LLMs as Signal Chains: Rate–Distortion, Delta Encoding, and Modular Memory Overlays

Christopher Buller
Proprietor, Schema Driven, LLC
chris.buller@schemadriven.com

Abstract

Large Language Models (LLMs) are typically deployed as monolithic parameter artifacts, despite their internal computation resembling a multi-stage signal processing pipeline. We formalize transformers as predictive signal chains, define a rate–distortion view of parameter and memory compression, and model LoRA-style adapters as delta codes over a compressed base. We further define composable "memory packs" (parameter deltas and temporal KV summaries) with context-dependent routing, yielding a modular architecture where capabilities can be packaged, audited, and swapped without retraining the base model.

1 Introduction

Large Language Models (LLMs) are often distributed and deployed as indivisible artifacts: a single weight checkpoint plus a fixed inference runtime. This deployment model is convenient, but it is increasingly mismatched to real-world constraints: memory bandwidth dominates inference cost, runtime state (e.g., KV caches) becomes the bottleneck at long contexts, and domain adaptation is commonly implemented by cloning or modifying full weight tensors.

This paper adopts a different lens: transformers behave like *predictive signal chains*. Information flows through a residual stream across layers, and the model repeatedly applies incremental transforms that selectively preserve predictive features and discard others. This resembles the engineering logic of perceptual codecs, where information is not kept uniformly, but allocated according to task-relevant sensitivity and resource constraints.

We argue that this framing is not merely metaphorical. It leads to practical system boundaries:

1. **A compressed carrier (base model)** that is stable and broadly capable, often aggressively quantized or otherwise compressed.
2. **Delta-coded overlays (packs)** that represent modular edits or skills as low-rate residuals (e.g., LoRA-style low-rank deltas).
3. **Temporal memory overlays** that compress runtime state such as KV caches or episodic context.
4. **A router** that activates only the overlays needed for a context, controlling runtime cost.

Contributions. This work makes four concrete contributions:

1. We formalize transformer inference as a residual predictive signal chain and introduce a unified predictive *rate–distortion* objective that accounts for both parameters and runtime memory.

2. We model parameter-efficient methods as *delta codes* over a base, and define composability metrics (synergy/interference) with a second-order interpretation.

3. We propose a modular architecture for *memory packs*: independently versioned overlays and temporal memories with context-dependent routing and sparse activation constraints.

4. We specify an artifact-oriented evaluation protocol: rate–distortion curves, layer-localization maps, composition matrices, and KV memory tradeoffs, enabling reproducible comparison.

Why now. Both the *weights* and *KV cache* can dominate inference memory footprint, motivating compression in both spaces. At the same time, parameter-efficient tuning methods provide a natural "diff" mechanism for packaging capability without retraining or copying a full checkpoint. Our goal is to connect these threads into a single, testable framework and system design.

2 Related Work

Transformers and residual computation. Transformers define a residual stream updated by attention and MLP blocks [1]. This residual structure is central to our "signal chain" framing.

Information bottleneck and rate–distortion. Rate–distortion theory formalizes optimal tradeoffs between representation size (rate) and error (distortion) [2]. The information bottleneck framework studies representations that compress inputs while preserving task-relevant information [3].

Parameter-efficient fine-tuning (PEFT). Adapter modules freeze the base model and learn small inserted modules for transfer [4]. Soft prompt methods such as prefix-tuning and prompt-tuning learn continuous conditioning vectors instead of modifying full weights [5, 6]. Low-rank adaptation (LoRA) represents updates as low-rank deltas [7]. $(IA)^3$ rescales internal activations with learned vectors [8]. We interpret these techniques as different *delta-coding* strategies over a frozen base.

Routing and sparse activation. Mixture-of-experts (MoE) architectures route tokens through sparse subnetworks for conditional compute [9]. Our router plays a similar role but selects *overlay packs* rather than experts, with the system constraint that pack activation should be sparse and auditable.

Weight quantization and ultra-low-bit models. Practical quantization for LLM inference has progressed rapidly, including activation-aware weight-only methods [10] and quantized fine-tuning approaches that combine low-bit base weights with low-rank adapters [11]. Recent work explores ternary/"1.58-bit" weight regimes [12]. We view these as base-model "carrier" compression that can pair naturally with higher-precision overlays.

