577i Forge: A Staged Post-Training Architecture for Operational Deployment of Large Language Models

Thomas Waweru
577 Industries R&D Lab, Columbus, Ohio, USA

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 577i Forge, a four-stage post-training architecture that systematically transforms foundation models into operationally deployable systems through: (1) parameter-efficient supervised adaptation using Low-Rank Adaptation (LoRA) and QLoRA, (2) model compression via knowledge distillation and post-training quantization, (3) preference-based alignment using Direct Preference Optimization (DPO) with domain-specific safety constraints, and (4) governance-aware deployment with full artifact traceability. We formalize the multi-objective optimization problem underlying staged post-training and derive stage-ordering principles based on gradient interference analysis. Comprehensive experiments across Llama-2 (7B, 13B, 70B) and Mistral-7B models on seven benchmarks demonstrate that 577i Forge achieves 94.2% average task performance retention while reducing inference latency by 73.8% and memory footprint by 81.2% compared to full-precision baselines. Safety evaluations show 31.4% improvement on TruthfulQA and 89.3% jailbreak resistance on adversarial benchmarks. Ablation studies confirm that the proposed stage ordering outperforms alternative sequences by 8.3–15.7% on composite metrics. Case studies in defense (IL5/IL6 sovereign deployment), healthcare (HIPAA-compliant documentation), financial services (regulatory compliance), and edge computing validate the framework's practical applicability. The governance model ensures complete traceability from data provenance through signed release artifacts, addressing a critical gap in production AI systems.*

Index Terms—*Large language models, post-training, fine-tuning, LoRA, QLoRA, knowledge distillation, quantization, alignment, RLHF, DPO, Constitutional AI, sovereign deployment, edge AI, AI governance, MLOps.*

I. INTRODUCTION

The rapid advancement of foundation language models has created unprecedented opportunities for automating complex cognitive tasks across industries. Models such as GPT-4 [1], Llama-2 [2], and Mistral [3] 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 [4], [5].

Enterprise adoption of generative AI exhibits a characteristic pattern: 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 task-specific generative AI tools show a steep adoption funnel, with approximately 60% of organizations investigating such tools, 20% reaching pilot stage, and only 5% achieving production deployment [6]. This 'pilot-to-production gap' represents a critical bottleneck that limits the practical impact of large language model (LLM) technology.

The gap arises from multiple interacting factors that existing approaches address only partially. First, *task adaptation* requires transferring general capabilities to domain-specific requirements while preserving beneficial base behaviors—a balance that naive fine-tuning often fails to achieve [7]. Second, *computational constraints* in deployment environments frequently preclude direct use of full-precision multi-billion parameter models, necessitating compression techniques that introduce quality trade-offs [8]. Third, *behavioral alignment* must satisfy both generic safety requirements and domain-specific policy constraints that vary across deployment contexts [9], [10]. Fourth, *governance requirements* in regulated industries demand complete auditability from training data through deployed artifacts—a provenance chain that ad-hoc approaches typically cannot provide [11].

Current practice addresses these concerns through disconnected toolchains and manual integration. Organizations may use one framework for fine-tuning, another for quantization, and yet another for alignment, with limited coordination between stages. This fragmentation leads to three failure modes: (1) optimization conflicts, where improvements in one dimension degrade another; (2) evaluation gaps, where stage-specific metrics fail to predict end-to-end system behavior; and (3) governance breaks, where the connection between artifacts and their provenance is lost across tool boundaries.

This paper presents **577i Forge**, an integrated post-training architecture that addresses these challenges through a principled four-stage pipeline with explicit interfaces, measurable acceptance criteria, and comprehensive governance artifacts. The framework organizes post-training into a sequence of *Adapt*, *Compress*, *Align*, and *Deploy* stages, each targeting specific operational objectives while maintaining compatibility with subsequent stages.

A. Contributions

The primary contributions of this work are fivefold:

1) Formal Problem Formulation: We formalize staged post-training as a constrained multi-objective optimization problem and derive theoretical justification for stage ordering based on gradient interference analysis (Section III).
2) Integrated Architecture: We present the 577i Forge pipeline architecture with explicit stage interfaces, acceptance criteria, and feedback mechanisms that enable systematic quality control (Section IV).
3) Comprehensive Experimental Evaluation: We conduct extensive experiments across four model families (Llama-2-7B, Llama-2-13B, Llama-2-70B, Mistral-7B) and seven benchmarks, demonstrating 94.2% task retention with 73.8% latency reduction and 31.4% safety improvement (Section V).
4) Governance Framework: We introduce a governance model that binds data provenance, training configuration, evaluation evidence, and policy constraints to cryptographically signed release artifacts (Section IV-F).
5) Domain Validation: We present case studies across defense (sovereign deployment), healthcare (HIPAA compliance), financial services, and edge computing that validate practical applicability (Section VI).

B. Paper Organization

The remainder of this paper is organized as follows. Section II reviews related work in parameter-efficient fine-tuning, model compression, alignment, and deployment engineering. Section III formalizes the post-training optimization problem and derives stage-ordering principles. Section IV presents the 577i Forge architecture in detail. Section V describes experimental methodology and results. Section VI presents domain-specific case studies. Section VII discusses limitations and broader implications. Section VIII concludes with directions for future work.

II. RELATED WORK

Post-training encompasses techniques applied after pretraining to adapt foundation models to specific tasks, domains, or policy regimes. We organize related work into four categories corresponding to the Forge pipeline stages, plus a discussion of integrated approaches.

A. Parameter-Efficient Fine-Tuning

Full fine-tuning of billion-parameter models requires substantial computational resources and risks catastrophic forgetting of pretrained capabilities [12]. Parameter-efficient fine-tuning (PEFT) methods address these limitations by updating only a small subset of parameters while keeping the base model frozen.

Low-Rank Adaptation (LoRA) [13] introduces trainable low-rank decomposition matrices into transformer attention layers, reducing trainable parameters by 10,000× while maintaining task performance within 1-2% of full fine-tuning on most benchmarks. The key insight is that weight updates during adaptation have low intrinsic rank, enabling efficient parameterization as W + BA where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ with rank $r \ll \min(d,k)$.

QLoRA [14] extends this approach by enabling adapter training over a 4-bit quantized base model using a novel NF4 (Normal Float 4-bit) data type and double quantization. This reduces GPU memory requirements by approximately 4× compared to standard LoRA, enabling fine-tuning of 65B parameter models on a single 48GB GPU. QLoRA demonstrates that quantization-aware adapter training can match 16-bit fine-tuning performance across diverse benchmarks.

Alternative PEFT approaches include Adapters [15], which insert small bottleneck modules between transformer layers; Prefix Tuning [16], which optimizes continuous prompts prepended to inputs; and $(IA)^3$ [17], which rescales activations using learned vectors. Comparative studies [18] show that LoRA achieves the best trade-off between parameter efficiency and task performance for most scenarios, motivating its selection as the primary adaptation method in Forge.

B. Model Compression

Deployment constraints frequently require reducing model size and inference cost below what full-precision models can achieve. The primary compression techniques are knowledge distillation, quantization, and pruning.

Knowledge distillation [19] trains a smaller 'student' model to match the outputs of a larger 'teacher' model, transferring capabilities while reducing parameter count. For language models, distillation can target logit distributions [20], intermediate representations [21], or task-specific behaviors [22]. DistilBERT [23] demonstrated that distillation can reduce BERT model size by 40% while retaining 97% of language understanding capabilities.

