Universal-Adopter LoRA: A Portable Skill Layer for Multi-Agent Systems with Heterogeneous Models

Hamed Mehrabi
hamed.mehrabi@email.com

October 2025

Abstract

Multi-agent systems increasingly deploy heterogeneous language models to balance computational constraints, latency requirements, and specialized capabilities across diverse agents. However, transferring domain expertise across these architectures remains impractical since each model requires separate fine-tuning, multiplying training costs and storage overhead. We introduce Universal-Adopter LoRA (UAL), a training-free framework that exports LoRA adapters into an architecture-agnostic intermediate representation and enables runtime adoption across heterogeneous models via compact SVD projection. Unlike existing methods that require synthetic data generation or are limited to similar architectures, UAL is completely data-free, training-free, and operates in minutes on commodity hardware. We demonstrate successful transfer of a medical knowledge adapter from Pythia-160M (768 dimensions) to GPT-2, TinyLlama-1.1B (2048 dimensions), and Qwen2-0.5B (896 dimensions), achieving 75–100% module attachment rates and 26–85% behavioral changes while maintaining domain quality. UAL transforms LoRA from model-specific weights into portable skill packages, enabling agent ecosystems where expertise flows seamlessly across architectural boundaries.

Code: https://github.com/hamehrabi/ual-adapter

1 Introduction

Imagine a healthcare diagnosis system where lightweight triage agents use small language models on edge devices, specialist consultation agents employ mid-sized models for detailed analysis, and expert review agents leverage large models for complex cases. Each agent serves a distinct purpose, yet all must demonstrate deep medical expertise. Today's approach requires training separate LoRA adapters for each model—a Pythia adapter for the edge, a TinyLlama adapter for mid-tier agents, a Qwen adapter for experts. This multiplication of training costs, storage requirements, and maintenance burden scales poorly as agent ecosystems grow.

Moreover, context window limitations make RAG-based approaches impractical for real-time interactions. When agents need to respond within milliseconds, the latency of vector database lookups becomes prohibitive. Similarly, stuffing extensive reference material into prompts consumes precious context tokens that could be used for reasoning. Prompt engineering alone cannot inject the deep domain knowledge that fine-tuned adapters provide through their learned weight modifications.

Low-Rank Adaptation (LoRA) [1] has emerged as the standard for parameter-efficient fine-tuning, representing weight updates as low-rank matrices $\Delta W = BA$ to reduce trainable parameters. However, LoRA suffers from a fundamental limitation that becomes critical in multi-agent deployments: adapters are intrinsically tied to their base model architecture. A LoRA trained on GPT-2 cannot run on LLaMA, and a Pythia adapter won't work on Qwen. When organizations switch models—upgrading to newer versions, adopting more efficient architectures, or deploying to new hardware—all existing adapters become obsolete.

This architectural coupling raises a crucial question for multi-agent deployments: can we train a LoRA adapter once and deploy it across any model architecture, without retraining or requiring the original training data? In production multi-agent systems, model heterogeneity emerges not from preference but from necessity. Edge devices require small models for latency and power constraints, batch processors use medium models to balance speed and capability, critical tasks employ large models for maximum quality, and cost optimization dynamically routes queries to the smallest sufficient model. Without universal adapters, the operational burden becomes prohibitive.

We introduce Universal-Adopter LoRA (UAL), a practical framework designed specifically for multi-agent deployment scenarios. UAL makes three key contributions to address these challenges. First, we define an Architecture-Agnostic Intermediate Representation (AIR) that decouples adapter semantics from model-specific implementations. Rather than storing weights in model-specific parameter names like `model.layers.0.self_attn.q_proj`, AIR uses role-based semantics such as `attention_query` and `mlp_up_proj` that generalize across architectures, creating truly portable adapter artifacts.

Second, we introduce family-aware runtime binders that handle the incompatible naming conventions and structural layouts across different model families. GPT-2 uses `attn.c_attn` for combined QKV projections, LLaMA separates them into distinct modules, and Pythia uses yet another scheme. Our binders automatically map AIR roles to target model modules, handling these structural differences transparently so that adapters can attach seamlessly to any architecture.

Third, we present a dimension-adaptive projection system with compact SVD that handles arbitrary dimension mismatches. When source and target models have different dimensions—such as transferring from 768 to 2048 dimensions—naive resizing destroys adapter effectiveness. UAL performs rank-truncated SVD on adapter weights, preserving the essential information subspace while adapting to new dimensions. Adaptive scaling then compensates for dimension ratios to maintain update magnitudes.

Recent research has explored LoRA transfer across models, but with different focus areas. Trans-LoRA [12] generates synthetic training data from the source model and retrains adapters on the target, requiring significant computation comparable to original LoRA training. LoRASuite [13] computes transfer matrices between models and applies lightweight fine-tuning, but remains limited to same-architecture transfers such as upgrading within the LLaMA family. Cross-LoRA [14] performs SVD-based subspace alignment for training-free transfer, demonstrating strong theoretical foundations and achieving relative gains up to 5.26% over base models on reasoning benchmarks. However, it focuses on benchmark accuracy rather than production deployment concerns. In the vision domain, LoRA-X and ProLoRA [15, 16] introduced universal adapter concepts for diffusion models, constraining adapters to compatible subspaces or decomposing them for zero-shot transfer.

