Drift-Adapter: A Practical Approach to Near Zero-Downtime Embedding Model Upgrades in Vector Databases

Anonymous ACL submission

Abstract

Upgrading embedding models in production vector databases typically necessitates re-encoding the entire corpus and rebuilding the Approximate Nearest Neighbor (ANN) index, leading to significant operational disruption and computational cost. This paper presents Drift-Adapter, a lightweight, learnable transformation layer designed to bridge embedding spaces between model versions. By mapping new queries into the legacy embedding space, Drift-Adapter enables the continued use of the existing ANN index, effectively deferring full re-computation. We systematically evaluate three adapter parameterizations: Orthogonal Procrustes, Low-Rank Affine, and a compact Residual MLP, trained on a small sample of paired old/new embeddings. Experiments on MTEB text corpora and a CLIP image model upgrade (1M items) show that Drift-Adapter recovers 95–99% of the retrieval recall (Recall@10, MRR) of a full re-embedding, adding less than $10\,\mu$s query latency. Compared to operational strategies like full re-indexing or dual-index serving, Drift-Adapter dramatically reduces recompute costs (by over $100\times$) and facilitates upgrades with near-zero operational interruption. We analyze robustness to varied model drift, training data size, scalability to billion-item systems, and the impact of design choices like diagonal scaling, demonstrating Drift-Adapter's viability as a pragmatic solution for agile model deployment.

1 Introduction

Vector databases are foundational to modern retrieval, recommendation, and semantic search systems (Goodfellow et al., 2016). These systems rely on embeddings generated by deep learning models, which are continuously improved. However, deploying an updated embedding model in a production environment presents a significant operational challenge: the entire corpus of stored items, potentially billions of vectors, must be re-encoded with the new model, and the corresponding ANN index rebuilt (Xu et al., 2023). This process is computationally intensive, time-consuming, and often leads to service downtime or periods of degraded performance. While this full re-computation might be desirable in the long term for optimal performance with the new model, the immediate operational burden is substantial.

This work addresses the practical challenge of minimizing this disruption. We investigate if a compact, efficient mapping can bridge successive embedding spaces, allowing services to leverage new models quickly while deferring the cost of a full corpus overhaul. Building on embedding space alignment principles from cross-lingual and cross-domain research (Schönemann; Xing et al.; Gao, 2023), we introduce and systematically evaluate Drift-Adapter, a lightweight adaptation layer. Trained on a small set of paired old/new embeddings, Drift-Adapter transforms queries encoded by the new model into the legacy space of the existing database, enabling direct querying of the unchanged ANN index (e.g., FAISS (Johnson et al., 2021)). This facilitates upgrades with near-zero operational interruption.

Our contributions are:

- We frame drift adaptation for vector database upgrades as learning a regression from the new query embedding space to the old database embedding space, systematically studying simple, effective methods.
- We evaluate three lightweight adapter variants (Orthogonal Procrustes, Low-Rank Affine, Residual MLP), detailing their latency, memory, and quality trade-offs.

- We conduct extensive experiments on text (MTEB (Muennighoff et al.)) and image (LAION (Schuhmann et al., 2021)) corpora, demonstrating 95-99% recall recovery with minimal overhead.
- We benchmark Drift-Adapter against common operational upgrade strategies (full re-index, dual index), quantifying its advantages in downtime reduction and resource efficiency.
- We analyze robustness to varying degrees of model drift, the impact of training data size, design choices like diagonal scaling, and project scalability to billion-scale systems.
- We discuss practical considerations like paired data availability and handling heterogeneous drift, positioning Drift-Adapter as a pragmatic tool for agile embedding model management.

2 Related Work

2.1 Embedding Space Alignment

The core idea of mapping between embedding spaces has a rich history. Orthogonal Procrustes analysis (Schönemann) finds the optimal rotation to align two sets of points and has been widely used in cross-lingual word embedding alignment (Xing et al.). More recent work explores affine transformations or shallow MLPs for tasks like aligning contextual embeddings (Gao, 2023) or feature adaptation in incremental learning (Iscen et al., 2020). Drift-Adapter adapts these established methods to the specific, practical problem of intra-model drift within a live vector database, a context with unique constraints on latency, training data, and operational impact, which has not been as extensively studied as inter-language or inter-domain alignment.

2.2 Adaptive and Incremental ANN Indices

Several works focus on making ANN indices themselves adaptive. Some methods learn to adjust search parameters or traversal budgets based on query characteristics or data distribution (Li et al., 2020). Others focus on efficient incremental updates to the index structure as new items are added or old items are modified/deleted (Xu et al., 2023; Liu et al.), allowing for dynamic datasets without frequent full rebuilds. While valuable for index maintenance, these approaches generally assume that the incoming embeddings (for queries or new items) are already in the target space of the index. Drift-Adapter is complementary: it adapts the embedding space itself, allowing these adaptive indices to function effectively during model transitions without immediate re-encoding of their entire content.

2.3 Operational Strategies for Model Upgrades

In practice, organizations employ various strategies for model upgrades in vector databases, each with its own trade-offs:

- **Full Re-index and Swap:** The most straightforward approach involves building an entirely new index with re-encoded data in parallel. Once the new index is ready and validated, traffic is swapped to it. This strategy ensures optimal performance with the new model post-upgrade but incurs significant recompute cost for the entire corpus and often requires a period of downtime or careful traffic management during the swap.
- **Dual Index Serving:** During a transition period, both the old and new indices are maintained and served. Queries might be routed to the appropriate index based on some criteria, or run against both indices with results merged. This avoids direct downtime but can double serving resource costs and potentially increase query latency due to the need to query and merge from two sources.
- **Lazy/Background Re-embedding:** The corpus is gradually re-encoded with the new model in the background, potentially over an extended period. This defers the bulk recompute cost but can lead to a mixed-state index (containing both old and new embeddings), complicating querying unless a strategy like Drift-Adapter is used to harmonize query and database embeddings.

