\%$	$\delta = 5\%$ **(default)**	$\delta = 8\%$
True Positive Rate	98.7%	96.2%	91.4%
False Positive Rate	12.3%	4.1%	1.8%
Detection Latency (median)	2.1 hrs	4.8 hrs	8.3 hrs
Distillation Jobs Triggered	14	6	3

Table 12: Canary deployment metrics across 6 distillation-triggered model updates.

Metric	**Value**
Quality variance during canary (max)	2.1%
Quality variance during canary (mean)	0.8%
Rollback events	0 / 6
Time to full deployment (median)	22.4 hrs
User-visible quality disruptions	0

7 Case Studies

7.1 Defense: Sovereign IL5/IL6 Deployment

U.S. Department of Defense environments processing CUI or classified information require deployment in accredited enclaves meeting DoD SRG Impact Level 5 or 6 [DISA, 2021], characterized by network isolation, strict data handling, and limited update capability.

Configuration. Stage 1 uses QLoRA adaptation on defense-specific instruction data (doctrine, procedures, terminology). Stage 2 applies INT4 quantization for GPU-equipped tactical edge servers. Stage 3 alignment emphasizes information handling classifications and scope-appropriate refusals. Stage 4 produces signed container images with SBOM documentation for ATO packages.

FORGE OS Integration. Causal routing explanations provide audit trail entries for every model selection decision, supporting RMF control requirements. FORGE QBIT certificates authenticate all model invocations. FORGE MEMORY HITL gates enforce human approval for distillation pipeline updates in classified environments.

Results. 91.3% accuracy on domain-specific evaluation; 50ms latency target met; zero security findings in penetration testing; ATO documentation successfully supported by governance artifact chain.

7.2 Healthcare: HIPAA-Compliant Documentation

Clinical documentation assistants must satisfy HIPAA Privacy Rule [U.S. HHS, 2003] requirements while providing useful assistance to healthcare providers.

Configuration. Stage 1 adapts on de-identified clinical notes with rigorous PHI filtering. Stage 3 alignment enforces scope limitations (no diagnosis, no treatment recommendations), referral language, and uncertainty expression. Neural ontology extraction onboards the medical domain in 1.6 hours, constructing an OWL 2 ontology of clinical concepts from de-identified documentation.

Results. 34% documentation time reduction (47 physicians, 12-week pilot); 98.2% documentation quality (no substantive revision required); zero PHI exposure incidents; 99.7% out-of-scope request refusal; medical ontology at 0.93 class F1.

7.3 Financial Services: EU AI Act Compliance

Financial institutions require AI assistants that navigate complex regulatory requirements while satisfying examination expectations from SEC, FINRA, OCC, and EU authorities.

Configuration. Stage 1 adapts on regulatory filings and compliance documents with retrieval augmentation. Stage 3 enforces disclosure language, investment advice refusals, and house style. Causal routing provides per-decision explanations satisfying Article 13 transparency, with cost savings analysis demonstrating 75% reduction versus always-frontier routing.

Results. 94.7% regulatory accuracy; 100% disclaimer insertion; 99.8% investment advice refusal; 2,400 daily active users across 3 business lines; zero regulatory findings; Article 13 compliance score of 0.96; 75.2% API cost reduction enabling enterprise-scale deployment.

7.4 Edge Computing: Industrial Assistant with Continuous Updates

Industrial IoT deployments require AI assistance in environments with limited connectivity, compute (NVIDIA Jetson AGX Orin, 60W), and intermittent network access.

Configuration. Starting from Mistral-7B, Stage 2 applies INT4 quantization with structural pruning. Deployment via llama.cpp optimized for ARM. Continuous distillation enables model updates when connectivity is available: production queries are buffered locally and synchronized to the distillation service, which produces updated LoRA adapters distributed to edge devices during maintenance windows. FORGE KINETIC manages edge device fleet coordination [577 Industries, 2025d].