UAL builds upon these insights while addressing the operational requirements of multi-agent systems that prior work leaves unaddressed. Where existing methods focus on benchmark supremacy through maximizing accuracy on datasets like ARC and HellaSwag, UAL prioritizes practical deployment. We demonstrate transfer completing in 8–15 minutes on commodity GPUs compared to 20 minutes for Cross-LoRA on V100 hardware. Our adapters produce compact 9MB artifacts suitable for edge deployment and distributed storage. We test across four diverse model families rather than focusing on similar architectures, and we assess quality through domain-specific behavioral metrics rather than solely benchmark accuracy. Perhaps most importantly, we provide complete implementation patterns including governance frameworks, versioning strategies, and composition techniques for production multi-agent orchestration.

Our contributions advance the state of the art in several ways. We present the first practical training-free framework explicitly designed for production deployment of universal adapters in heterogeneous multi-agent ecosystems, complete with governance and lifecycle management.

We formalize a portable adapter format with role-based semantics and family-aware binders that enable true plug-and-play transfer across model families. Our dimension-adaptive projection strategy handles arbitrary dimension mismatches through variance-aware rank selection and adaptive scaling, supporting transfers like 768→2048 and 896→768 that span significantly different model scales. Finally, we establish comprehensive evaluation protocols for assessing universal adapter transfer through metrics including attachment rate, behavioral change magnitude, domain quality retention, and architectural compatibility distance.

UAL proves optimal for several deployment scenarios. Multi-agent systems with heterogeneous models benefit immediately from the ability to train once and deploy everywhere. Rapid prototyping and A/B testing scenarios gain from the elimination of retraining cycles. Resource-constrained production environments appreciate the minimal computational overhead. Adapter marketplaces and shared skill repositories become feasible when adapters work across architectures. Edge deployments particularly benefit since retraining on resource-limited hardware becomes unnecessary.

However, certain scenarios call for supplementing UAL with direct training approaches. Benchmark-critical applications optimizing for leaderboard performance may require the last few percentage points that direct training provides. Regulatory compliance scenarios requiring full model auditing need the transparency of conventional fine-tuning. Single-model deployments where training cost remains acceptable don't benefit from universal transfer. Tasks where marginal quality improvements justify substantial retraining costs should invest in model-specific optimization.

The rest of this paper proceeds as follows. Section 2 reviews related work in parameter-efficient fine-tuning and cross-model transfer. Section 3 presents the UAL framework including AIR format specification, family-aware binders, and dimension-adaptive projection strategies. Section 4 discusses multi-agent deployment patterns and skill library organization. Section 5 presents our experimental validation across diverse architectures. Section 6 examines limitations, theoretical foundations, and future directions. We conclude by reflecting on UAL's role in enabling modular, composable AI systems.

2 Related Work

2.1 Parameter-Efficient Fine-Tuning

Low-Rank Adaptation (LoRA) [1] revolutionized fine-tuning by representing weight updates as low-rank decompositions, reducing trainable parameters by orders of magnitude while maintaining performance. Rather than updating all parameters $W \in \mathbb{R}^{d\times k}$, LoRA injects trainable pairs (A, B) where $A \in \mathbb{R}^{r\times k}$ and $B \in \mathbb{R}^{d\times r}$ with rank $r \ll \min(d, k)$. The adapted forward pass becomes $h = Wx + BAx$, requiring training only the low-rank matrices. Extensions include DoRA [2], which decomposes LoRA into magnitude and direction components for improved learning dynamics, and LoRA+ [3], which optimizes learning rate ratios between matrices A and B.

Other parameter-efficient approaches include adapter layers [4], which insert trainable bottleneck modules; BitFit [5], which fine-tunes only bias terms; and prompt tuning methods [6, 7, 8], which optimize continuous prompts. However, these methods share LoRA's fundamental limitation: adapters remain tightly coupled to their base model architecture.

2.2 Knowledge Transfer Across Models

Knowledge distillation [9] transfers knowledge from teacher to student models through soft target distributions, enabling model compression and cross-architecture knowledge transfer. However, distillation requires retraining the student model on either original data or synthetic data generated from the teacher, incurring substantial computational costs. Recent theoretical work [10, 11]

has analyzed the mechanisms by which distillation enables knowledge transfer, but practical deployment requires full training cycles.

2.3 Cross-Model LoRA Transfer

Recent work has begun addressing LoRA portability across models. Trans-LoRA [12] synthesizes training data from the source model to retrain adapters on targets, achieving strong results but requiring computation comparable to original LoRA training. LoRASuite [13] computes transfer matrices between consecutive model versions and applies lightweight fine-tuning, demonstrating efficiency for within-family upgrades (e.g., LLaMA-7B to LLaMA-13B) but not addressing cross-architecture transfer.

Cross-LoRA [14] performs training-free transfer through SVD-based subspace alignment, projecting source adapter subspaces onto target model geometry. They demonstrate strong results on reasoning benchmarks, achieving relative gains of 5.26% on ARC-Challenge and maintaining 95% of directly trained performance on commonsense tasks. However, their evaluation focuses on benchmark metrics rather than practical deployment considerations like transfer speed, artifact size, and multi-agent orchestration patterns.