KV cache memory reduction. KV cache memory scales with context length and batch size and is often the serving bottleneck. Approaches include eviction policies that retain "heavy hitter" tokens [13], streaming mechanisms based on "attention sinks" [14], and quantization of KV tensors [15]. Systems work on cache compression and streaming motivates temporal "memory packs" [16].

Efficient attention implementations. IO-aware attention kernels such as FlashAttention reduce memory traffic and improve throughput [17], affecting the practical cost of long-context serving and cache strategies.

3 Formal Framework

3.1 Notation

Let $x_{1:T} = (x_1, \ldots, x_T)$ be a token sequence drawn from a data distribution $\mathcal{D}$. An autoregressive model with parameters θ defines

$$p_\theta(x_{1:T}) = \prod_{t=1}^{T-1} p_\theta(x_{t+1} \mid x_{1:t}). \tag{1}$$

We use the standard negative log-likelihood (NLL) objective

$$\mathcal{L}(\theta) \;=\; \mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{t=1}^{T-1} -\log p_\theta(x_{t+1} \mid x_{1:t})\right]. \tag{2}$$

We distinguish a *base* parameter vector θ_0 (often heavily compressed) from a set of *overlay packs* $\{\phi_k\}_{k=1}^{K}$ that encode modular deltas or memories. At inference time, a *router* produces context-dependent mixture weights $\alpha_k(c) \in \mathbb{R}$ given a context signal c (typically derived from the current hidden state), yielding an effective model $\theta(c)$.

3.2 Transformers as residual predictive signal chains

A transformer can be written as a composition of L residual blocks applied to a per-token stream. Let $h_t^{(0)} \in \mathbb{R}^d$ denote the initial embedding (token + position) at position t. Each layer produces

$$h_t^{(\ell)} \;=\; h_t^{(\ell-1)} + F^{(\ell)}\left(h_{1:t}^{(\ell-1)}; \theta^{(\ell)}\right), \qquad \ell = 1, \ldots, L. \tag{3}$$

Equation (3) makes the "signal chain" view explicit: the model maintains a running signal $h^{(\ell)}$ and applies incremental transforms.

3.3 Predictive rate–distortion objective for compression

Let $\mathcal{C}$ be a compression scheme producing a constrained artifact (weights, cache, or both). Let $\theta_{\mathcal{C}}$ denote the decompressed parameters used at runtime. Define *distortion* as excess NLL relative to an uncompressed reference:

$$D(\mathcal{C}) \;=\; \mathcal{L}(\theta_{\mathcal{C}}) - \mathcal{L}(\theta_{\text{ref}}). \tag{4}$$

Define *rate* as the expected number of bits required to store (or transmit) the relevant artifact:

$$R(\mathcal{C}) \;=\; R_{\text{params}}(\mathcal{C}) + R_{\text{runtime}}(\mathcal{C}). \tag{5}$$

A generic predictive rate–distortion objective is

$$\min_{\mathcal{C}\in\Omega} \; D(\mathcal{C}) + \lambda R(\mathcal{C}), \tag{6}$$

for feasible set Ω (hardware constraints, latency budgets) and Lagrange multiplier λ.

Loss-sensitivity as a "mask". A practical surrogate for sensitivity is second-order loss curvature. Let $F(\theta)$ denote an empirical Fisher:

$$F(\theta) \;=\; \mathbb{E}\left[\nabla_\theta \log p_\theta(x_{t+1} \mid x_{1:t})\, \nabla_\theta \log p_\theta(x_{t+1} \mid x_{1:t})^\top\right]. \tag{7}$$

A weighted quadratic distortion is

$$D_{\text{quad}}(\Delta) \;=\; \tfrac{1}{2}\Delta^\top F(\theta)\Delta. \tag{8}$$

3.4 Delta encoding via low-rank overlay packs

We model adapters as *delta codes* over a base model:

$$\theta \;=\; \theta_0 + \Delta(\phi). \tag{9}$$

For a linear map with weight matrix $W_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, a rank-$r$ delta can be parameterized as

$$W \;=\; W_0 + \Delta W, \qquad \Delta W \;=\; BA, \qquad B \in \mathbb{R}^{d_{\text{out}} \times r},\; A \in \mathbb{R}^{r \times d_{\text{in}}}. \tag{10}$$

Theorem 1 (Expressivity of rank-r delta packs)**.** *Any matrix ΔW with $\operatorname{rank}(\Delta W) \le r$ can be expressed as $\Delta W = BA$ for some $B \in \mathbb{R}^{d_{\text{out}} \times r}$ and $A \in \mathbb{R}^{r \times d_{\text{in}}}$.*