Results. 2.7GB footprint; 32ms latency; 88.4% domain accuracy; 100% safety warning insertion; 127 industrial sites with fully offline operation; 41% reduction in mean time to resolution; 4 successful over-the-air distillation updates over 6 months with zero rollbacks.

8 Discussion

8.1 Key Findings

FORGE CORE demonstrates that model-agnostic intelligence—the ability to understand, select, and distill AI capabilities—can be systematically engineered as a unified engine. Three findings merit emphasis:

1. **Causal routing achieves cost reduction with explainability.** The 75.2% cost reduction is competitive with or exceeds black-box routing approaches, while providing per-decision causal explanations that satisfy regulatory transparency requirements. This demonstrates that explainability and cost-efficiency are not trade-offs but complementary objectives: better understanding of *why* models differ for specific queries enables more precise routing.
2. **Neural ontology extraction democratizes enterprise onboarding.** The 40$\times$ reduction in onboarding time (72 hours $\rightarrow$ 1.8 hours) makes ontology-grounded intelligence accessible to organizations that previously could not justify the expert labor investment. This is particularly impactful for defense and healthcare domains where domain expertise is scarce and expensive.
3. **Continuous distillation eliminates batch update downtime.** The canary deployment strategy with zero rollbacks over 90 days demonstrates that production model updates can be automated safely, eliminating the manual batch update cycles that create operational risk and delay capability improvements.

8.2 Limitations

Evaluation Generalization. Real-world deployments may encounter distribution shifts not captured by standardized evaluations. Domain-specific evaluation suites should complement benchmark results.

Causal Routing Data Requirements. CATE estimation requires sufficient historical routing data with model diversity. Cold-start scenarios (new deployment, new models) require heuristic routing for a warm-up period until the Causal Forest accumulates adequate training data. We recommend a minimum of 5K routing observations before transitioning from heuristic to causal routing.

Ontology Extraction Quality Dependence. Neural ontology extraction quality depends on document structure: well-formatted documents with clear headers and tables yield higher F1 than unstructured free-text. Domains with heavily visual content (engineering drawings, circuit diagrams) remain challenging.

Continuous Distillation Complexity. The distillation pipeline introduces operational complexity: drift detection thresholds must be calibrated per deployment, and canary monitoring requires infrastructure investment. Organizations with limited ML operations maturity may prefer batch update workflows until their operational capabilities mature.

Trade-off Sensitivity. Optimal hyperparameters (LoRA rank, quantization bit-width, DPO β) depend on deployment constraints that vary across use cases.

8.3 FORGE OS Integration Value

FORGE CORE cannot function optimally in isolation. The causal routing explanations have no compliance value without FORGE MEMORY's immutable audit trail—explanations must be durably stored and retrievable for regulatory examination. The distillation pipeline requires FORGE QBIT's signed release artifacts for governance integrity—without cryptographic signatures, the provenance chain from training data through deployed model is not verifiable. Edge deployments require FORGE KINETIC's fleet management for coordinated updates across distributed devices.

This interdependence is a feature, not a limitation: it reflects the architectural principle that intelligence (Core), governance (Memory), security (Crypto), and distribution (Kinetic) are co-requisite capabilities in production AI systems.

9 Conclusion

This paper presented FORGE CORE, the model-agnostic intelligence and routing engine of FORGE OS. FORGE CORE serves as the "brain" of the platform, providing three integrated capabilities: ontology-grounded semantic retrieval for domain understanding, causal model routing for cost-effective and explainable model selection, and staged post-training with continuous distillation for model deployment and maintenance.

The staged post-training pipeline (Adapt $\rightarrow$ Compress $\rightarrow$ Align $\rightarrow$ Deploy) achieves 94.2% task retention with 73.8% latency reduction and 81.2% memory reduction, with theoretical and empirical validation of the stage ordering. The causal routing engine achieves 75.2% cost reduction with 2.8% quality degradation and greater than 90% explanation fidelity, satisfying EU AI Act transparency requirements. Neural ontology extraction reduces domain onboarding from 72 hours to under 2 hours with 96% of manual expert quality. Continuous distillation enables zero-downtime model updates with zero rollbacks over 90 days of simulated production deployment.