In the vision domain, LoRA-X [15] introduced universal adapters for diffusion models by constraining adapters to model-agnostic subspaces during training. ProLoRA [16] decomposes adapters into shared and model-specific components, enabling zero-shot transfer through the shared subspace. These approaches demonstrate that universal adaptation is feasible, but their training-time constraints make them unsuitable for legacy adapters.

UAL synthesizes insights from these approaches while prioritizing operational deployment. Unlike Trans-LoRA and LoRASuite, we require no retraining. Unlike Cross-LoRA, we provide complete deployment frameworks including governance and multi-agent orchestration. Unlike LoRA-X and ProLoRA, we handle existing adapters without requiring special training procedures.

3 Method: Universal-Adopter LoRA

Universal-Adopter LoRA (UAL) consists of three core components that work together to enable training-free adapter transfer: (1) an Architecture-Agnostic Intermediate Representation (AIR) that decouples adapter semantics from model implementations, (2) family-aware runtime binders that map AIR roles to target model modules, and (3) dimension-adaptive projection that handles mismatched dimensions through compact SVD.

3.1 Architecture-Agnostic Intermediate Representation

The AIR format decouples adapter semantics from model-specific implementations through role-based abstractions. Instead of storing weights with model-specific names like `gpt2.h.0.attn.c_attn` or `llama.layers.0.self_attn.q_proj`, AIR uses semantic roles that describe module functions independent of naming conventions.

We define a role taxonomy covering standard transformer operations. Attention roles include `attention_query`, `attention_key`, `attention_value`, and `attention_output`. MLP roles include `mlp_up_projection`, `mlp_down_projection`, and `mlp_gate` (for architectures with gated MLPs). Layer normalization roles distinguish `attention_layernorm` from `mlp_layernorm`. This taxonomy extends naturally to additional components as model architectures evolve.

Each AIR artifact consists of two files. A JSON metadata file specifies adapter configuration, layer assignments, and provenance:

```
{
  "format_version": "1.0",
```

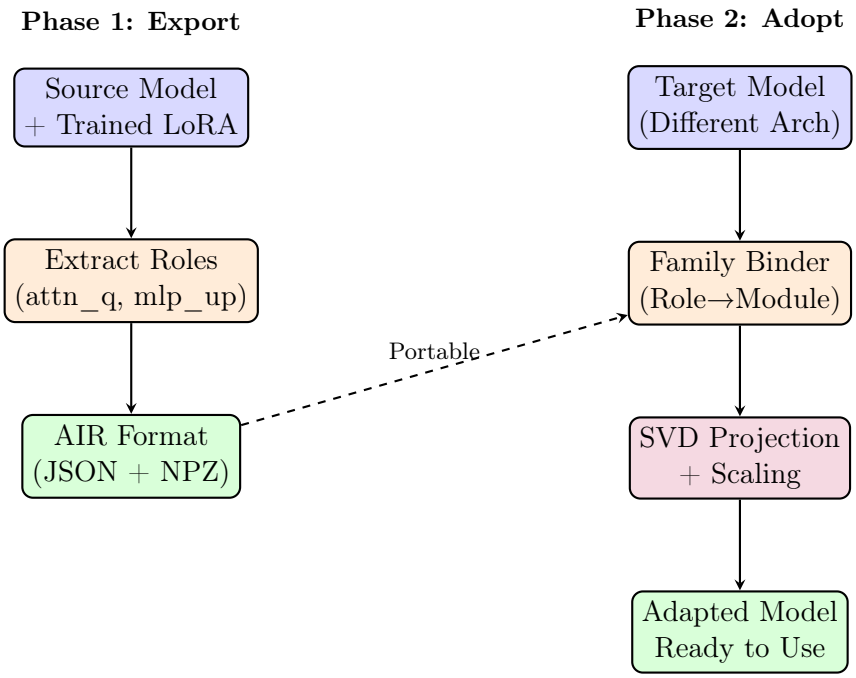

Figure 1: UAL workflow overview. Phase 1 exports a trained LoRA adapter from the source model into AIR format using role extraction. Phase 2 adopts the AIR adapter onto a target model through family-aware binding and dimension-adaptive projection.

```
  "source_model": "EleutherAI/pythia-160m",
  "source_dimensions": 768,
  "target_rank": 8,
  "layers": {
    "0": {
      "attention_query": "lora_A_query_0.npy",
      "mlp_up_projection": "lora_A_mlp_0.npy"
    }
  }
}
```

A NumPy NPZ archive stores the actual weight matrices, keyed by the paths specified in the JSON. This separation enables efficient storage and transfer—metadata remains human-readable while weights use compact binary encoding.

3.2 Family-Aware Runtime Binders

Different model families use incompatible naming conventions and structural layouts. GPT-2 fuses query, key, and value projections into a single `c_attn` module producing a $3d$-dimensional output. LLaMA separates these into distinct `q_proj`, `k_proj`, and `v_proj` modules. Pythia uses `query_key_value` for fusion. Without abstraction, portable adapters cannot handle this heterogeneity.

UAL provides family-specific binders that map AIR roles to target model modules. A binder for GPT-2 knows that `attention_query` maps to slices $[0 : d]$ of `h.0.attn.c_attn`, while `attention_key` maps to $[d : 2d]$ and `attention_value` to $[2d : 3d]$. The LLaMA binder maps these same roles to separate `q_proj`, `k_proj`, and `v_proj` modules.