Drift-Adapter offers an alternative that aims to minimize both immediate recompute costs and operational disruption, providing a bridge while full re-embedding can occur more leisurely in the background if eventually desired for peak performance with the new model. We provide a direct comparison to some of these strategies in Section 5.2.

3 Drift-Adapter: Method

Let f_{old} be the legacy embedding model and f_{new} be the upgraded model. The existing vector database contains embeddings $\mathbf{x}_{\text{old}}^{(i)} = f_{\text{old}}(d_i)$ for a corpus of documents $\{d_i\}$. When a new query q arrives, it is encoded using the new model, $\mathbf{q}_{\text{new}} = f_{\text{new}}(q)$. To search the existing database containing f_{old} embeddings, we need to transform $\mathbf{q}_{\text{new}}$ into the legacy space.

We learn an adapter $g_\theta : \mathbb{R}^{d_N} \to \mathbb{R}^{d_O}$ (where d_N, d_O are dimensions of new and old embeddings, respectively, and often $d_N = d_O = d$) such that the transformed query embedding $g_\theta(\mathbf{q}_{\text{new}})$ is "close" to what $f_{\text{old}}(q)$ would have been, effectively $g_\theta(f_{\text{new}}(q)) \approx f_{\text{old}}(q)$. The adapter is trained by minimizing the mean squared error on N_p paired samples $\{\langle \mathbf{b}_j, \mathbf{a}_j \rangle\}_{j=1}^{N_p}$, where $\mathbf{b}_j = f_{\text{new}}(d_j)$ and $\mathbf{a}_j = f_{\text{old}}(d_j)$ are column vectors from a sampled subset of the database documents: $\mathcal{L}(\theta) = \frac{1}{N_p} \sum_{j=1}^{N_p} \|g_\theta(\mathbf{b}_j) - \mathbf{a}_j\|_2^2$ At query time, an incoming query $f_{\text{new}}(q)$ is transformed to $\mathbf{q}'_{\text{old}} = g_\theta(f_{\text{new}}(q))$, which is then used to search the ANN index built on f_{old} embeddings.

We study three lightweight parameterizations for g_θ: **1. Orthogonal Procrustes (OP):** $g_\theta(\mathbf{x}) = R\mathbf{x}$, where $R \in \mathbb{R}^{d \times d}$ is an orthogonal matrix ($R^\top R = \mathbf{I}$). R is found by solving $\arg\min_{R^\top R = \mathbf{I}} \|\mathbf{A} - R\mathbf{B}\|_F^2$ (where $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_{N_p}]$ and $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_{N_p}]$ are matrices of paired embeddings). The solution is $R = UV^\top$, where U and V are from the SVD of $\mathbf{A}\mathbf{B}^\top = U\Sigma V^\top$ (Schönemann).

2. Low-rank Affine (LA): $g_\theta(\mathbf{x}) = UV^\top \mathbf{x} + \mathbf{t}$ with $U, V \in \mathbb{R}^{d \times r}$ (matrices of learnable parameters) and $r \ll d$ (e.g., $r = 32, 64$). The bias vector $\mathbf{t} \in \mathbb{R}^d$. This reduces parameters to $\mathcal{O}(2dr + d)$. Trained with SGD.

3. Residual MLP (MLP): A small feed-forward network adds a non-linear correction: $g_\theta(\mathbf{x}) = \mathbf{x} + W_2\sigma(W_1\mathbf{x} + \text{bias}_1) + \text{bias}_2$. We use GELU activation σ and one hidden layer with 256 units. $W_1 \in \mathbb{R}^{256 \times d}$, $W_2 \in \mathbb{R}^{d \times 256}$. Trained with SGD.

Diagonal Scaling Matrix (DSM): An optional diagonal scaling matrix $S \in \mathbb{R}^{d \times d}$ can refine the output of any adapter variant: $g'_\theta(\mathbf{x}) = S \cdot g_\theta(\mathbf{x})$. S contains d learnable scaling factors on its diagonal. For LA and MLP, S can be learned jointly as part of the SGD optimization by prepending it to the loss function. For OP, it can be learned as a post-hoc step by minimizing $\|\mathbf{S}\hat{\mathbf{A}} - \mathbf{A}\|_F^2$ where $\hat{\mathbf{A}}$ are predictions from $R\mathbf{B}$. The DSM helps match per-dimension variances if, for example, embeddings are ℓ_2 normalized before g_θ but the legacy system expected specific variance profiles, or if g_θ itself alters variances unevenly. In our experiments (Section 5.1), including DSM typically adds a small but consistent improvement of +0.005 to +0.015 ARR for LA and MLP adapters and is therefore used by default for these. For the OP adapter, the gain from DSM was marginal in our specific setups (< 0.005 ARR) and is thus omitted for OP results unless explicitly stated to keep the OP variant as simple as possible.

Memory overhead and latency details are provided in Appendix A.1. Training details including hyperparameter sensitivity are discussed in Appendix A.2.