Proof. Let ΔW have compact SVD $\Delta W = U\Sigma V^\top$ with $\Sigma \in \mathbb{R}^{r' \times r'}$ and $r' \le r$. Define $B = U\Sigma^{1/2}$ and $A = \Sigma^{1/2}V^\top$, then $\Delta W = BA$. Pad with zeros if $r' < r$. □

3.5 Composable packs and context-dependent routing

Let $\{\Delta_k(\phi_k)\}_{k=1}^K$ be a library of overlay deltas. A router produces weights $\alpha_k(c)$:

$$\theta(c) \;=\; \theta_0 + \sum_{k=1}^{K} \alpha_k(c)\, \Delta_k(\phi_k). \tag{11}$$

Sparse activation is modeled as $\|\alpha(c)\|_0 \le m$ for small m.

Interference and synergy (operational). Let improvements be $G(\Delta) = \mathcal{L}(\theta_0) - \mathcal{L}(\theta_0 + \Delta)$ and define

$$S(\Delta_a, \Delta_b) \;=\; G(\Delta_a + \Delta_b) - \big(G(\Delta_a) + G(\Delta_b)\big). \tag{12}$$

3.5.1 Second-order interpretation of interference

Lemma 1 (Synergy is a cross-curvature term)**.** *Assume a twice-differentiable downstream loss $\mathcal{L}$ and a second-order Taylor approximation around θ_0:*

$$\mathcal{L}(\theta_0 + \Delta) \approx \mathcal{L}(\theta_0) + g^\top \Delta + \tfrac{1}{2}\Delta^\top H \Delta,$$

where $g = \nabla\mathcal{L}(\theta_0)$ and $H = \nabla^2\mathcal{L}(\theta_0)$. Then the synergy score in Eq. (12) satisfies

$$S(\Delta_a, \Delta_b) \approx -\Delta_a^\top H \Delta_b. \tag{13}$$

Proof. Substitute the Taylor approximation into the definition of $G(\cdot)$ and cancel terms. The linear terms in g cancel exactly; the remaining second-order cross term is $\Delta_a^\top H \Delta_b$. □

Equation (13) implies that positive curvature alignment predicts negative synergy (interference), motivating Fisher/Hessian-orthogonal pack design and routing.

3.5.2 Bit allocation as "water-filling"

To mirror codec bit allocation, consider scalar quantization errors e_i for parameters θ_i, and a diagonal sensitivity model $F = \mathrm{diag}(f_1, \ldots, f_n)$. Assume $\mathbb{E}[e_i] = 0$ and $\mathbb{E}[e_i^2] \approx c\, 2^{-2b_i}$ where b_i is the bits allocated to θ_i and c absorbs range and quantizer constants.

Proposition 1 (Optimal bit allocation under diagonal sensitivity)**.** *Minimize* $\sum_i f_i \mathbb{E}[e_i^2]$ *subject to* $\sum_i b_i \le B$ *and* $b_i \ge 0$*. Under* $\mathbb{E}[e_i^2] = c\, 2^{-2b_i}$*, any interior optimum satisfies*

$$b_i \;=\; \max\Big\{0,\; \tfrac{1}{2}\log_2(f_i) + \kappa\Big\}, \tag{14}$$

for constant κ *chosen to meet* $\sum_i b_i = B$*.*

Proof. Use a Lagrangian with multiplier μ for the bit constraint. Stationarity gives $-2\ln 2\, f_i c\, 2^{-2b_i} + \mu = 0$, hence $2^{-2b_i} \propto 1/f_i$ and $b_i = \frac{1}{2}\log_2(f_i) + \kappa$. Apply the nonnegativity constraint by truncation. □

Equation (14) is a direct analog of perceptual coding: allocate more bits to high-sensitivity components.