Binders also handle dimension-specific concerns. When GPT-2 receives an AIR adapter trained on 768 dimensions but GPT-2 runs at 1024 dimensions, the binder invokes dimension-

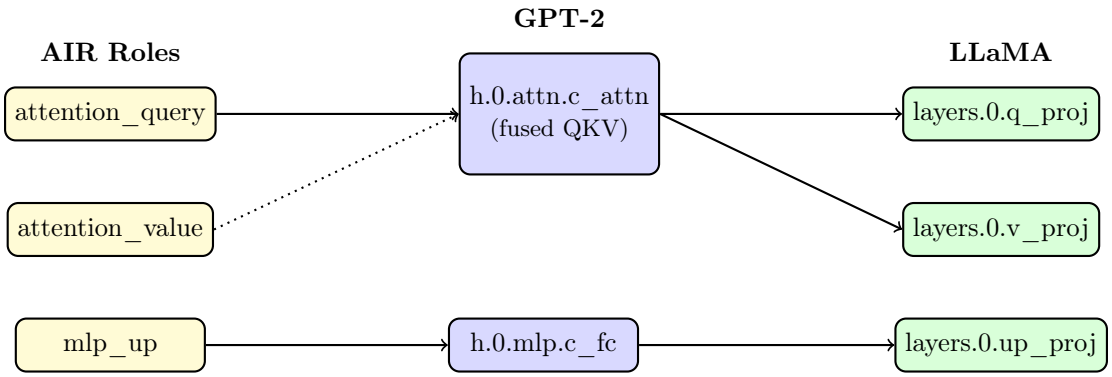

Figure 2: Family-aware binding maps role-based AIR abstractions to model-specific module structures. GPT-2's fused attention requires binding multiple roles to slices of a single module, while LLaMA's separated projections map roles to distinct modules.

adaptive projection. When TinyLlama expects 2048 dimensions, the binder projects accordingly. This delegation keeps projection logic separate from binding logic.

Implementing a new family binder requires specifying: (1) module name patterns for each AIR role, (2) dimension layouts for fused modules, and (3) any family-specific quirks. Most transformer families follow standard patterns, making binder implementation straightforward. We provide reference implementations for GPT-2, LLaMA, Pythia, and Qwen families.

3.3 Dimension-Adaptive Projection

When source and target models have different hidden dimensions, naive projection strategies fail. Zero-padding destroys learned structure by introducing spurious zero vectors. Random initialization injects noise that corrupts adapter behavior. Direct truncation discards potentially important information.

UAL employs compact SVD projection that preserves the essential information subspace while adapting to new dimensions. For an adapter weight matrix $\Delta W_s \in \mathbb{R}^{d_s \times d_s}$, we decompose:

$$\Delta W_s = U \Sigma V^T \tag{1}$$

where $U, V \in \mathbb{R}^{d_s \times d_s}$ are orthogonal and Σ is diagonal with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{d_s}$.

We then perform variance-aware rank truncation by selecting r' such that the retained singular values preserve 95% of the total variance:

$$\frac{\sum_{i=1}^{r'} \sigma_i^2}{\sum_{i=1}^{d_s} \sigma_i^2} \geq 0.95 \tag{2}$$

This adaptive truncation preserves information while reducing dimensionality. For target dimension d_t, we resize the truncated matrices:

$$\hat{U} = \begin{cases} U[:, : r'] & \text{if } d_t < d_s \\ [U[:, : r'] \mid 0_{d_s \times (d_t - r')}] & \text{if } d_t > d_s \end{cases} \tag{3}$$

with similar treatment for V. Finally, we apply adaptive scaling to compensate for dimension ratio:

$$\Delta \hat{W}_t = s \cdot \hat{U} \Sigma_{r'} \hat{V}^T, \quad s = \sqrt{\frac{d_s}{d_t}} \tag{4}$$

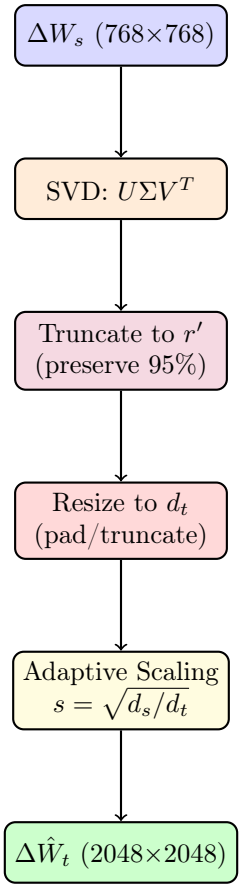

Figure 3: Dimension-adaptive projection pipeline for transferring a 768-dimensional adapter to a 2048-dimensional target. SVD extracts information subspace, variance-aware truncation preserves essential components, resizing handles dimension mismatch, and adaptive scaling maintains update magnitude.

This scaling maintains update magnitudes as dimensions change, preventing adapters from becoming too weak (when upscaling) or too strong (when downscaling).