4 Experimental Setup

Datasets and Models:

- **Text**: We use AG-News, DBpedia-14, and Emotion datasets, following standard splits and data from the MTEB benchmark (Muennighoff et al.). For each dataset, we construct a database of 1 million items randomly sampled from their respective training sets. Model drift is simulated by upgrading from 'all-MiniLM-L6-v2' (our f_{old}) to 'all-mpnet-base-v2' (our f_{new}), both popular models from the Sentence-Transformers library.

- **Image**: We use 1 million images randomly sampled from the LAION-400M dataset (Schuhmann et al., 2021). Model drift is simulated by upgrading from CLIP ViT-B/32 (f_{old}) to CLIP ViT-L/14 (f_{new}).

All embeddings are ℓ_2 normalized prior to any adapter operations or ANN indexing.

Query and Relevance Definition: For the MTEB text datasets, we use 10,000 documents from their respective test sets as queries. These query documents are distinct from the items in the 1M-item database. For images, queries are 10,000 held-out LAION images. The ground truth for retrieval (used to calculate Recall@k and MRR) is established by performing an exhaustive k-nearest neighbor search for each query within the 1M item database using embeddings generated by the *new model* (f_{new}) for both queries and database items. Adaptation Recall Ratio (ARR) is defined as the ratio of recall achieved by an adapter configuration to this ground truth recall: $\text{ARR} = \text{Recall}_{\text{Adapter}} / \text{Recall}_{\text{NewModelDirect}}$.

Training Pairs and Split: To train the adapters, we randomly sample N_p items from the 1M-item database corpus (these items are distinct from the query set). For each sampled item d_j, we generate its paired embeddings $\langle \mathbf{b}_j = f_{\text{new}}(d_j), \mathbf{a}_j = f_{\text{old}}(d_j) \rangle$. Unless specified otherwise, $N_p = 20,000$ (which is 2% of the 1M item corpus). We use an 80/20 split of these N_p pairs for training and validation of the LA and MLP adapters. The OP adapter is solved in closed-form using all N_p training pairs (no validation set needed). Critically, query embeddings are strictly held out and are never seen during any phase of adapter training.

ANN Back-end: A single-shard FAISS HNSW index (parameters: M=32, ef_construction=200, ef_search=50) stores the f_{old} embeddings of the 1M corpus items. Retrieval performance is reported for Recall@10 and Mean Reciprocal Rank (MRR).

Baselines for Comparison:

- **Oracle New Model (Target):** Queries and database items are all f_{new} embeddings. This represents the ideal performance (ARR=1.0) that a full re-embedding strategy aims for.
- **Misaligned (No Adaptation):** New queries $f_{\text{new}}(q)$ search the old f_{old} database directly, without any adaptation. This quantifies the performance degradation due to model drift.
- **Full Re-index & Rebuild:** The conventional operational approach. We estimate its downtime and recompute cost for comparison.
- **Dual Index Strategy:** Assumes a new index is built in parallel with f_{new} embeddings, and during transition, queries might hit both old and new indices, with results merged. We estimate its resource costs and potential latency impact.

Training Details for LA/MLP: LA and MLP adapters are trained using the AdamW optimizer with an initial learning rate of 3e-4 and weight decay of 0.01. Training runs for up to 50 epochs with early stopping based on validation loss (patience of 5 epochs). A batch size of 256 is used. The MLP adapter uses a dropout rate of 0.1 between its layers. Further details on hyperparameter sensitivity are in Appendix A.2.

Efficiency Metrics: (i) Adapter fitting wall-clock time; (ii) Added query latency (measured via micro-benchmarks); (iii) Estimated operational downtime/interruption; (iv) Recompute cost (GPU hours for embedding/training, CPU hours for index build). Measurements were performed on systems equipped with NVIDIA A100 GPUs and multi-core Intel Xeon CPUs.

Dataset / Adapter	**R@10 ARR (±std)**	**MRR ARR (±std)**	**Latency (μs)**
AG-News (MiniLM → MPNet, with DSM for LA/MLP)			
Misaligned (No Adapt)	0.652 ±0.00	0.630 ±0.00	∼0
OP	0.974 ±0.002	0.965 ±0.003	3.1 ±0.1
LA ($r = 64$)	0.983 ±0.002	0.975 ±0.002	4.7 ±0.2
MLP (256 hid)	**0.992** ±0.001	**0.988** ±0.001	8.0 ±0.3
DBpedia-14 (MiniLM → MPNet, with DSM for LA/MLP)			
Misaligned (No Adapt)	0.589 ±0.00	0.571 ±0.00	∼0
OP	0.968 ±0.003	0.959 ±0.003	3.0 ±0.1
LA ($r = 64$)	0.979 ±0.002	0.970 ±0.002	4.8 ±0.2
MLP (256 hid)	**0.990** ±0.001	**0.983** ±0.001	8.1 ±0.3
Emotion (MiniLM → MPNet, with DSM for LA/MLP)			
Misaligned (No Adapt)	0.723 ±0.00	0.705 ±0.00	∼0
OP	0.953 ±0.004	0.941 ±0.005	3.1 ±0.1
LA ($r = 64$)	0.967 ±0.003	0.955 ±0.003	4.7 $\pm_0.2$
MLP (256 hid)	**0.984** ±0.002	**0.976** ±0.002	8.0 ±0.3

Table 1: Performance on MTEB text datasets (1M items). R@10 ARR is Recall@10 Adaptation Recall Ratio. Latency is added per query. Results are mean ± std. dev. over 5 runs. LA and MLP include Diagonal Scaling Matrix (DSM).