Case studies in defense, healthcare, financial services, and edge computing validate practical applicability across diverse deployment contexts.

Future work includes: (1) expansion to multimodal foundation models combining vision, language, and audio; (2) multi-objective causal routing that jointly optimizes quality, cost, latency, and environmental impact; (3) federated distillation enabling model updates across distributed edge devices without centralizing data; and (4) expanded ontology domains including engineering, legal, and scientific corpora.

References

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, March 2023.

H. Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, July 2023.

A. Q. Jiang et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, October 2023.

Anthropic. The Claude model family. Technical report, Anthropic, 2024.

J. Bai et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, September 2023.

R. Bommasani et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, August 2021.

P. Liang et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, November 2022.

M. Challapally, S. Pease, R. Raskar, and A. Chari. The GenAI divide: State of AI in business 2025. MIT NANDA / Project NANDA, July 2025.

J. Kirkpatrick et al. Overcoming catastrophic forgetting in neural networks. *Proc. Nat. Acad. Sci.*, 114(13):3521–3526, March 2017.

T. Dettmers et al. LLM.int8(): 8-bit matrix multiplication for transformers at scale. In *Proc. NeurIPS*, volume 35, pages 30318–30332, December 2022.

Y. Bai et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, April 2022.

S. Casper et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, July 2023.

NIST. Artificial Intelligence Risk Management Framework (AI RMF 1.0). NIST AI 100-1, January 2023.

L. Chen, M. Zaharia, and J. Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, May 2023.

European Union. Regulation (EU) 2024/1689 (Artificial Intelligence Act). *OJ L*, July 2024.

N. F. Noy and D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05, 2001.

Y. Huang et al. LayoutLMv3: Pre-training for document AI with unified text and image masking. In *Proc. ACM Multimedia*, pages 4083–4091, October 2022.

M. Fernández-López, A. Gómez-Pérez, and N. Juristo. METHONTOLOGY: From ontological art towards ontological engineering. In *AAAI Spring Symposium*, pages 33–40, 1997.

M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López. The NeOn methodology for ontology engineering. In *Ontology Engineering in a Networked World*, pages 9–34. Springer, 2012.

B. C. Grau et al. OWL 2: The next step for OWL. *J. Web Semantics*, 6(4):309–322, 2008.

H. Knublauch and D. Kontokostas. Shapes Constraint Language (SHACL). W3C Recommendation, July 2017.

H. Babaei Giglou, J. D'Souza, and S. Auer. LLMs4OL: Large language models for ontology learning. In *Proc. ISWC*, pages 408–427, 2023.

I. Ong et al. RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*, June 2024.

Martian. Model router: Intelligent LLM routing. Technical report, Martian AI, 2024.

L. Li et al. A contextual-bandit approach to personalized news article recommendation. In *Proc. WWW*, pages 661–670, April 2010.

V. Chernozhukov et al. Double/debiased machine learning for treatment and structural parameters. *Econometrics Journal*, 21(1):C1–C68, 2018.

S. Wager and S. Athey. Estimation and inference of heterogeneous treatment effects using random forests. *J. American Statistical Association*, 113(523):1228–1242, 2018.

S. Athey, J. Tibshirani, and S. Wager. Generalized random forests. *Annals of Statistics*, 47(2):1148–1178, 2019.

A. Ramasesh, A. Lewkowycz, and E. Dyer. Effect of scale on catastrophic forgetting in neural networks. In *Proc. ICLR*, April 2022.

E. J. Hu et al. LoRA: Low-rank adaptation of large language models. In *Proc. ICLR*, April 2022.

T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Proc. NeurIPS*, volume 36, December 2023.