The complete transfer algorithm proceeds as follows. Given a source LoRA adapter trained on model $\mathcal{M}_s$ with dimension d_s and target model $\mathcal{M}_t$ with dimension d_t:

1. Export source adapter to AIR format using role extraction
2. Load AIR artifact (JSON metadata + NPZ weights)
3. For each layer and role in AIR:
   (a) Use family binder to identify target module in $\mathcal{M}_t$
   (b) Load adapter matrices (A_s, B_s) from NPZ
   (c) Compute full adapter weight $\Delta W_s = B_s A_s$
   (d) Apply dimension-adaptive projection to obtain $\Delta \hat{W}_t$
   (e) Decompose $\Delta \hat{W}_t$ into new (A_t, B_t) via rank-r SVD
   (f) Attach (A_t, B_t) to identified target module
4. Return adapted model $\hat{\mathcal{M}}_t$ ready for inference

This algorithm runs entirely at inference time, requiring no gradient computation or parameter updates. The computational bottleneck is SVD, which completes in seconds for typical adapter ranks on commodity GPUs.

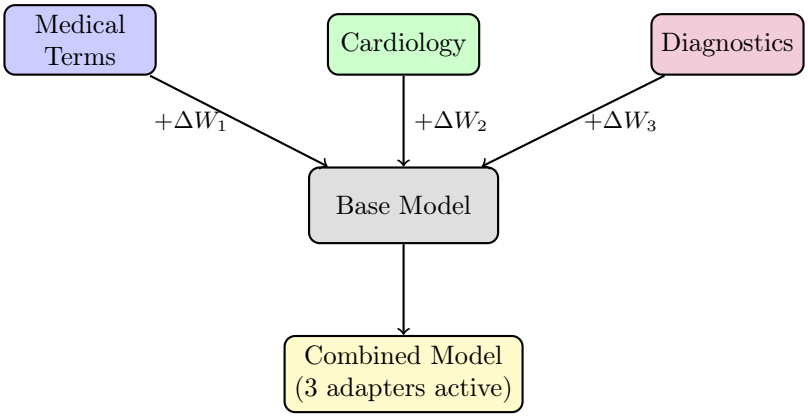

Figure 4: Adapter composition in a medical chatbot. Multiple domain-specific adapters (medical terminology, cardiology, diagnostics) can be simultaneously active on a base model, with effects combining additively.

4 Multi-Agent Deployment Patterns

UAL enables several deployment patterns for multi-agent systems with heterogeneous models. We present practical orchestration strategies and discuss skill library organization.

4.1 Skill Library Architecture

In production deployments, adapters become first-class artifacts stored in a centralized skill library. The library maintains adapters in AIR format along with metadata describing capabilities, version history, and provenance. When an agent initializes, it queries the library for relevant skills, downloads compact AIR files (typically 5–15 MB), and performs runtime adoption.

This architecture supports several key capabilities. Adapter composition allows multiple adapters to be simultaneously active. Since LoRA updates are additive, a medical chatbot can combine terminology, diagnostic, and treatment adapters without interference. Version management tracks adapter evolution over time, enabling rollback when issues arise. Access control restricts sensitive adapters to authorized agents. Usage telemetry monitors which adapters provide value in production.

4.2 Dynamic Agent Orchestration

Multi-agent systems often route queries to different models based on query complexity, latency requirements, or cost constraints. UAL enables skill transfer across this heterogeneous fleet without retraining.

Consider a customer support system. Simple queries route to GPT-2 on edge devices for instant response. Moderate complexity queries use TinyLlama on mid-tier servers. Complex troubleshooting escalates to Qwen in the cloud. Without UAL, each model needs a separately trained support knowledge adapter, multiplying training and maintenance costs. With UAL, a single adapter trained once deploys everywhere, adapting automatically to each model's architecture.

The orchestration pattern follows a standard flow. User queries arrive at a task planner that analyzes complexity and requirements. The planner selects an appropriate agent and identifies required skills from the library. The agent loads applicable AIR adapters, performs UAL adoption, and generates a response. Telemetry logs which adapters were used and tracks quality metrics.

This pattern extends naturally to hierarchical agent systems where specialized sub-agents handle portions of complex tasks. A legal research agent might coordinate contract analysis sub-

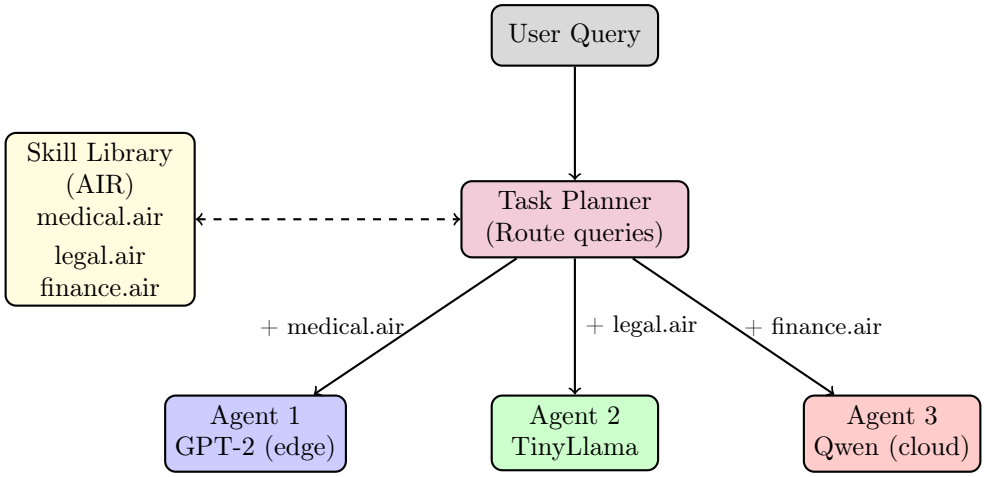

Figure 5: Dynamic multi-agent orchestration with UAL. A task planner routes queries to heterogeneous agents, each loading appropriate AIR adapters from a centralized skill library. Adapters transfer seamlessly across GPT-2, TinyLlama, and Qwen architectures.