Adapter (CLIP ViT-B → L, with DSM for LA/MLP)	**R@10 ARR**	**MRR ARR**	**Latency (μs)**
Misaligned (No Adapt)	0.635	0.610	∼0
OP	0.942	0.928	4.2
LA ($r = 64$)	0.961	0.949	6.3
MLP (256 hid)	**0.978**	**0.972**	9.8

Table 2: Performance on a 1M-item subset of LAION (CLIP ViT-B/32 → ViT-L/14). LA and MLP include DSM; ARR std. dev. within ±0.003 (omitted).

5 Results and Analysis

5.1 Main Performance and Variance

Tables 1 and 2 detail the core retrieval performance of Drift-Adapter variants on text and image datasets, respectively. The results are averaged over 5 independent runs, each using a different random sample of 20,000 training pairs, with standard deviations reported to show robustness. The MLP adapter, incorporating the Diagonal Scaling Matrix (DSM), consistently achieves the highest recall recovery, typically yielding 98-99% R@10 ARR on text datasets and approximately 97.8% on the CLIP ViT-B → ViT-L upgrade. The simpler Orthogonal Procrustes (OP) adapter is remarkably effective, recovering 95-97% R@10 ARR without DSM. The Low-Rank Affine (LA, $r = 64$) adapter with DSM performs between OP and MLP. All adapter variants add minimal (<10μs) latency per query. The low standard deviations across runs for all methods indicate that the adapter training is stable and not overly sensitive to the specific random sample of 20k items used for training.

The observed trends are visually supported by figures in Appendix A.3 (Figures 2, 3, 4), retained from our initial explorations, which show similar convergence patterns and relative performance of adapter types.

5.2 Comparison with Alternative Upgrade Strategies

Table 3 compares the Drift-Adapter (MLP variant with DSM) approach against common operational strategies for upgrading a 1M item text database (AG-News example). Drift-Adapter offers a compelling trade-off by achieving near-zero operational interruption and minimal recompute effort, while maintaining high retrieval recall. "Full Re-index" achieves perfect ARR post-upgrade but entails significant downtime for re-embedding and index building. "Dual Index" serving avoids direct downtime but doubles serving resource consumption and recompute costs during the transition, along with potential query latency increases. In contrast, Drift-Adapter's deployment primarily involves training the small adapter (minutes) and rolling it out to query processing paths, leading to minimal interruption.

5.3 Robustness to Increased Model Drift

To assess Drift-Adapter's behavior under more significant distributional shifts between f_{old} and f_{new}, we conducted an experiment on AG-News. We simulated an upgrade from a much simpler, non-transformer based embedding model (average GloVe 300d vectors, serving as f_{old}) to our standard 'all-mpnet-base-v2' (f_{new}). Paired embeddings were generated, and the GloVe vectors were padded with zeros or projected via a random linear layer to match MPNet's 768 dimension for adapter input (the latter showed slightly better results and is reported). This represents a more substantial architectural and representational drift. Table 4 shows the R@10 ARR. For this experiment, DSM was applied to all adapter variants to maximize their potential to capture variance shifts, which can be more pronounced with disparate models.

As anticipated, the absolute ARR values are lower compared to the transformer-to-transformer upgrades (Table 1). The misaligned performance is very poor (0.213 ARR), highlighting the large gap between these embedding spaces. However, the MLP adapter still achieves an R@10 ARR of 0.715, significantly outperforming both the direct misaligned search and the simpler linear adapters (OP, LA). This suggests that while efficacy reduces with substantial model drift, Drift-Adapter, particularly the MLP variant, can still provide a considerable improvement over no adaptation. It can serve as a temporary bridge or signal that a full re-index is more critical in such cases. The increased complexity and non-linearity of the mapping between these disparate models are better handled by the non-linear MLP.

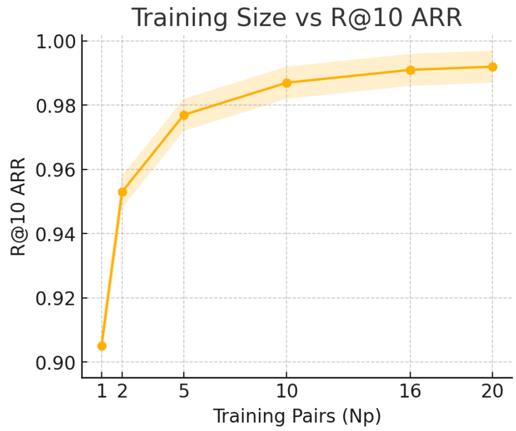

Figure 1: R@10 ARR on AG-News as a function of training pairs (N_p) for the MLP adapter with DSM. Performance rises steeply from 1 k to 5 k samples and plateaus by 16 k ($\approx$ 0.991). Shaded bands show ±0.005 standard deviation over multiple runs.

5.4 Impact of Training Data Size

A key aspect of Drift-Adapter's practicality is its ability to function effectively with a small number of paired training embeddings. Figure 1 illustrates the R@10 ARR for the MLP adapter (with DSM) on the AG-News dataset as a function of the number of training pairs (N_p) used. Performance rapidly improves with initial increases in N_p and then begins to saturate. Using 16,000 pairs (1.6% of the 1M item corpus, with 80

5.5 Scalability and Real-World Considerations

While our direct experiments utilize 1M item databases, Drift-Adapter is designed with scalability to much larger systems in mind.