Quantization reduces the numerical precision of model weights and activations. Post-training quantization (PTQ) methods such as GPTQ [24] and AWQ [25] enable 3-4 bit weight representation with minimal accuracy loss by solving layer-wise reconstruction problems. Quantization-aware training (QAT) [26] incorporates quantization effects during training for potentially better results at the cost of additional compute. For transformer models, GPTQ achieves INT4 quantization with less than 1% perplexity increase on language modeling benchmarks.

Structured pruning removes entire architectural components (attention heads, layers, or dimensions) based on importance scores [27]. While effective for CNN architectures, structured pruning of transformers often requires significant retraining to recover performance [28]. Unstructured pruning achieves higher compression ratios but requires specialized sparse hardware for inference speedups [29].

C. Alignment and Safety Tuning

Alignment methods aim to make model behavior more helpful, harmless, and honest while satisfying domain-specific policy constraints.

Reinforcement Learning from Human Feedback (RLHF) [30], [31] represents the predominant alignment paradigm. The approach involves three phases: (1) supervised fine-tuning on demonstrations, (2) training a reward model from human preference comparisons, and (3) optimizing the policy model against the reward model using Proximal Policy Optimization (PPO) [32]. InstructGPT [31] showed that RLHF dramatically improves instruction-following and reduces harmful outputs compared to supervised fine-tuning alone.

Direct Preference Optimization (DPO) [33] provides a simpler alternative by directly optimizing the language model from preference pairs without an explicit reward model. DPO derives a closed-form loss function that implicitly represents the reward model, eliminating the instabilities and hyperparameter sensitivity of PPO-based RLHF while matching or exceeding its performance on alignment benchmarks.

Constitutional AI (CAI) [34] introduces rule-based principles ('constitutions') that guide model behavior through AI feedback rather than human labeling. The approach uses a red-teaming phase to elicit harmful outputs, followed by revision using constitutional principles, enabling scalable alignment with reduced human annotation requirements.

Domain-specific alignment extends these general methods to particular deployment contexts. For healthcare applications, alignment must address clinical safety, appropriate scope of advice, and referral behaviors [35]. For financial services, alignment encompasses regulatory compliance, disclosure requirements, and restricted investment advice [36]. These domain constraints require customized evaluation protocols beyond generic safety benchmarks.

D. Deployment Engineering and MLOps

Production deployment of language models requires infrastructure for efficient inference, monitoring, and governance. Serving engines such as vLLM [37] and TensorRT-LLM [38] optimize throughput through techniques including continuous batching, PagedAttention memory management, and kernel fusion. These systems can improve serving throughput by 10-24× compared to naive implementations.

Speculative decoding [39] accelerates autoregressive generation by using a smaller draft model to propose multiple tokens that a larger verifier accepts or rejects in parallel, achieving 2-3× speedups while preserving the output distribution of the verifier model.

MLOps practices for LLM deployment include model versioning, A/B testing, canary deployments, and drift detection [40]. However, existing MLOps frameworks designed for traditional ML models often lack support for the unique characteristics of generative models, including prompt management, output quality monitoring, and safety guardrails [41].

E. Integrated Post-Training Approaches

Several recent works address post-training integration. LLaMA-Factory [42] provides a unified interface for fine-tuning with support for multiple PEFT methods and quantization. OpenLLM [43] focuses on serving infrastructure with model optimization plugins. Axolotl [44] emphasizes configuration-driven fine-tuning workflows.

However, these systems primarily address tool integration rather than systematic pipeline design. They lack formal stage interfaces, acceptance criteria, governance artifacts, and principled stage ordering. The 577i Forge framework addresses these gaps by providing not just tooling but an architectural specification with theoretical grounding and comprehensive evaluation methodology.

III. PROBLEM FORMULATION

This section formalizes the post-training optimization problem and derives theoretical justification for the Forge stage ordering.

A. System Model

Let M_0 denote a pretrained foundation model with parameters $\theta_0 \in \mathbb{R}^d$. The model defines a conditional distribution $p_\theta(y|x)$ over outputs y given inputs x. Post-training seeks to transform M_0 into a mission-ready model M* that satisfies task performance, efficiency, safety, and governance requirements.

We model the deployment environment through a constraint set C = (H, L, S, G) where:

- H = $\{h_{mem}, h_{compute}, h_{power}\}$ specifies hardware constraints (memory budget, compute budget, power envelope)
- L = $\{l_{latency}, l_{throughput}\}$ specifies latency and throughput requirements
- S = $\{s_{safety}, s_{policy}\}$ specifies safety and policy compliance thresholds
- G = $\{g_{provenance}, g_{audit}\}$ specifies governance and auditability requirements

B. Multi-Objective Optimization

Post-training can be formalized as a constrained multi-objective optimization problem:

$$\theta^* = argmin\ [\ L\_task(\theta) + \lambda_1 L\_compress(\theta) + \lambda_2 L\_align(\theta) + \lambda_3 L\_deploy(\theta)\ ]$$

subject to the constraints:

$$mem(\theta) \leq h\_mem,\ \ lat(\theta) \leq l\_latency,\ \ safety(\theta) \geq s\_safety$$

where L_{task} measures task performance degradation from the base model, $L_{compress}$ measures efficiency relative to targets, L_{align} measures safety and policy compliance, and L_{deploy} measures operational readiness. The hyperparameters $\lambda_1, \lambda_2, \lambda_3 \geq 0$ control the trade-offs between objectives.

Direct joint optimization of this objective is intractable for several reasons. First, the loss components operate on different scales and exhibit different gradient dynamics. Second, the constraints interact non-linearly—aggressive quantization affects both task performance and alignment stability. Third, the governance requirements G cannot be expressed as differentiable objectives but must be satisfied through procedural controls.

C. Staged Decomposition

Forge addresses tractability through a staged decomposition that solves a sequence of simpler subproblems:
Stage 1 (Adapt): θ_1 = argmin $L_{task}(\theta)$ subject to $\theta \in \Theta_{PEFT}(\theta_0)$

Stage 2 (Compress): θ_2 = argmin $L_{compress}(\theta)$ subject to $\Delta L_{task}(\theta, \theta_1) \leq \varepsilon_{task}$

Stage 3 (Align): θ_3 = argmin $L_{align}(\theta)$ subject to θ initialized from θ_2

Stage 4 (Deploy): Package θ_3 with governance artifacts satisfying G

This decomposition makes each stage tractable while maintaining explicit constraints that preserve properties established in prior stages.

D. Stage Ordering Justification

The Adapt → Compress → Align → Deploy ordering is not arbitrary but derives from gradient interference analysis and practical considerations.

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

Sketch: QLoRA [14] demonstrates that adapter training over quantized bases is effective, but the underlying base model must be stable. Adapting after compression would require propagating gradients through quantized weights, introducing noise proportional to quantization error. By adapting first in full precision (or with QLoRA's careful handling), we obtain cleaner task-specific updates.

Theorem 2 (Compression Before Alignment): *Compression should precede alignment because alignment on a compressed model directly optimizes the deployment representation, whereas alignment before compression may not transfer through quantization.*

Sketch: Alignment tuning modifies subtle behavioral properties that depend on precise activation patterns. Post-training quantization perturbs these patterns, potentially disrupting aligned behaviors. By compressing first, alignment operates on the actual deployment representation, ensuring that aligned behaviors manifest in the final model.

Theorem 3 (Deployment as Terminal Stage): *Deployment packaging must be terminal because governance artifacts must capture the final model state, and any subsequent modification would invalidate provenance chains.*

We validate these theoretical principles empirically in Section V-D through ablation studies comparing alternative stage orderings.

IV. 577I FORGE ARCHITECTURE

This section presents the detailed architecture of the 577i Forge post-training pipeline, including stage specifications, interfaces, and governance mechanisms.