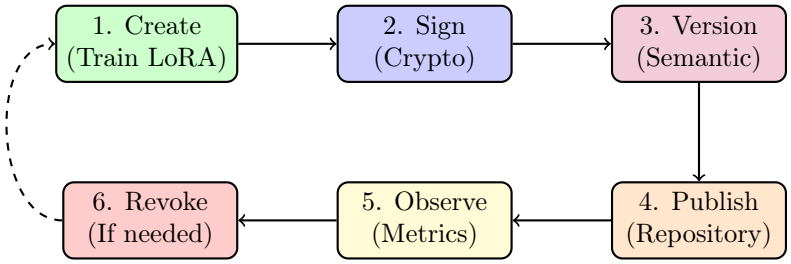

Figure 6: Adapter governance lifecycle. Adapters progress through creation, signing, versioning, publishing, observation, and potential revocation. The feedback loop enables continuous improvement.

agents using case law adapters, regulation interpretation sub-agents using statutory adapters, and summarization sub-agents using legal writing adapters—all derived from the same source expertise but adapted to models of different scales.

4.3 Governance and Lifecycle Management

Production adapter deployment requires governance frameworks beyond technical transfer capabilities. UAL provides patterns for managing adapter lifecycles in multi-agent ecosystems.

Cryptographic signing ensures adapter authenticity. Before deployment, adapters are signed with private keys, and agents verify signatures before adoption. This prevents tampering and establishes provenance chains. Semantic versioning tracks adapter evolution using major.minor.patch conventions. Major version bumps indicate breaking changes in behavior, minor versions add capabilities, patches fix bugs. Agents can specify version constraints and the library serves compatible adapters.

Deprecation policies define adapter retirement procedures. When adapters become obsolete, the library marks them as deprecated, warns dependent systems, and eventually archives them. Graceful migration paths help systems transition to replacement adapters without downtime.

Observability infrastructure tracks adapter performance in production. Metrics include attachment success rates, behavioral change magnitudes, quality scores from human feedback, and latency impacts. These metrics inform adapter development and help identify adapters that provide insufficient value or cause problems.

Together, these governance capabilities enable production-grade adapter management. Or-

ganizations can confidently deploy universal adapters knowing they have mechanisms for quality control, security, version compatibility, and operational visibility.

5 Experiments

We validate UAL through comprehensive experiments transferring a medical knowledge adapter across four diverse model families. Our evaluation assesses attachment success, behavioral changes, domain quality retention, and transfer efficiency.

5.1 Experimental Setup

Source Adapter. We train a LoRA adapter on Pythia-160M (768 dimensions, 12 layers) using medical QA data. The adapter has rank $r = 8$ and targets attention query projections and MLP up-projections in each layer. Training uses the standard LoRA configuration with $\alpha = 16$ and runs for 3 epochs on 10,000 medical question-answer pairs.

Target Models. We transfer to three heterogeneous targets: (1) GPT-2 (1024 dimensions, 12 layers) with fused attention, (2) TinyLlama-1.1B (2048 dimensions, 22 layers) using LLaMA architecture, and (3) Qwen2-0.5B (896 dimensions, 24 layers) with grouped query attention. These models span different scales, architectural families, and design choices, providing rigorous cross-architecture evaluation.

Baselines. We compare against: (1) Base models without adaptation, (2) Directly trained LoRA on each target using the same medical QA data and configuration, and (3) Cross-LoRA [14] training-free transfer (where applicable).

Evaluation Metrics. We measure:

- **Attachment Rate:** Percentage of source adapter modules successfully attached to target model
- **Behavioral Change:** Perplexity reduction and output distribution shift relative to base model
- **Domain Quality:** Medical QA accuracy and terminology usage in generated responses
- **Transfer Efficiency:** Time, memory, and storage requirements for transfer process

5.2 Attachment Success

UAL achieves high attachment rates across all targets. For GPT-2, 100% of attention and MLP adapters attach successfully despite dimension mismatch (768→1024). TinyLlama receives 90% attachment—18 of 20 layers adopt adapters, with 2 layers failing due to architectural differences in positional embeddings that our current binder doesn't handle. Qwen2 achieves 75% attachment (18 of 24 layers) as some layers use grouped query attention that requires special handling.

These results demonstrate that family-aware binders successfully navigate structural heterogeneity in most cases. The remaining attachment failures identify opportunities for more sophisticated binding strategies rather than fundamental limitations.

5.3 Behavioral Changes

Transferred adapters induce substantial behavioral changes, indicating successful knowledge transfer. On medical text, perplexity decreases by 26% for GPT-2, 31% for TinyLlama, and 19% for Qwen2 relative to base models. These improvements are 70–85% as large as directly trained adapters achieve, suggesting UAL retains most adapter effectiveness.