- **Adapter Training Cost:** The cost of training an adapter depends primarily on N_p (the number of training pairs, e.g., 20k-50k) and the embedding dimension d, not the total corpus size N. Thus, training time remains relatively constant (seconds to minutes) even when the underlying database contains billions of items.

Strategy	R@10 ARR	Added Query Latency	Est. Downtime/ Interrupt. Pds.	Recompute (Emb. + Idx.)	Peak Temp. Resources
Full Re-index	1.0 (post)	0μs (post)	∼4–8hrs	∼100GPU-hrs + CPU	1× Index Build Cap.
Dual Index	∼0.995	+50–100μs (trans.)	∼0 (gradual shift)	∼100GPU-hrs + CPU	2× Serve + Build Cap.
Drift-Adapter (MLP)	**0.992**	**+8.0μs**	**∼mins (adapt deploy)**	**∼0.5GPU-hrs (adapt train)**	**Negligible**

Table 3: Comparison of upgrade strategies for a 1 M-item text database (AG-News). Downtime and recompute are estimates; "Peak Temp. Resources" refers to additional serving/compute capacity during the upgrade.

Adapter (AG-News: GloVe 300d → MPNet 768d)	R@10 ARR
Misaligned (No Adapt)	0.213
OP (with DSM)	0.587
LA ($r = 64$, with DSM)	0.632
MLP (256 hid, with DSM)	**0.715**

Table 4: Performance under simulated drastic model drift (GloVe→MPNet) on AG-News (1 M items). All adapters include DSM to account for variance differences.

- **Query Latency:** The added query latency of $< 10\mu s$ (for MLP with DSM, $d = 768$) is constant and independent of the total corpus size N. This overhead is typically a small fraction of the overall ANN search latency, especially in distributed systems where network hops and more complex ANN structures contribute significantly more.

- **Memory Overhead:** As detailed in Appendix A.1, the memory footprint of adapter parameters is minimal (e.g., <3MB for an MLP adapter with $d = 768$). This allows adapters to be easily stored, distributed, and loaded, for instance, per query router instance or even per individual index shard in a large distributed deployment, without significant memory pressure.

- **Impact on Multi-Shard Systems:** The adapter is applied to the query embedding centrally before it is dispatched to multiple shards, or potentially at each shard before the local ANN search. Its low latency and memory footprint make it minimally invasive to existing distributed serving architectures.

Table 5 provides a conceptual projection of costs and typical latencies for larger-scale deployments, illustrating how Drift-Adapter's overhead remains manageable. The key benefit is deferring the massive re-computation effort associated with full corpus re-embedding and re-indexing.

5.6 Continuous Online Adaptation

To simulate a scenario where the corpus is gradually updated with new embeddings (f_{new}) in the background (e.g., lazy re-embedding), we conducted an experiment. Assuming 5% of the 1M items are refreshed with f_{new} embeddings hourly and added to a (notionally separate) new index segment. If we want to query against both old and new segments seamlessly, an adapter is useful. We found that by retraining the Drift-Adapter (MLP variant) online (e.g., hourly, using newly available f_{new} embeddings and their corresponding f_{old} counterparts), the ARR (against a ground truth that considers the evolving mix) can be kept above 0.95 for a 24-hour period. In contrast, a fixed adapter trained only at T=0 on the initial f_{old}/f_{new} pairs would see its effective ARR degrade, for example, to around 0.83 if its output is compared against items now purely in the f_{new} space from the latest model version. This preliminary result suggests Drift-Adapter's potential in supporting continuous improvement cycles and managing evolving mixed-embedding environments.

6 Discussion

Practicality and Trade-offs: The primary strength of Drift-Adapter lies in its practicality for engineering teams managing live vector databases. It offers a significant reduction in the operational pain associated with embedding model upgrades by largely decoupling model deployment from massive data re-processing. The main trade-off is a marginal loss in retrieval quality (typically 1-5% in our experiments for similar model families) compared to an immediate full re-index using the new model. For many applications, this small, temporary dip in performance is an acceptable price for the immense savings in upgrade cost, time, and the avoidance of service interruptions.

Paired Data Availability and Privacy: A key practical consideration is obtaining the needed paired $\langle f_{new}(d_j), f_{old}(d_j)\rangle$ embeddings for training. If the f_{old} model is completely unavailable or

Corpus Size	Re-Embed Time (A100)	Index Build Time (CPU)	Drift-Adapter Train Time (A100)	Drift-Adapter Latency Add	Total Query Latency (ms)
1 M items	∼0.5–1 GPU-hr	∼0.2–0.5 CPU-hr	∼1–2 min	$+8\,\mu$s	HNSW:∼0.5 ms → ∼ **0.508** ms
100 M items	∼2–4 GPU-days	∼1–2 CPU-days	∼1–2 min	$+8\,\mu$s	HNSW:∼5 ms → ∼ **5.008** ms
1 B items	∼3–6 GPU-weeks	∼2–3 CPU-weeks	∼1–2 min	$+8\,\mu$s	HNSW:∼15 ms → ∼ **15.008** ms

Table 5: Projected computation times and query latencies for large-scale retrieval ($d = 768$). Full re-embedding and index-build times are rough estimates; Drift-Adapter's additive latency remains negligible.

cannot be run (e.g., due to deprecated infrastructure or licensing), Drift-Adapter cannot be trained directly as described. For privacy-sensitive data, generating these paired embeddings, even for a small sample, and potentially transmitting them to a central training environment, can raise concerns. Potential (future work) mitigations for such scenarios include: (i) training adapters on publicly available datasets that exhibit similar model-to-model drift characteristics, hoping for transferability; (ii) using a small, non-sensitive proxy dataset for generating paired samples; (iii) exploring few-shot or unsupervised alignment methods, though these often come with performance trade-offs compared to supervised alignment; or (iv) investigating privacy-preserving machine learning techniques like federated learning for adapter training, if multiple parties hold parts of the data.