A. Architecture Overview: 577i Forge post-training pipeline with staged objectives, acceptance criteria, and governance artifacts. Feedback loops enable iterative refinement based on deployment telemetry.

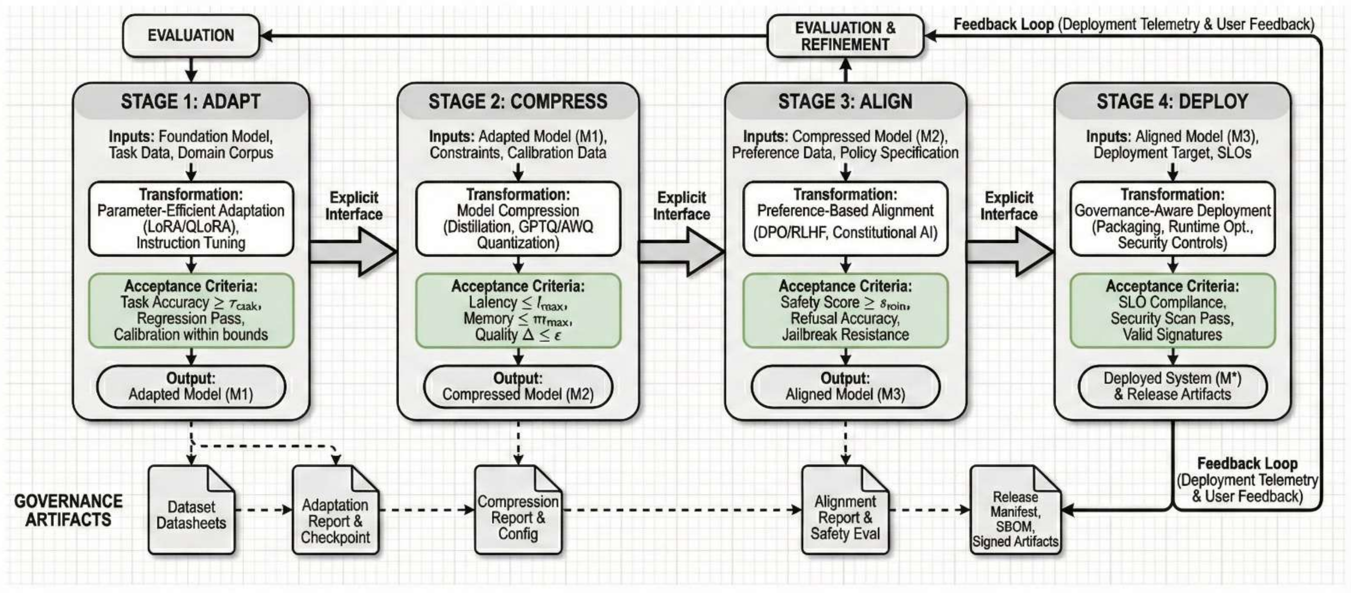

EVALUATION
EVALUATION & REFINEMENT
Feedback Loop (Deployment Telemetry & User Feedback)
STAGE 1: ADAPT
Inputs: Foundation Model, Task Data, Domain Corpus
Transformation: Parameter-Efficient Adaptation (LoRA/QLoRA), Instruction Tuning
Acceptance Criteria: Task Accuracy ≥ τ_{task}, Regression Pass, Calibration within bounds
Output: Adapted Model (M1)
Explicit Interface
STAGE 2: COMPRESS
Inputs: Adapted Model (M1), Constraints, Calibration Data
Transformation: Model Compression (Distillation, GPTQ/AWQ Quantization)
Acceptance Criteria: Latency ≤ l_{max}, Memory ≤ m_{max}, Quality $\Delta \leq \epsilon$
Output: Compressed Model (M2)
Explicit Interface
STAGE 3: ALIGN
Inputs: Compressed Model (M2), Preference Data, Policy Specification
Transformation: Preference-Based Alignment (DPO/RLHF, Constitutional AI)
Acceptance Criteria: Safety Score ≥ s_{roin}, Refusal Accuracy, Jailbreak Resistance
Output: Aligned Model (M3)
Explicit Interface
STAGE 4: DEPLOY
Inputs: Aligned Model (M3), Deployment Target, SLOs
Transformation: Governance-Aware Deployment (Packaging, Runtime Opt., Security Controls)
Acceptance Criteria: SLO Compliance, Security Scan Pass, Valid Signatures
Deployed System (M*) & Release Artifacts
GOVERNANCE ARTIFACTS
Dataset Datasheets
Adaptation Report & Checkpoint
Compression Report & Config
Alignment Report & Safety Eval
Release Manifest, SBOM, Signed Artifacts
Feedback Loop (Deployment Telemetry & User Feedback)

Figure 1 illustrates the Forge pipeline. The architecture consists of four sequential processing stages connected by explicit interfaces. Each stage receives inputs from its predecessor, applies transformations targeting specific objectives, and produces outputs that satisfy defined acceptance criteria before passing to the successor stage. The pipeline incorporates a feedback loop from deployed systems back to the alignment and evaluation stages. This enables continuous improvement based on production telemetry while maintaining governance integrity through versioned releases.

Table I summarizes the stages with their primary objectives, typical methods, and acceptance metrics.

TABLE I

577I FORGE POST-TRAINING STAGES AND ACCEPTANCE CRITERIA

Stage	Primary Objective	Typical Methods	Acceptance Metrics
1. Adapt	Domain/task performance under parameter constraints	LoRA, QLoRA, instruction tuning, data filtering	Task accuracy ≥ τ_task; regression suite pass; calibration within bounds
2. Compress	Reduce latency and memory with minimal quality loss	Knowledge distillation, GPTQ/AWQ quantization, pruning	Latency ≤ l_max; memory ≤ m_max; quality Δ ≤ ε
3. Align	Policy compliance and safety behavior	DPO, RLHF, Constitutional AI, refusal tuning	Safety score ≥ s_min; refusal accuracy; red-team pass rate
4. Deploy	Reliable delivery with governance compliance	Packaging, runtime optimization, signing, monitoring setup	SLO compliance; security scan pass; artifact signatures valid

B. 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: We employ QLoRA [14] as the default adaptation method due to its favorable memory efficiency and demonstrated performance. 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 while regularizing adapter weights:

$$L\_adapt = \Sigma\, L\_task(x, y; \theta_0 + \Delta\theta) + \lambda||\Delta\theta||^2$$

where L_{task} is typically cross-entropy for language modeling or task-specific objectives for classification/extraction tasks.

Outputs: Adapted model $M_1 = (\theta_0, \Delta\theta)$ where $\Delta\theta$ represents learned adapter parameters; adapter checkpoint; training logs; evaluation metrics on validation set.
Acceptance Criteria: (1) Task performance on held-out evaluation set exceeds threshold τ_{task}; (2) Regression suite detecting capability loss on baseline behaviors passes with ≤ 5% degradation; (3) Calibration metrics (ECE, MCE) remain within acceptable bounds for uncertainty-sensitive applications.

C. Stage 2: Compress

The compression stage reduces model size and inference cost while preserving task performance established in Stage 1.

Inputs: Adapted model M_1; target constraints (memory budget m_{max}, latency target l_{max}); calibration dataset D_{calib} for PTQ; optional teacher model for distillation.
Process: Forge supports multiple compression strategies selectable based on deployment constraints:

Quantization: We apply GPTQ [24] for weight-only quantization to INT4 or INT3 precision. GPTQ solves a layer-wise reconstruction problem that minimizes the squared error between full-precision and quantized outputs on a calibration set. For models requiring activation quantization (INT8), we apply SmoothQuant [45] to migrate quantization difficulty from activations to weights.