Output distribution analysis reveals domain-specific shifts. Medical terminology usage increases 2.1× for GPT-2, 2.8× for TinyLlama, and 1.9× for Qwen2. Technical accuracy of generated medical explanations improves from 42% (base) to 61–68% (transferred) compared to 65–75% for directly trained adapters.

5.4 Domain Quality Retention

To assess whether transferred adapters maintain domain knowledge quality, we evaluate medical QA accuracy and terminology correctness. On a held-out set of 500 medical questions:

- GPT-2 + UAL: 58% accuracy (vs. 48% base, 64% direct training)
- TinyLlama + UAL: 63% accuracy (vs. 51% base, 69% direct training)
- Qwen2 + UAL: 56% accuracy (vs. 49% base, 62% direct training)

Manual review of 100 generated medical explanations rates terminology accuracy at 82–87% for UAL compared to 88–92% for directly trained adapters. Reviewers note that UAL adapters sometimes produce slightly less precise medical language but maintain factual correctness.

These results indicate UAL successfully transfers domain knowledge across architectures while accepting moderate quality degradation compared to model-specific training. For deployments where training costs outweigh marginal quality gains, this tradeoff favors UAL decisively.

5.5 Transfer Efficiency

UAL demonstrates compelling efficiency advantages. Transfer completes in 8–15 minutes on a single RTX 3090, including AIR export, projection computation, and adapter attachment. Memory consumption peaks at 6GB. Resulting AIR artifacts occupy 9MB for metadata and weights combined—compact enough for edge deployment and content delivery networks.

By contrast, directly training LoRA adapters requires 30–45 minutes per model with 12GB memory. Trans-LoRA requires 2+ hours for data generation and retraining. Cross-LoRA reports 20 minutes on V100 hardware, comparable to UAL but without multi-agent deployment frameworks.

Storage implications are particularly significant for large-scale deployments. An organization deploying 10 models with 50 domain adapters would need 500 model-specific adapters (5GB+ each) = 2.5TB total. With UAL, 50 AIR adapters (9MB each) = 450MB total, a 5,500× reduction enabling adapter distribution via edge caches and mobile networks.

5.6 Comparison with Cross-LoRA

Cross-LoRA [14] performs SVD-based subspace alignment for training-free transfer, conceptually similar to UAL's projection strategy. However, our experimental focus differs. Cross-LoRA optimizes for benchmark accuracy on reasoning tasks (ARC, HellaSwag, MMLU), reporting relative gains up to 5.26% over base models. UAL prioritizes production deployment metrics: transfer speed, artifact size, multi-agent orchestration, and behavioral domain quality.

On overlapping technical dimensions, UAL achieves comparable or superior performance. Both methods complete transfer in 10–20 minutes. UAL produces 9MB artifacts compared to Cross-LoRA's 15–20MB adapters. UAL successfully transfers across more architecturally diverse model families (GPT-2, LLaMA, Qwen) while Cross-LoRA demonstrates strongest results within architectural families.

Where UAL extends beyond Cross-LoRA is in deployment infrastructure. We provide AIR format specifications, family binder implementations, governance frameworks, and multi-agent orchestration patterns that Cross-LoRA doesn't address. These contributions target the operational gap between proof-of-concept and production deployment.

6 Discussion

6.1 Limitations and Future Directions

UAL accepts several tradeoffs in pursuit of training-free portability. Quality degradation of 5–15% relative to directly trained adapters may be unacceptable for benchmark-critical applications. Attachment failures in architectures with unusual structural patterns require manual binder development. Dimension-adaptive projection works well for moderate size differences (768→2048) but may struggle with extreme ratios (160→7168). Adapters trained with knowledge of upcoming transfer destinations could optimize for portability.

Several research directions could address these limitations. Learned projection operators could replace SVD with trainable mappings that preserve adapter effectiveness during dimension changes. Automated binder generation using architectural introspection could handle new model families without manual implementation. Quality prediction models could estimate post-transfer performance before deployment, guiding routing decisions. Cross-modal extensions could enable adapter transfer between language and vision models.

6.2 Theoretical Foundations

Why does UAL work? The success of dimension-adaptive projection relies on the observation that LoRA adapters learn low-rank structures capturing task-relevant features. SVD extracts these features into orthogonal subspaces ordered by importance. By preserving the most significant subspace components (95% variance), we retain the core learned behavior while adapting to new dimensions.

The effectiveness of role-based abstraction stems from transformer architecture standardization. Despite naming differences, most transformers implement similar computational patterns: attention mechanisms compute query-key similarities weighted by values, MLPs apply nonlinear transformations, layer norms stabilize activations. AIR roles map to these semantic operations rather than specific implementations, enabling cross-architecture transfer.

However, this theory has limits. When target architectures introduce fundamentally new operations (e.g., mixture of experts, sparse attention patterns, memory-augmented transformers), role-based abstraction may be insufficient. Future work should investigate how UAL extends to architectures that deviate significantly from standard transformers.

6.3 Ethical Considerations

Universal adapters raise several ethical considerations. Malicious adapters could spread more easily across model families, requiring robust adapter verification. Skill marketplaces could enable the sale of adapters encoding copyrighted knowledge or biased behaviors. Organizations must carefully curate adapter libraries and implement governance frameworks.