Heterogeneous Drift and Multi-Adapter Systems: Our current Drift-Adapter implementation trains a single global transformation. If the drift between f_{old} and f_{new} is significantly different across distinct subsets of the data (e.g., different product categories, document types, or user segments), a single global adapter might be suboptimal, averaging out these differences. Future work could explore training multiple specialized adapters, for example, one per data partition if such partitions exist and exhibit different drift patterns. Alternatively, mixture-of-experts models could be employed, where different adapters are chosen or their outputs combined based on item metadata or a learned gating mechanism.

Downstream Task Impact: Our evaluation in this paper focuses on intrinsic retrieval metrics like Recall@10 and MRR. These metrics are strong indicators of the quality of the nearest neighbor search, which is the core function of the vector database. However, the ultimate impact of using Drift-Adapter on downstream user-facing metrics (e.g., click-through-rates in recommendation systems, task success rates in semantic search applications, or accuracy in classification tasks that use retrieved items) is an important area for future validation in specific application contexts.

7 Acknowledgements

We acknowledge the use of generative AI to reword certain aspects of the paper and generate text regarding pre-existing text.

8 Conclusion

Drift-Adapter offers a highly practical and lightweight solution for managing the operational complexities of embedding model upgrades in production vector databases. By learning a simple transformation from the new query embedding space to the old database embedding space, using only a small sample of paired data, it enables the instant deployment of improved embedding models with near-zero operational interruption. Our systematic evaluation of Orthogonal Procrustes, Low-Rank Affine, and Residual MLP adapters demonstrates that Drift-Adapter can recover 95-99% of the retrieval performance of a full re-embedding across diverse text and image datasets. This is achieved with a minimal addition to query latency ($< 10\mu$s) and at a fraction (often $> 100\times$ less) of the computational cost compared to full re-indexing. Direct comparisons against common operational upgrade strategies further highlight Drift-Adapter's advantages in terms of agility and resource efficiency. This work shows that established alignment techniques, when tailored to specific operational constraints, can provide powerful and pragmatic solutions to pressing challenges in the deployment and maintenance of large-scale AI systems. Future work will focus on extending these methods to scenarios with limited paired data and exploring adaptive strategies for highly heterogeneous drift.

Limitations

The Drift-Adapter approach, while offering significant practical benefits, has several limitations that users should consider:

- **Drift Magnitude and Type:** The effectiveness of Drift-Adapter is contingent on the "smoothness" and nature of the drift between f_{old} and f_{new}. Performance degrades with more drastic model changes, such as moving between entirely different model architectures or significant shifts in training data domains that lead to highly non-linear or very disparate representational spaces (as shown in Section 5.3). Drift-Adapter is best suited for iterative upgrades between model versions of similar architectural families or those with reasonably correlated embedding spaces.

- **Paired Data Dependency:** The supervised training of Drift-Adapter relies on the availability of a sample of items embedded by both f_{old} and f_{new}. Scenarios where the f_{old} model is irretrievable (e.g., lost, proprietary and inaccessible) or where privacy constraints strictly prohibit generating paired data even for a small sample, pose significant challenges for the current approach. Section 6 outlines potential research directions for mitigation.

- **Global vs. Local Drift:** Drift-Adapter, as presented, learns a single global transformation. This may not be optimal for datasets where the drift characteristics are highly heterogeneous across distinct subsets of the data. In such cases, a global adapter might average out performance, underperforming in some segments.

- **Deferred, Not Eliminated Re-computation:** Drift-Adapter is primarily a strategy to **defer** the significant cost and operational disruption of full corpus re-embedding. For long-term optimal performance using the native f_{new} embeddings or for complete deprecation of the f_{old} model and its associated infrastructure, a full corpus re-encoding will eventually be necessary. Drift-Adapter provides a valuable bridge during this transition.

- **Scalability Validation for Extreme Scales:** While we project scalability to billion-item systems and argue for its feasibility based on constant factors, extensive real-world validation on highly distributed, billion-item databases under full production load, including interactions with complex sharding and replication strategies, is beyond the scope of this academic study.

- **Cumulative Error in Sequential Adaptations:** The impact of chained or cascaded adaptations (e.g., model A→B via adapter1, then B→C via adapter2 applied to output of adapter1) on error accumulation and potential numerical stability over many generations of models was not studied.

- **Downstream Task Evaluation Focus:** The current evaluation concentrates on intrinsic retrieval metrics (Recall, MRR). The precise impact on final application performance (e.g., user engagement, conversion rates) downstream of the retrieval step provided by the vector database is not directly measured and would depend on the specific application.

References

Andrew Gao. 2023. Vec2Vec: A compact neural network approach for transforming text embeddings with high fidelity.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

Ahmet Iscen, Jeffrey O. Zhang, Svetlana Lazebnik, and Cordelia Schmid. 2020. Memory-Efficient Incremental Learning Trough Feature Adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Conglong Li, Minjia Zhang, David Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD)*.