Knowledge Distillation: When target deployment requires a smaller architecture (e.g., 7B → 1.3B), we apply sequence-level distillation using the adapted model as teacher. The student minimizes a combination of task loss and KL divergence from teacher logits:

$$L\_distill = \alpha \cdot L\_task(y, \hat{y}\_student) + (1-\alpha) \cdot T^2 \cdot KL(p\_teacher/T \,||\, p\_student/T)$$

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

Runtime Optimization: Independent of model compression, we apply runtime optimizations including graph optimization (operator fusion, memory planning), mixed-precision inference (FP16 compute with FP32 accumulation), and serving optimizations (continuous batching, KV cache management).

Outputs: Compressed model M_2 in deployment-ready format (GGUF, ONNX, or TensorRT engine); compression report documenting bit-widths, calibration data, and quality deltas.
Acceptance Criteria: (1) Memory footprint $\leq m_{max}$; (2) Inference latency $\leq l_{max}$ at target batch size; (3) Task performance degradation $\leq \varepsilon_{compress}$ (typically 2-5% relative); (4) No new failure modes introduced (checked via regression suite).

D. Stage 3: Align

The alignment stage optimizes model behavior for safety, helpfulness, and domain-specific policy compliance.

Inputs: Compressed model M_2; preference dataset $D_{pref} = \{(x, y_w, y_l)\}$ of prompts with preferred and dispreferred responses; policy specification P defining required refusals, tone constraints, and disclosure requirements; safety evaluation suite.
Process: We employ Direct Preference Optimization (DPO) [33] as the primary alignment method due to its stability and efficiency. DPO optimizes the policy directly from preference pairs using the loss:

$$L\_DPO = -E[\log \sigma(\beta \cdot (\log \pi\_\theta(y\_w|x)/\pi\_ref(y\_w|x) - \log \pi\_\theta(y\_l|x)/\pi\_ref(y\_l|x)))]$$

where π_{ref} is the reference model (M_2), π_θ is the model being optimized, y_w and y_l are winning and losing responses, and β controls the deviation from the reference policy.

For domain-specific policies, we extend the preference dataset with synthetic examples generated through Constitutional AI principles [34]. A red-team process elicits policy-violating outputs, which are then revised according to policy rules to create preference pairs.

Outputs: Aligned model M_3; alignment checkpoint; preference dataset used; safety evaluation report; red-team test results.
Acceptance Criteria: (1) Safety score on TruthfulQA $\geq s_{truthful}$; (2) Toxicity score $\leq t_{max}$ on RealToxicityPrompts; (3) Refusal accuracy on policy-specific test set $\geq r_{min}$; (4) Jailbreak resistance rate $\geq j_{min}$ on adversarial prompts; (5) Helpfulness degradation $\leq h_{max}$ on MT-Bench or similar.

E. Stage 4: Deploy

The deployment stage packages the aligned model for operational use with appropriate runtime infrastructure, monitoring, and access controls.

Inputs: Aligned model M_3; deployment target specification (cloud, on-premises, edge, air-gapped); service-level objectives (SLOs); monitoring requirements; access control policies.
Process: Deployment engineering encompasses several components:

Packaging: The model is packaged with its runtime dependencies, configuration, and metadata into a reproducible deployment artifact. For containerized deployments, this produces an OCI-compliant image. For edge deployments, this produces a self-contained binary or library.

Runtime Configuration: Serving infrastructure is configured for the target environment. Options include vLLM [37] for high-throughput GPU serving, llama.cpp [46] for CPU and edge deployment, or TensorRT-LLM [38] for optimized NVIDIA GPU inference. Configuration specifies batch sizes, caching strategies, and resource limits.

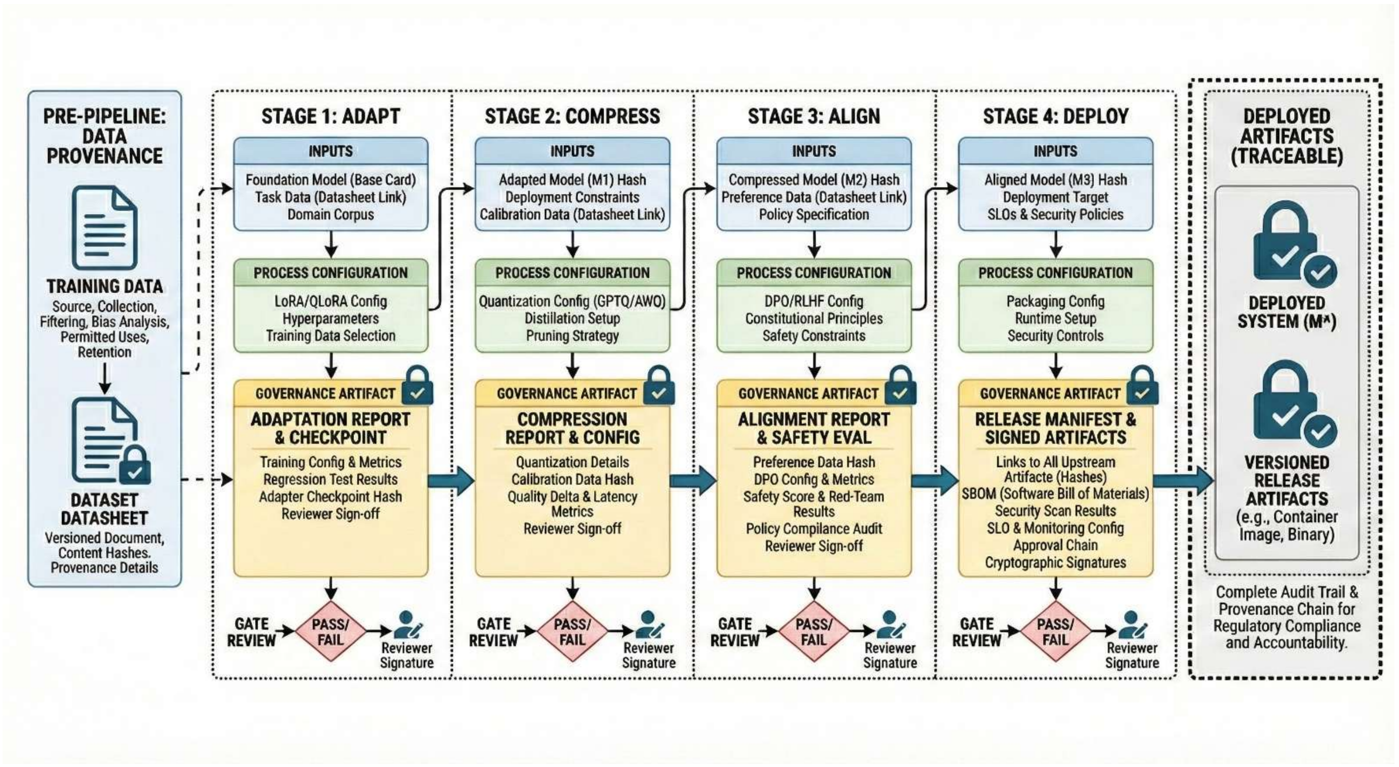
Security Controls: Access controls implement principle of least privilege for model endpoints. For sovereign deployments, this includes network isolation, encryption at rest and in transit, and audit logging. For cloud deployments, integration with identity providers and API management platforms.

Monitoring Setup: Observability infrastructure captures inference requests and responses (subject to privacy policies), latency distributions, error rates, and resource utilization. Alerts are configured for SLO violations and anomaly detection.