N. Houlsby et al. Parameter-efficient transfer learning for NLP. In *Proc. ICML*, pages 2790–2799, June 2019.

X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proc. ACL*, pages 4582–4597, August 2021.

H. Liu et al. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Proc. NeurIPS*, volume 35, December 2022.

V. Lialin, V. Deshpande, and A. Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.15647*, March 2023.

G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, March 2015.

V. Sanh et al. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, October 2019.

E. Frantar et al. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *Proc. ICLR*, May 2023.

J. Lin et al. AWQ: Activation-aware weight quantization for LLM compression and acceleration. *arXiv preprint arXiv:2306.00978*, June 2023.

G. Xiao et al. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proc. ICML*, July 2023.

P. Michel, O. Levy, and G. Neubig. Are sixteen heads really better than one? In *Proc. NeurIPS*, volume 32, December 2019.

Z. Xia et al. Sheared LLaMA: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*, October 2023.

P. Christiano et al. Deep reinforcement learning from human preferences. In *Proc. NeurIPS*, volume 30, December 2017.

L. Ouyang et al. Training language models to follow instructions with human feedback. In *Proc. NeurIPS*, volume 35, pages 27730–27744, December 2022.

R. Rafailov et al. Direct preference optimization: Your language model is secretly a reward model. In *Proc. NeurIPS*, volume 36, December 2023.

Y. Bai et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, December 2022.

K. Singhal et al. Large language models encode clinical knowledge. *Nature*, 620:172–180, July 2023.

S. Wu et al. BloombergGPT: A large language model for finance. *arXiv preprint arXiv:2303.17564*, March 2023.

W. Kwon et al. Efficient memory management for large language model serving with PagedAttention. In *Proc. SOSP*, October 2023.

NVIDIA. TensorRT-LLM. GitHub repository, 2023.

Y. Leviathan, M. Kalman, and Y. Matias. Fast inference from transformers via speculative decoding. In *Proc. ICML*, July 2023.

D. Sculley et al. Hidden technical debt in machine learning systems. In *Proc. NeurIPS*, volume 28, December 2015.

H. Chen et al. Chatbot Arena: An open platform for evaluating LLMs by human preference. *arXiv preprint arXiv:2403.04132*, March 2024.

Y. Zheng et al. LlamaFactory: Unified efficient fine-tuning of 100+ language models. *arXiv preprint arXiv:2403.13372*, March 2024.

OpenAccess AI Collective. Axolotl. GitHub repository, 2023.

D. Hendrycks et al. Measuring massive multitask language understanding. In *Proc. ICLR*, May 2021.

M. Chen et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, July 2021.

S. Lin, J. Hilton, and O. Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proc. ACL*, pages 3214–3252, May 2022.

L. Zheng et al. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Proc. NeurIPS*, volume 36, December 2023.

A. Zou et al. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, July 2023.

S. Gehman et al. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of EMNLP*, pages 3356–3369, November 2020.

Defense Information Systems Agency (DISA). DoD Cloud Computing Security Requirements Guide (SRG). v1, Release 7, April 2021.

U.S. Dept. Health & Human Services. Summary of the HIPAA Privacy Rule. 2003.

577 Industries R&D Lab. FORGE OS: The Agent-Legible Operating System — Platform Spine Specification. Technical report, 577 Industries Incorporated, 2025.

577 Industries R&D Lab. FORGE QBit: Heterogeneous Post-Quantum Cryptography and Physics-Informed Computation for Sovereign AI Systems. Technical report, 577 Industries Incorporated, 2025.

577 Industries R&D Lab. FORGE Memory: Governed Multi-Agent Orchestration with Predictive Human-in-the-Loop Intelligence. Technical report, 577 Industries Incorporated, 2025.

577 Industries R&D Lab. FORGE Kinetic: Autonomous Multi-Agent Swarm Intelligence for Sovereign Edge Deployment. Technical report, 577 Industries Incorporated, 2025.