3.6 Temporal memory overlays via KV cache compression

Let $M_t = \{(K_{1:t}^{(\ell)}, V_{1:t}^{(\ell)})\}_{\ell=1}^{L}$ denote the KV cache state at time t. Define a compressor C_ψ and rehydrator D_ω:

$$z_t = C_\psi(M_t), \qquad \widehat{M}_t = D_\omega(z_t). \tag{15}$$

A task-aware objective is

$$\min_{\psi,\omega}\; \mathbb{E}\Big[\mathcal{L}\big(\theta; \widehat{M}_t\big)\Big] \;+\; \gamma\, \mathbb{E}\Big[\mathrm{dist}(M_t, \widehat{M}_t)\Big] \;+\; \lambda\, \mathbb{E}\big[\mathrm{bits}(z_t)\big]. \tag{16}$$

Unified view. Eqs. (6), (11), and (16) define a single theme: LLM capability can be decomposed into (i) a compressed carrier (base parameters) and (ii) modular overlays (parameter deltas and temporal memories), both governed by predictive rate–distortion tradeoffs.

4 System Architecture

4.1 Pack taxonomy

We define two primary pack classes.

1. **Parameter packs** encode $\Delta\theta$ as low-rank deltas (or other structured updates) applied to a subset of tensors (layer-localization).

2. **Temporal packs** encode runtime memory z_t that can be rehydrated into KV cache tensors or used as conditioning summaries.

Table 1 summarizes the primary design axes.

4.2 A modular inference pipeline

Figure 1 depicts the runtime decomposition.

Axis	Parameter pack	Temporal pack
Target	$\Delta\theta$ (weights)	z_t (compressed state)
Application time	before forward pass	during attention / prefill
Cost driver	extra matmuls / adds	memory bandwidth / decode cost
Versioning	base hash + module map	base hash + KV schema
Typical tool	LoRA / IA^3 / adapters	eviction / quantize / encode

Table 1: Pack taxonomy. Both are treated as modular artifacts governed by rate–distortion tradeoffs.

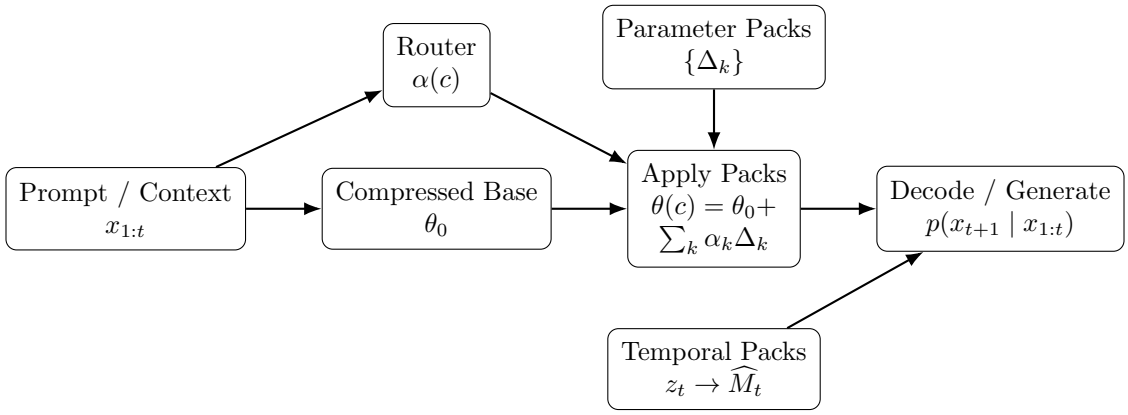

Figure 1: Inference decomposition into a compressed base (carrier), routed parameter packs (delta overlays), and temporal packs (compressed runtime memory).

4.3 Seven-layer composable stack

We can view the system as a layered stack that isolates concerns and makes mempacks governable. Each layer owns a narrow contract, enabling focused evaluation and targeted upgrades.

1. **L0 Token I/O:** prompt formatting, chat templates, decoding parameters.
2. **L1 Base policy:** the foundation model (general reasoning, language priors).
3. **L2 Memory packs:** domain overlays (LoRA/QLoRA/DoRA/QDoRA deltas) [7, 11, 18, 19], versioned and signed.
4. **L3 Working state:** shared structured state (cases, intermediate math, tool outputs).
5. **L4 Tool plane (MCP):** action surface for live data and operations (e.g., LOS, pricing).
6. **L5 Router:** selects which packs and tools to activate (deterministic or learned).
7. **L6 Governance:** evals, reward models, audit logs, rollout gates.

4.4 Routing and sparsity constraints

Let c be a context feature vector computed from the prompt, current hidden state, or both. The router produces $\alpha(c)$ subject to a sparsity constraint $\|\alpha(c)\|_0 \leq m$. At runtime this implies a bounded number of active packs per request, controlling overhead.