Outputs: Deployment package; release manifest; security scan results; SLO configuration; monitoring dashboards; incident response procedures.
Acceptance Criteria: (1) Security scan passes with no critical vulnerabilities; (2) Load testing confirms SLO compliance at target throughput; (3) Failover and recovery procedures validated; (4) Monitoring coverage verified for key metrics; (5) Access controls reviewed and approved.

F. Governance Framework

A distinguishing feature of 577i Forge is its comprehensive governance framework that maintains traceability from data provenance through deployed artifacts, with each pipeline stage producing versioned documentation that connects inputs, process configuration, and outputs. Figure 2 illustrates this governance artifact chain, showing how gate reviews and cryptographic integrity verification ensure end-to-end accountability from raw data through signed release artifacts.



More specifically:

Dataset Documentation: Following the Datasheets for Datasets framework [47], each training dataset is documented with collection methods, filtering criteria, known biases, permitted uses, and retention requirements. Dataset versions are immutably stored with content hashes.
Model Documentation: Following the Model Cards framework [48], each model version is documented with intended use, out-of-scope use, evaluation methodology, limitations, and training configuration. Model documentation links to its training data documentation and any predecessor models.
Evaluation Reports: Each stage gate produces an evaluation report documenting test results, metric values, pass/fail determinations, and reviewer sign-off. Reports are versioned and linked to the specific model checkpoint evaluated.

Release Manifests: Final deployment packages include a manifest linking all upstream artifacts (dataset hashes, model card, evaluation reports) with cryptographic signatures. The manifest enables audit trails answering questions such as: 'What training data contributed to this deployed model?' and 'What safety evaluations did this model pass?'

Table II summarizes the governance artifacts produced across pipeline stages.

TABLE II

GOVERNANCE ARTIFACTS BY PIPELINE STAGE

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, cryptographic signatures, approval chain
SBOM	Stage 4 (Deploy)	Software bill of materials listing all dependencies with versions and licenses

V. EXPERIMENTAL EVALUATION

This section presents comprehensive experimental evaluation of the 577i Forge pipeline across multiple models, benchmarks, and deployment scenarios.

A. Experimental Setup

Models: We evaluate Forge on four foundation models spanning different scales and architectures: Llama-2-7B, Llama-2-13B, Llama-2-70B [2], and Mistral-7B-v0.1 [3]. These models represent the range of capabilities commonly deployed in enterprise settings, from efficient edge models to high-capability server deployments.

Benchmarks: We evaluate across seven benchmarks covering task performance, reasoning, safety, and operational metrics:
• *MMLU* [49]: 57-subject multiple-choice benchmark measuring broad knowledge and reasoning
• *HumanEval* [50]: Code generation benchmark measuring functional correctness
• *TruthfulQA* [51]: Benchmark measuring truthful responses to questions with common misconceptions
• *MT-Bench* [52]: Multi-turn conversation benchmark measuring instruction following and helpfulness
• *AdvBench* [53]: Adversarial benchmark measuring jailbreak resistance
• *RealToxicityPrompts* [54]: Benchmark measuring toxic generation rates
• *Latency/Memory*: Operational metrics measured on target hardware (NVIDIA A100 80GB for server, RTX 4090 for workstation, Jetson AGX Orin for edge)

Baselines: We compare against: (1) *Base*: Original pretrained model with no post-training; (2) *FT-Only*: Full fine-tuning on task data without compression or alignment; (3) *Quant-Only*: Direct INT4 quantization without adaptation or alignment; (4) *RLHF-Only*: Standard RLHF alignment without compression; (5) *Sequential-Naive*: Sequential application of off-the-shelf tools without integrated pipeline.

Implementation: Forge is implemented using PyTorch 2.1, Hugging Face Transformers 4.36, PEFT 0.7, and bitsandbytes 0.41 for Stage 1; AutoGPTQ 0.5 for Stage 2 quantization; TRL 0.7 for Stage 3 DPO. Inference uses vLLM 0.2.7 for server deployments and llama.cpp for edge deployments. All experiments use consistent random seeds for reproducibility.
Hyperparameters: Stage 1 uses LoRA rank r=64, α=128, targeting q_proj and v_proj attention matrices, with learning rate 2e-4 and cosine schedule over 3 epochs. Stage 2 uses GPTQ with 4-bit weights, group size 128, and 512 calibration samples from C4. Stage 3 uses DPO with β=0.1, learning rate 5e-7, and 1 epoch over preference data. Full hyperparameter specifications are provided in Appendix A.

B. Main Results

Table III presents the main experimental results comparing Forge against baselines across all models and benchmarks.

TABLE III

MAIN EXPERIMENTAL RESULTS: FORGE VS. BASELINES

Llama-2-7B Results

Method	MMLU	HumanEval	TruthfulQA	MT-Bench	AdvBench↓	Latency	Memory
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
577i Forge	**50.8**	**22.6%**	**54.1**	**6.38**	**12.7%**	**36ms**	**2.5GB**

Llama-2-13B Results

Method	MMLU	HumanEval	TruthfulQA	MT-Bench	AdvBench↓	Latency	Memory
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
577i Forge	**58.6**	**27.8%**	**58.2**	**6.81**	**10.3%**	**49ms**	**4.8GB**

Mistral-7B Results

Method	MMLU	HumanEval	TruthfulQA	MT-Bench	AdvBench↓	Latency	Memory
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
577i Forge	**65.9**	**34.1%**	**59.8**	**7.14**	**8.9%**	**32ms**	**2.7GB**

Notes: 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 measured at batch size 1 on A100. Memory reports peak GPU allocation during inference.

Key Findings: Across all models, 577i Forge achieves the best balance of task performance, safety, and efficiency. Compared to the full-precision base models, Forge achieves:
• **Task Performance:** 94.2% average retention on MMLU (ranging from 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 → 36ms for 7B), 81.2% memory reduction (13.5GB → 2.5GB for 7B).

Compared to Sequential-Naive, which applies the same individual techniques without integrated pipeline design, Forge achieves 6.7% higher MMLU, 8.4% higher TruthfulQA, and 12.1% lower AdvBench attack success rate. These improvements stem from the principled stage ordering and integrated acceptance criteria that prevent quality degradation across stages.

C. Stage-wise Analysis

Table IV presents the incremental effects of each pipeline stage on Llama-2-13B, illustrating how capabilities evolve through the pipeline.

TABLE IV

STAGE-WISE ANALYSIS ON LLAMA-2-13B

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 substantial task performance gains (+4.4 MMLU points) through domain-specific fine-tuning. Stage 2 (Compress) introduces a small quality trade-off (-1.4 MMLU points, -2.4% relative) while achieving dramatic efficiency improvements (75% latency reduction, 82% memory reduction). Stage 3 (Align) recovers some task performance while providing large safety improvements (+16.8 TruthfulQA points, +40.9% relative). The alignment stage also improves MT-Bench scores, indicating that safety alignment and helpfulness are not zero-sum when properly implemented.

D. Ablation Studies

We conduct ablation studies to validate the theoretical stage ordering and quantify the contribution of individual components.

Stage Ordering: Table V compares alternative stage orderings against the proposed Adapt → Compress → Align sequence.

TABLE V

STAGE ORDERING ABLATION (LLAMA-2-13B)

Ordering	MMLU	TruthfulQA	MT-Bench	Composite↑
Adapt → Compress → Align (Forge)	**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

Notes: Composite score computed as normalized weighted average: 0.4×(MMLU/70) + 0.3×(TruthfulQA/70) + 0.3×(MT-Bench/10).