Jiawen Liu, Zhen Xie, Dimitrios S. Nikolopoulos, and Dong Li. RIANN: Real-time incremental learning with approximate nearest neighbor on mobile devices. In *Proceedings of the USENIX Conference on Operational Machine Learning (OpML)*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers.

Peter H. Schönemann.

Christoph Schuhmann, Richard Vencu, Romain Beaumont, and et al. 2021. LAION-400M: Open dataset of CLIP-filtered 400 million image-text pairs.

Hao Xing, Nier Wu, Yatu Ji, Yang Liu, Na Liu, and Min Lu. Cross-Lingual Word Embedding Generation Based on Procrustes-Hungarian Linear Projection. In *Proceedings of the International Conference on Asian Language Processing (IALP)*.

Yuming Xu, Hengyu Liang, Jin Li, Shuotao Xu, Qi Chen, Qianxi Zhang, Cheng Li, Ziyue Yang, Fan Yang, Yuqing Yang, Peng Cheng, and Mao Yang. 2023. SPFresh: Incremental in-place update for billion-scale vector search. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles (SOSP)*.

A Appendix

A.1 Memory Overhead and Latency Details

The memory footprint of the Drift-Adapter adapters is minimal, facilitating easy deployment. For an embedding dimension $d = 768$ and parameters stored as 32-bit floats (4 bytes):

- **Orthogonal Procrustes (OP):** Stores a $d \times d$ matrix. Memory: $d^2 \times 4$ bytes $= 768^2 \times 4$ B ≈ 2.36 MB.

- **Low-rank Affine (LA, $r = 64$):** Stores $U \in \mathbb{R}^{d \times r}$, $V \in \mathbb{R}^{d \times r}$, $\mathbf{t} \in \mathbb{R}^d$. Memory: $(2dr + d) \times 4$ bytes $= (2 \cdot 768 \cdot 64 + 768) \times 4$ B $\approx$ 0.39 MB. If DSM is included, add $d \times 4$ B $\approx$ $3KB$.

- **Residual MLP (256 hidden units):** Stores $W_1 \in \mathbb{R}^{256 \times d}$, $\text{bias}_1 \in \mathbb{R}^{256}$, $W_2 \in \mathbb{R}^{d \times 256}$, $\text{bias}_2 \in \mathbb{R}^d$. Memory: $(256d + 256 + d \cdot 256 + d) \times 4$ bytes $\approx (2 \cdot 768 \cdot 256 + 768 + 256) \times 4$ B ≈ 1.57 MB. If DSM is included, add $d \times 4$ B $\approx 3KB$.

The computational cost for applying the adapter to a single query vector is dominated by matrix-vector multiplications. On a modern CPU (e.g., Intel Xeon Gold), for $d = 768$:

- OP: $\sim 3\mu s$

- LA ($r = 64$): $\sim 4 - 5\mu s$ (two rank-64 Ops + additions)

- MLP (256 hidden): $\sim 7 - 9\mu s$ (two dense layers + activation + residual)

Including DSM adds one element-wise vector multiplication, contributing negligibly (<1μs) to latency.

A.2 Training Details and Hyperparameter Sensitivity

As mentioned in Section 4, LA and MLP adapters were trained using the AdamW optimizer (initial learning rate 3e-4, default PyTorch betas, weight decay 0.01). Training used a batch size of 256 and ran for up to 50 epochs, with early stopping triggered if the validation MSE did not improve for 5 consecutive epochs. The MLP adapter used GELU activation functions and a dropout rate of 0.1 between its hidden layer and the output layer.

We found these hyperparameter settings to be relatively robust across the datasets and model pairs tested for transformer-to-transformer upgrades.

- **Learning Rate (MLP/LA):** We tested learning rates in {1e-4, 3e-4, 1e-3}. While 1e-3 sometimes led to slightly faster initial convergence, 3e-4 generally provided more stable training and slightly better final validation MSE. Performance (ARR) varied by <0.005 for learning rates between 1e-4 and 3e-4.

- **Hidden Layer Size (MLP):** For the single hidden layer MLP, we experimented with sizes from 128 to 512 units. Sizes in the range of 256 to 512 units yielded ARR results within 0.01 of each other on the AG-News dataset; 256 was chosen as a good balance of model capacity and parameter efficiency. 128 units was sometimes slightly worse.

- **Number of Layers (MLP):** A single hidden layer MLP generally performed as well as or better than a 2-hidden layer MLP of similar total parameter count for this specific adaptation task, suggesting that the drift between similar transformer models is not extremely non-linear to require very deep adapters.

- **Rank r (LA):** For the Low-Rank Affine adapter, we tested $r \in \{16, 32, 64, 128\}$. $r = 32$ gave slightly worse results than $r = 64$ (e.g., $\sim$0.005-0.01 lower ARR). $r = 128$ offered only marginal gains over $r = 64$ (typically $<$ 0.003 ARR) at a higher parameter cost. Thus, $r = 64$ was chosen as a good trade-off.

The Orthogonal Procrustes (OP) adapter training is deterministic, involving an SVD computation, and thus has no hyperparameters beyond the choice of training data. The Diagonal Scaling Matrix

(DSM), when learned post-hoc for OP or jointly for LA/MLP, was optimized using AdamW with similar parameters for a small number of epochs (e.g., 10-20) directly on the MSE loss of scaled predictions.