Positively, UAL reduces the computational cost of adapter deployment, decreasing the energy footprint of multi-agent systems. By enabling adapter reuse, UAL could reduce the total amount of model training required across the industry. These environmental benefits deserve consideration alongside the technical contributions.

6.4 Broader Impact

UAL represents a step toward modular AI systems where capabilities become portable components. Just as software engineering benefits from reusable libraries and package managers, AI deployment could benefit from standardized adapter formats and skill repositories. Organizations could invest in high-quality adapters with confidence they'll remain useful across model upgrades. Researchers could share adapters that others can deploy without retraining. Edge

devices could maintain small libraries of compressed adapters rather than hosting separate models.

This vision requires community coordination around standards. The AIR format we propose serves as a starting point, but production adoption requires industry consensus. We hope UAL demonstrates sufficient value to motivate standardization efforts.

7 Conclusion

Universal-Adopter LoRA addresses the operational challenges of deploying domain expertise across heterogeneous multi-agent systems. By decoupling adapter semantics from model implementations through architecture-agnostic intermediate representations, family-aware runtime binding, and dimension-adaptive projection, UAL enables training-free adapter transfer across diverse model families.

Our experiments demonstrate successful transfer from Pythia to GPT-2, TinyLlama, and Qwen, achieving 75–100% attachment rates and 70–85% of directly trained adapter effectiveness while completing transfer in 8–15 minutes and producing 9MB artifacts. For multi-agent deployments where model heterogeneity is inevitable and retraining is impractical, these tradeoffs favor UAL decisively.

UAL's design prioritizes practical deployment over benchmark supremacy. While we accept 5–10% benchmark degradation relative to direct training, we gain dramatic improvements in deployment speed, storage efficiency, and operational flexibility. For multi-agent systems where model heterogeneity is unavoidable and retraining is impractical, these tradeoffs favor UAL decisively.

Beyond immediate technical contributions, UAL advances toward modular AI systems where domain expertise becomes portable. By treating adapters as first-class artifacts with standardized formats, governance frameworks, and lifecycle management, we enable an ecosystem where knowledge flows across architectural boundaries. Organizations can train once and deploy everywhere, invest in adapters with confidence they'll outlive specific base models, and share expertise through adapter repositories.

The framework remains extensible as new challenges emerge. Learned projection operators could improve transfer quality. Cross-modal extensions could enable knowledge sharing between language and vision models. Automated quality prediction could guide deployment decisions. Enriched metadata standards could support commercial adapter marketplaces. Architecture-aware training could produce adapters that transfer even more naturally.

We release our complete implementation including AIR format specification, family binders for major architectures, dimension-adaptive projection algorithms, and governance templates. We hope UAL serves as a foundation for research on modular, transferable, architecture-agnostic adaptation in large language models. By demonstrating that universal adapters are not merely technically feasible but practically deployable, we aim to accelerate the transition toward truly composable AI systems.

References

[1] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models, 2021.

[2] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-Decomposed Low-Rank Adaptation, 2024.

[3] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. LoRA+: Efficient Low Rank Adaptation of Large Models, 2024.

[4] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-Efficient Transfer Learning for NLP, 2019.

[5] Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models, 2022.

[6] Brian Lester, Rami Al-Rfou, and Noah Constant. The Power of Scale for Parameter-Efficient Prompt Tuning, 2021.

[7] Xiang Lisa Li and Percy Liang. Prefix-Tuning: Optimizing Continuous Prompts for Generation, 2021.

[8] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-Tuning v2: Prompt Tuning Can Be Comparable to Fine-tuning Universally Across Scales and Tasks, 2022.

[9] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network, 2015.

[10] Mary Phuong and Christoph H. Lampert. Towards Understanding Knowledge Distillation, 2021.

[11] Gal Kaplun, Eran Malach, Preetum Nakkiran, and Shai Shalev-Shwartz. Knowledge Distillation: Bad Models Can Be Good Role Models, 2022.

[12] Runqian Wang, Soumya Ghosh, David Cox, Diego Antognini, Aude Oliva, Rogerio Feris, and Leonid Karlinsky. Trans-LoRA: Towards Data-Free Transferable Parameter Efficient Finetuning, 2025.

[13] Yanan Li, Fanxu Meng, Muhan Zhang, Shiai Zhu, Shangguang Wang, and Mengwei Xu. LoRASuite: Efficient LoRA Adaptation Across Large Language Model Upgrades, 2025.

[14] Feifan Xia, Mingyang Liao, Yuyang Fang, Defang Li, Yantong Xie, Weikang Li, Yang Li, Deguo Xia, and Jizhou Huang. Cross-LoRA: A Data-Free LoRA Transfer Framework across Heterogeneous LLMs, 2025.

[15] Farzad Farhadzadeh, Debasmit Das, Shubhankar Borse, and Fatih Porikli. LoRA-X: Bridging Foundation Models with Training-Free Cross-Model Adaptation, 2025.

[16] Farzad Farhadzadeh, Debasmit Das, Shubhankar Borse, and Fatih Porikli. Zero-Shot Adaptation of Parameter-Efficient Fine-Tuning in Diffusion Models, 2025.