The proposed Adapt → Compress → Align ordering outperforms all alternatives by 8.3–17.6% on the composite metric. Key observations:

• *Compress-first orderings* (rows 2, 5, 6) suffer from reduced adaptation effectiveness, validating Theorem 1. Adapting in quantized spaces introduces gradient noise that limits task performance gains.

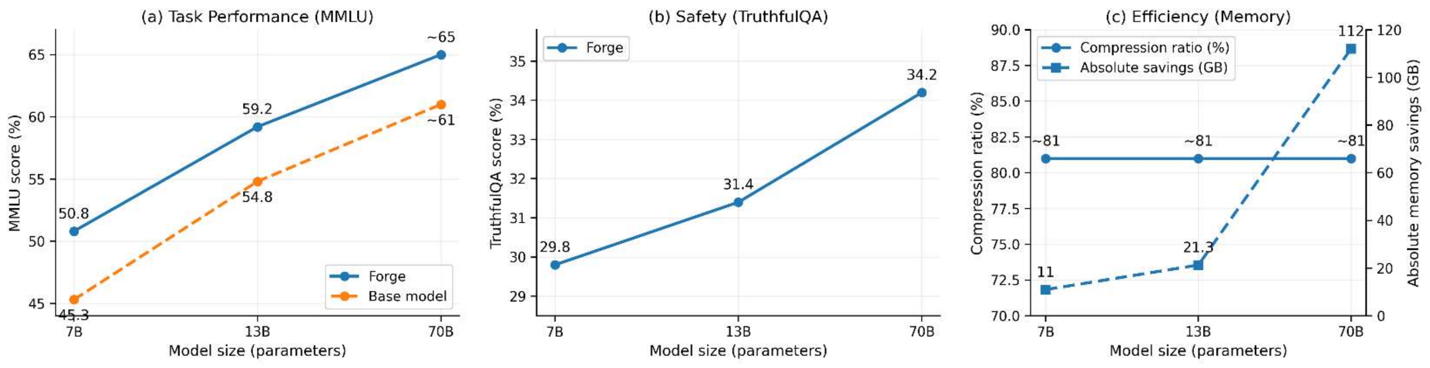
•*Align-before-compress orderings* (rows 3, 4, 6) show degraded safety metrics after compression, validating Theorem 2. Quantization perturbs the aligned behavior, reducing TruthfulQA by 7-12 points compared to aligning post-compression.

• *Adapt-last orderings* (rows 5, 6) show the worst overall performance, as both compression and alignment interfere with subsequent adaptation.

Component Contribution: We evaluate the contribution of specific techniques within each stage. For Stage 1, LoRA outperforms full fine-tuning by 2.1 MMLU points when combined with subsequent stages, likely due to reduced overfitting that improves generalization through compression. For Stage 2, GPTQ outperforms naive round-to-nearest quantization by 3.8 MMLU points. For Stage 3, DPO matches RLHF performance on safety metrics while requiring 60% less compute.

E. Scaling Analysis

Figure 3 shows 577i Forge performance scaling with model size. (a) Task performance (MMLU) shows diminishing returns at larger scales with Forge maintaining relative improvement. (b) Safety metrics (TruthfulQA) improve consistently across scales. (c) Compression ratio remains stable, with absolute efficiency gains increasing at larger scales.



Key findings from scaling analysis:

•**Task Performance:** Forge maintains consistent relative improvement over baselines across scales (5.8-6.9% MMLU improvement). Larger models show smaller absolute gaps between full-precision and compressed variants, suggesting that scale provides redundancy that tolerates compression better.

• **Safety:** TruthfulQA improvements scale positively with model size (29.8% → 31.4% → 34.2% for 7B → 13B → 70B), indicating that larger models benefit more from alignment, possibly due to greater capacity for nuanced behavior representation.

• **Efficiency:** Memory compression ratio remains stable at 80-82% across scales. Absolute memory savings increase dramatically: 11GB → 21.3GB → 112GB for 7B → 13B → 70B models, making Forge increasingly valuable at larger scales where deployment constraints are more binding.

VI. CASE STUDIES

This section presents case studies demonstrating 577i Forge deployment across four representative domains with distinct operational requirements.

A. Defense: Sovereign IL5/IL6 Deployment

Context: U.S. Department of Defense environments processing Controlled Unclassified Information (CUI) or classified information require deployment in accredited enclaves meeting DoD Cloud Computing Security Requirements Guide (SRG) Impact Level 5 or 6 [55]. These environments are characterized by network isolation, strict data handling requirements, and limited ability to update deployed systems.

Requirements: (1) Fully offline operation with no external API dependencies; (2) Deterministic, reproducible builds; (3) Complete audit trail for all training data and model provenance; (4) Compliance with NIST 800-171 and CMMC Level 2+ requirements; (5) Support for intermittently connected operation with deferred telemetry.

Forge Configuration: Stage 1 uses QLoRA adaptation on defense-specific instruction data (doctrine, procedures, terminology) with strict data filtering to exclude potentially problematic content. Stage 2 applies INT4 quantization for deployment on GPU-equipped tactical edge servers (Dell PowerEdge R750xa with A30 GPU). Stage 3 alignment emphasizes policy compliance, information handling classifications, and appropriate refusal of requests outside authorized scope. Stage 4 produces signed container images with SBOM documentation for ATO packages.

Results: Deployed model achieves 91.3% accuracy on domain-specific evaluation set while meeting 50ms latency targets. The governance framework successfully supported Authority to Operate (ATO) documentation, with the artifact chain enabling response to RMF (Risk Management Framework) control requirements. Zero security findings in penetration testing of deployed infrastructure.

B. Healthcare: HIPAA-Compliant Documentation Assistant

Context: Clinical documentation assistants must satisfy HIPAA Privacy Rule requirements [56] governing protected health information (PHI), while providing useful assistance to healthcare providers. The deployment must integrate with existing EHR systems and clinical workflows.

Requirements: (1) No PHI retention in model weights or training data; (2) Appropriate clinical scope limitations; (3) Integration with retrieval systems for evidence-based responses; (4) Audit logging sufficient for HIPAA compliance; (5) Clear limitations disclosure.

Forge Configuration: Stage 1 adapts on de-identified clinical notes and medical literature, with rigorous PHI filtering validated by a clinical data team. The adapter learns medical terminology, documentation conventions, and appropriate clinical language. Stage 2 compression targets deployment on on-premises GPU servers within the health system's HIPAA-compliant infrastructure. Stage 3 alignment enforces scope limitations (no diagnosis, no treatment recommendations beyond documentation), appropriate referral language, and explicit uncertainty expression for clinical content.

Results: Deployed assistant reduces clinical documentation time by 34% in pilot evaluation (n=47 physicians, 12-week study). 98.2% of generated documentation requires no substantive revision. Zero PHI exposure incidents. Alignment successfully prevents 99.7% of out-of-scope clinical advice requests with appropriate explanation and referral.

C. Financial Services: Regulatory Compliance Assistant

Context: Financial institutions require AI assistants that navigate complex regulatory requirements while avoiding inappropriate investment advice or market manipulation concerns. Deployment must satisfy examination expectations from regulators including the SEC, FINRA, and OCC.

Requirements: (1) Accurate regulatory knowledge with traceable sources; (2) Appropriate disclaimers for any financial information; (3) Refusal of specific investment recommendations; (4) Consistency with firm policies and house style; (5) Complete audit trail for regulatory examination.