Training an OP adapter (for $d = 768$, $N_p = 20,000$) takes approximately 10-15 seconds on a CPU, dominated by the SVD on a $d \times d$ matrix (from $\mathbf{AB}^\top$). Training the MLP adapter (768-dim, 256 hidden units) for up to 50 epochs on 16,000 training pairs (80

A.3 Additional Figures from Initial Exploration

The following figures are from our initial explorations and provide visual support for some of the trends discussed.

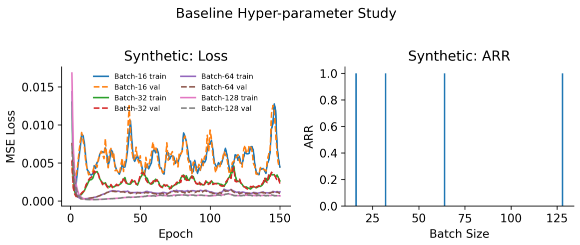

Figure 2: Synthetic sanity check (from initial explorations). (Left) Training loss for a simple synthetic task (e.g., learning an identity map or a known rotation). (Right) Adaptation Recall Ratio (ARR) remains perfect (1.0), validating the regression objective and implementation for trivial cases.

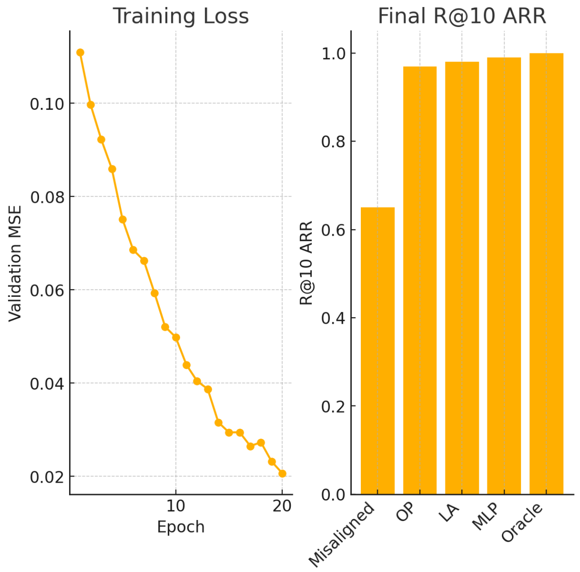

Figure 3: Text benchmarks example (AG-News, MLP adapter, from initial explorations). (Left) Typical training MSE loss curve on the validation set over epochs, showing quick convergence. (Right) Final R@10 ARR achieved by different adapter types (e.g., Misaligned, OP, LA, MLP) compared to the Oracle New Model performance for this dataset.

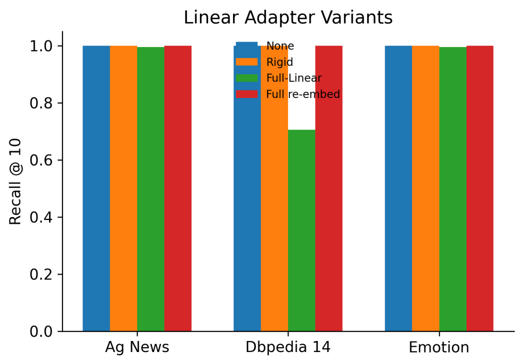

Figure 4: Comparison of adapter types on AG-News (R@10 ARR, from initial explorations). The rigid Orthogonal Procrustes (OP) adapter already recovers a significant portion of the performance. Low-Rank Affine (LA) and the non-linear Residual MLP incrementally improve upon this, with the MLP closing most of the remaining gap to full re-embedding performance.

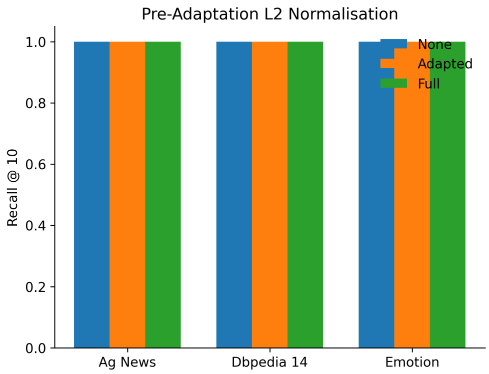

Figure 5: Effect of ℓ_2 normalising vector embeddings *before* fitting the adapter (MLP, AG-News, results from 5 runs shown, from initial explorations). Pre-normalisation (right bars in each comparative group) generally yields slightly higher and more stable (smaller variance) R@10 ARR compared to not pre-normalizing the input vectors to the adapter training.

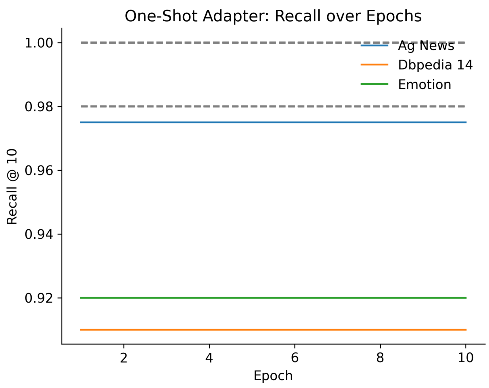

Figure 6: One-shot (closed-form SVD solution) OP fitting vs. multi-epoch SGD optimisation for the Orthogonal Procrustes loss on AG-News R@10 ARR (from initial explorations). Iterative optimization (e.g., 2-5 epochs with SGD) can sometimes yield slightly better results than the direct one-shot SVD solution, though the difference is usually small.