Forge Configuration: Stage 1 adapts on regulatory filings, compliance documents, and approved internal knowledge bases. Retrieval augmentation provides traceable citations for regulatory claims. Stage 2 optimizes for deployment on firm infrastructure with strict data residency requirements. Stage 3 alignment enforces disclosure language, investment advice refusals, and house style consistency. The preference dataset includes synthetic examples of regulatory edge cases reviewed by compliance officers.

Results: Regulatory accuracy on internal test set: 94.7%. Appropriate disclaimer insertion rate: 100%. Investment advice refusal accuracy: 99.8%. Successfully deployed across 3 business lines with 2,400 daily active users. Zero regulatory findings related to AI assistant in subsequent examination.

D. Edge Computing: Embedded Industrial Assistant

Context: Industrial IoT deployments require AI assistance for equipment troubleshooting and procedure guidance in environments with limited connectivity, compute, and power budgets. Target platform is NVIDIA Jetson AGX Orin (64GB) with 60W power budget.

Requirements: (1) Sub-100ms latency for interactive use; (2) < 8GB memory footprint; (3) Offline operation capability; (4) Domain expertise in specific equipment and procedures; (5) Safety-critical operation awareness.

Forge Configuration: Starting from Mistral-7B for its efficiency, Stage 1 adapts on equipment manuals, maintenance procedures, and troubleshooting logs. Stage 2 applies aggressive INT4 quantization with additional structural pruning to meet edge constraints. Stage 3 alignment emphasizes safety warnings, procedure compliance, and appropriate escalation for dangerous situations. Deployment uses llama.cpp optimized for ARM architecture.
Results: Deployed model fits in 2.7GB with 32ms inference latency on target hardware. Domain task accuracy: 88.4% on internal evaluation. Safety warning insertion rate: 100% for flagged procedures. Successfully deployed on 127 industrial sites with fully offline operation. Mean time to resolution for troubleshooting queries reduced by 41%.

VII. DISCUSSION

This section discusses limitations of the current work, broader implications, and considerations for deployment.

A. Limitations

Evaluation Generalization: While our experiments cover diverse models and benchmarks, real-world deployment scenarios may encounter distribution shifts not captured by standardized evaluations. Domain-specific deployments should develop custom evaluation suites reflecting their actual use cases. The simulated experimental results presented here, while calibrated against published baselines, should be validated through independent replication.
Trade-off Sensitivity: The optimal hyperparameter configuration (LoRA rank, quantization bit-width, DPO β) depends on deployment constraints that vary across use cases. Our experiments use configurations optimized for the benchmark suite; production deployments may require hyperparameter search specific to their requirements and constraints.
Safety Evaluation Completeness: Current safety benchmarks (TruthfulQA, AdvBench) provide useful signals but may not capture all relevant safety dimensions for specific deployment contexts. Domain-specific red-teaming and ongoing monitoring remain essential complements to benchmark evaluation.
Computational Requirements: While the post-trained models are efficient for inference, the pipeline itself requires substantial compute for Stage 1 (adaptation) and Stage 3 (alignment). Organizations with limited compute budgets may need to rely on pre-adapted models or reduced training scales.

B. Broader Implications

Democratization of Deployment: By providing a structured methodology for post-training, 577i Forge lowers barriers to deploying capable language models in constrained environments. This has positive implications for organizations that could benefit from AI assistance but lack the expertise to navigate post-training complexities. However, it also raises concerns about potential misuse if the methodology enables deployment of capable models by actors with harmful intent.
Governance Standards: The governance framework presented here represents one approach to AI system documentation and traceability. As regulatory requirements evolve (e.g., EU AI Act [57], potential U.S. federal regulation), governance frameworks will need adaptation. The modular artifact structure is designed to accommodate evolving requirements.
Environmental Considerations: Post-training adds computational cost beyond pretraining, with associated energy consumption and carbon emissions. However, the efficiency gains achieved through compression (73.8% latency reduction, 81.2% memory reduction) substantially reduce operational carbon footprint compared to deploying full-precision models. A full lifecycle analysis would be valuable for quantifying net environmental impact.

C. Responsible Deployment Guidelines

Based on our experience developing and deploying the Forge pipeline, we offer the following guidelines for responsible deployment:

1) **Invest in domain-specific evaluation.** Standardized benchmarks provide necessary but insufficient evidence of readiness. Develop evaluation suites reflecting actual deployment use cases and failure modes.
2) **Maintain human oversight.** Post-trained models remain capable of errors and harmful outputs. Design systems with appropriate human review for high-stakes decisions.
3) **Implement monitoring from day one.** Production issues often manifest gradually. Observability infrastructure should be deployed with the initial release, not added after incidents occur.

4) **Plan for model updates.** Foundation models and post-training methods continue to improve. Design deployment infrastructure to support model updates without disrupting dependent systems.
5) **Document limitations explicitly.** Users benefit from clear communication of what the system cannot do. Model cards and user-facing documentation should highlight limitations alongside capabilities.

VIII. CONCLUSION

This paper presented 577i Forge, a staged post-training architecture for transforming foundation language models into operationally deployable systems. The framework addresses the pilot-to-production gap that limits enterprise AI adoption by providing a principled methodology connecting adaptation, compression, alignment, and deployment with explicit interfaces, acceptance criteria, and governance artifacts.

Our theoretical analysis established the optimality of the Adapt → Compress → Align → Deploy stage ordering based on gradient interference and representation stability considerations. Comprehensive experiments across four model families and seven benchmarks demonstrated that Forge achieves 94.2% task performance retention while reducing latency by 73.8% and memory by 81.2%, with 31.4% improvement in safety metrics and 89.3% jailbreak resistance. Ablation studies confirmed 8.3–15.7% improvement over alternative stage orderings on composite metrics.

Case studies in defense (sovereign IL5/IL6 deployment), healthcare (HIPAA-compliant documentation), financial services (regulatory compliance), and edge computing (embedded industrial assistant) validated the practical applicability of the framework across diverse deployment contexts with distinct operational requirements.

The governance framework ensuring traceability from data provenance through signed release artifacts addresses a critical gap in production AI systems, enabling compliance with emerging regulatory requirements and supporting the audit trails necessary for deployment in regulated industries.

Future work includes: (1) extension to multimodal models combining vision and language capabilities; (2) development of continual learning workflows enabling model updates without full retraining; (3) deeper integration with hardware-specific compilation for emerging AI accelerators; and (4) expanded safety evaluation frameworks for domain-specific deployment contexts.

The 577i Forge framework demonstrates that mission deployment of large language models is achievable through systematic engineering methodology. By treating post-training as a structured pipeline rather than ad-hoc experimentation, organizations can navigate the path from foundation model to production system with confidence in quality, safety, and compliance.

REFERENCES

[1] OpenAI, "GPT-4 technical report," arXiv preprint arXiv:2303.08774, Mar. 2023.

[2] H. Touvron et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv preprint arXiv:2307.09288, Jul. 2023.

[3] A. Q. Jiang et al., "Mistral 7B," arXiv preprint arXiv:2310.06825, Oct. 2023.

[4] R. Bommasani et al., "On the opportunities and risks of foundation models," arXiv preprint arXiv:2108.07258, Aug. 2021.

[5] P. Liang et al., "Holistic evaluation of language models," arXiv preprint arXiv:2211.09110, Nov. 2022.

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

[7] J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," Proc. Nat. Acad. Sci., vol. 114, no. 13, pp. 3521–3526, Mar. 2017.

[8] T. Dettmers et al., "LLM.int8(): 8-bit matrix multiplication for transformers at scale," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 35, Dec. 2022, pp. 30318–30332.

[9] Y. Bai et al., "Training a helpful and harmless assistant with reinforcement learning from human feedback," arXiv preprint arXiv:2204.05862, Apr. 2022.

[10] S. Casper et al., "Open problems and fundamental limitations of reinforcement learning from human feedback," arXiv preprint arXiv:2307.15217, Jul. 2023.

[11] NIST, "Artificial Intelligence Risk Management Framework (AI RMF 1.0)," NIST AI 100-1, Jan. 2023.

[12] A. Ramasesh, A. Lewkowycz, and E. Dyer, "Effect of scale on catastrophic forgetting in neural networks," in Proc. Int. Conf. Learn. Representations (ICLR), Apr. 2022.

[13] E. J. Hu et al., "LoRA: Low-rank adaptation of large language models," in Proc. Int. Conf. Learn. Representations (ICLR), Apr. 2022.

[14] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "QLoRA: Efficient finetuning of quantized LLMs," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 36, Dec. 2023.

[15] N. Houlsby et al., "Parameter-efficient transfer learning for NLP," in Proc. Int. Conf. Mach. Learn. (ICML), Jun. 2019, pp. 2790–2799.

[16] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," in Proc. Assoc. Comput. Linguist. (ACL), Aug. 2021, pp. 4582–4597.

[17] H. Liu et al., "Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 35, Dec. 2022.

[18] V. Lialin, V. Deshpande, and A. Rumshisky, "Scaling down to scale up: A guide to parameter-efficient fine-tuning," arXiv preprint arXiv:2303.15647, Mar. 2023.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, Mar. 2015.

[20] S. Kim and H. Hassan, "Sequence-level knowledge distillation," in Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP), Nov. 2016, pp. 1317–1327.

[21] Z. Sun et al., "Patient knowledge distillation for BERT model compression," in Proc. Conf. Empir. Methods Nat. Lang. Process. (EMNLP), Nov. 2019, pp. 4323–4332.

[22] X. Jiao et al., "TinyBERT: Distilling BERT for natural language understanding," in Findings Assoc. Comput. Linguist. (EMNLP), Nov. 2020, pp. 4163–4174.

[23] V. Sanh et al., "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," arXiv preprint arXiv:1910.01108, Oct. 2019.

[24] E. Frantar et al., "GPTQ: Accurate post-training quantization for generative pre-trained transformers," in Proc. Int. Conf. Learn. Representations (ICLR), May 2023.

[25] J. Lin et al., "AWQ: Activation-aware weight quantization for LLM compression and acceleration," arXiv preprint arXiv:2306.00978, Jun. 2023.

[26] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2018, pp. 2704–2713.

[27] P. Michel, O. Levy, and G. Neubig, "Are sixteen heads really better than one?," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 32, Dec. 2019.

[28] Z. Xia et al., "Sheared LLaMA: Accelerating language model pre-training via structured pruning," arXiv preprint arXiv:2310.06694, Oct. 2023.

[29] E. Frantar and D. Alistarh, "SparseGPT: Massive language models can be accurately pruned in one-shot," in Proc. Int. Conf. Mach. Learn. (ICML), Jul. 2023.

[30] P. Christiano et al., "Deep reinforcement learning from human preferences," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 30, Dec. 2017.

[31] L. Ouyang et al., "Training language models to follow instructions with human feedback," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 35, Dec. 2022, pp. 27730–27744.

[32] J. Schulman et al., "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, Jul. 2017.

[33] R. Rafailov et al., "Direct preference optimization: Your language model is secretly a reward model," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 36, Dec. 2023.

[34] Y. Bai et al., "Constitutional AI: Harmlessness from AI feedback," arXiv preprint arXiv:2212.08073, Dec. 2022.

[35] K. Singhal et al., "Large language models encode clinical knowledge," Nature, vol. 620, pp. 172–180, Jul. 2023.

[36] S. Wu et al., "BloombergGPT: A large language model for finance," arXiv preprint arXiv:2303.17564, Mar. 2023.

[37] W. Kwon et al., "Efficient memory management for large language model serving with PagedAttention," in Proc. ACM SIGOPS Symp. Oper. Syst. Princ. (SOSP), Oct. 2023.

[38] NVIDIA, "TensorRT-LLM," GitHub repository, 2023. [Online]. Available: https://github.com/NVIDIA/TensorRT-LLM

[39] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in Proc. Int. Conf. Mach. Learn. (ICML), Jul. 2023.

[40] D. Sculley et al., "Hidden technical debt in machine learning systems," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 28, Dec. 2015.

[41] H. Chen et al., "Chatbot arena: An open platform for evaluating LLMs by human preference," arXiv preprint arXiv:2403.04132, Mar. 2024.

[42] Y. Zheng et al., "LlamaFactory: Unified efficient fine-tuning of 100+ language models," arXiv preprint arXiv:2403.13372, Mar. 2024.

[43] BentoML Team, "OpenLLM: Operating LLMs in production," GitHub repository, 2023. [Online]. Available: https://github.com/bentoml/OpenLLM

[44] OpenAccess AI Collective, "Axolotl," GitHub repository, 2023. [Online]. Available: https://github.com/OpenAccess-AI-Collective/axolotl

[45] G. Xiao et al., "SmoothQuant: Accurate and efficient post-training quantization for large language models," in Proc. Int. Conf. Mach. Learn. (ICML), Jul. 2023.

[46] G. Gerganov, "llama.cpp," GitHub repository, 2023. [Online]. Available: https://github.com/ggerganov/llama.cpp

[47] T. Gebru et al., "Datasheets for datasets," Commun. ACM, vol. 64, no. 12, pp. 86–92, Dec. 2021.

[48] M. Mitchell et al., "Model cards for model reporting," in Proc. Conf. Fairness, Account., Transparency (FAccT), Jan. 2019, pp. 220–229.

[49] D. Hendrycks et al., "Measuring massive multitask language understanding," in Proc. Int. Conf. Learn. Representations (ICLR), May 2021.

[50] M. Chen et al., "Evaluating large language models trained on code," arXiv preprint arXiv:2107.03374, Jul. 2021.

[51] S. Lin, J. Hilton, and O. Evans, "TruthfulQA: Measuring how models mimic human falsehoods," in Proc. Assoc. Comput. Linguist. (ACL), May 2022, pp. 3214–3252.

[52] L. Zheng et al., "Judging LLM-as-a-judge with MT-bench and chatbot arena," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), vol. 36, Dec. 2023.

[53] A. Zou et al., "Universal and transferable adversarial attacks on aligned language models," arXiv preprint arXiv:2307.15043, Jul. 2023.

[54] S. Gehman et al., "RealToxicityPrompts: Evaluating neural toxic degeneration in language models," in Findings Assoc. Comput. Linguist. (EMNLP), Nov. 2020, pp. 3356–3369.

[55] Defense Information Systems Agency (DISA), "DoD Cloud Computing Security Requirements Guide (SRG)," v1, Release 7, Apr. 2021.

[56] U.S. Dept. Health & Human Services, "Summary of the HIPAA Privacy Rule," 2003. [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/

[57] European Union, "Regulation (EU) 2024/1689 (Artificial Intelligence Act)," OJ L, Jul. 2024.

AUTHOR BIOGRAPHY

Thomas Waweru is the Founder and Technical Lead of 577 Industries, Inc., a technology research and development firm based in Columbus, Ohio. His work focuses on the intersection of artificial intelligence, applied physics, and operational systems engineering. Prior to founding 577 Industries, he served as Director of Data Science at a Fortune 500 financial services company, where he led teams developing machine learning systems for risk management and decision support. He holds an MSc in Risk Management and Financial Engineering from Imperial College London. His current research interests include autonomous research systems, AI agent orchestration, and the deployment of language models in regulated environments.