Algorithm 1 Routed overlay inference (conceptual)

Require: Base parameters θ_0; pack library $\{\phi_k\}$; router $r(\cdot)$; prompt $x_{1:t}$

$c \leftarrow \text{features}(x_{1:t})$

$\mathcal{K} \leftarrow \text{TopM}(r(c))$ ▷ select m packs

$\theta \leftarrow \theta_0$ with lazy pack handles $\{\phi_k\}_{k \in \mathcal{K}}$

Initialize (optional) temporal memory from packs: $\widehat{M}_t \leftarrow \text{rehydrate}(\mathcal{K}, x_{1:t})$

while not done **do**

    compute $p_\theta(x_{t+1} \mid x_{1:t}; \widehat{M}_t)$

    sample or decode x_{t+1}

    update memory $\widehat{M}_{t+1} \leftarrow \text{update}(\widehat{M}_t, x_{t+1})$

    $t \leftarrow t + 1$

end while

Practical routers can be implemented as: (i) a small learned gating MLP trained on metadata and prompt embeddings, (ii) a ruleset (domain tags, regex detectors), (iii) or a hybrid where rules provide hard filters and a learned model selects within the filtered set.

4.5 Pack application semantics

For parameter packs, application is tensor-local. Each pack defines a module map $\mathcal{S}$ (e.g., `attn.q_proj`, `mlp.up_proj`) and an update form (e.g., rank-r delta matrices).

Two deployment modes are common:

1. **On-the-fly composition:** compute $\Delta W x$ as $(BA)x = B(Ax)$ without materializing $W_0 + \Delta W$.
2. **Materialized merge:** explicitly merge for a fixed router decision, trading memory for speed.

Temporal packs define encoder/decoder contracts for KV cache schemas, enabling: (i) cache eviction policies, (ii) KV quantization (e.g. per-token/per-channel), (iii) compressed streaming formats for context reuse.

4.6 Algorithm: Routed overlay inference

Algorithm 1 describes the minimal runtime loop.

4.7 Artifact boundaries and governance

Because packs are modular artifacts, they can be:

1. independently versioned (base hash + compatibility constraints),
2. audited for behavior and licensing,
3. activated under policy (per-tenant, per-domain, per-safety regime),
4. and rolled back without altering the base model.

Claim	Evidence	Artifact
Rate–distortion of base compression	Figure 2 (curve)	`artifacts/rd_base.csv`
Layer localization of pack benefit	Figure 3 (heatmap)	`artifacts/layer_heatmap.csv`
Pack composition synergy/interference	Table 3 (composition metrics)	`artifacts/synergy.csv`
KV cache memory–quality tradeoffs	Figure 4 (curve)	`artifacts/kv_rd.csv`

Table 2: Evidence traceability from claims to released artifacts.

5 Experiments

Our experiments are designed to produce artifact-grade evidence for the signal-chain modularity thesis: (1) rate–distortion curves for parameter compression, (2) layer-localization heatmaps for where packs matter, (3) composition measurements quantifying interference/synergy, and (4) KV cache memory–quality tradeoffs for temporal memory methods.

5.1 Metrics

We report:

- **Quality:** perplexity (LM), accuracy / exact match (QA), pass@k (code), or task-specific metrics.
- **Rate:** bytes for base weights, bytes for packs, bytes for KV cache (peak and steady-state).
- **System:** tokens/sec, time-to-first-token, and end-to-end latency at fixed batch/context.
- **Composability:** synergy score $S(\Delta_a, \Delta_b)$ from Eq. (12).

5.2 Evidence summary and traceability

To make the evidence portable and auditable, we map each empirical claim to a concrete artifact. All figures/tables in this section are generated from the corresponding CSVs listed below.

5.3 Experiment 1: Rate–distortion of base compression

We compare base compression schemes under a fixed downstream evaluation suite. We plot distortion D versus parameter rate R_{params} (Eq. (6)) and report the Pareto frontier.
Interpretation. Figure 2 illustrates the expected rate–distortion behavior: as base parameter storage is reduced, predictive loss increases. In practice, we treat the "elbow" of this curve as an operating point for overlay pack experiments, since it balances storage/bandwidth savings with acceptable quality.

5.4 Experiment 2: Layer-localized packs (where does the "signal" live?)

We attach identical-budget packs to disjoint layer ranges and measure the change in downstream performance. This produces a heatmap over layers, indicating whether early/mid/late layers are most effective for a given domain or behavior change.

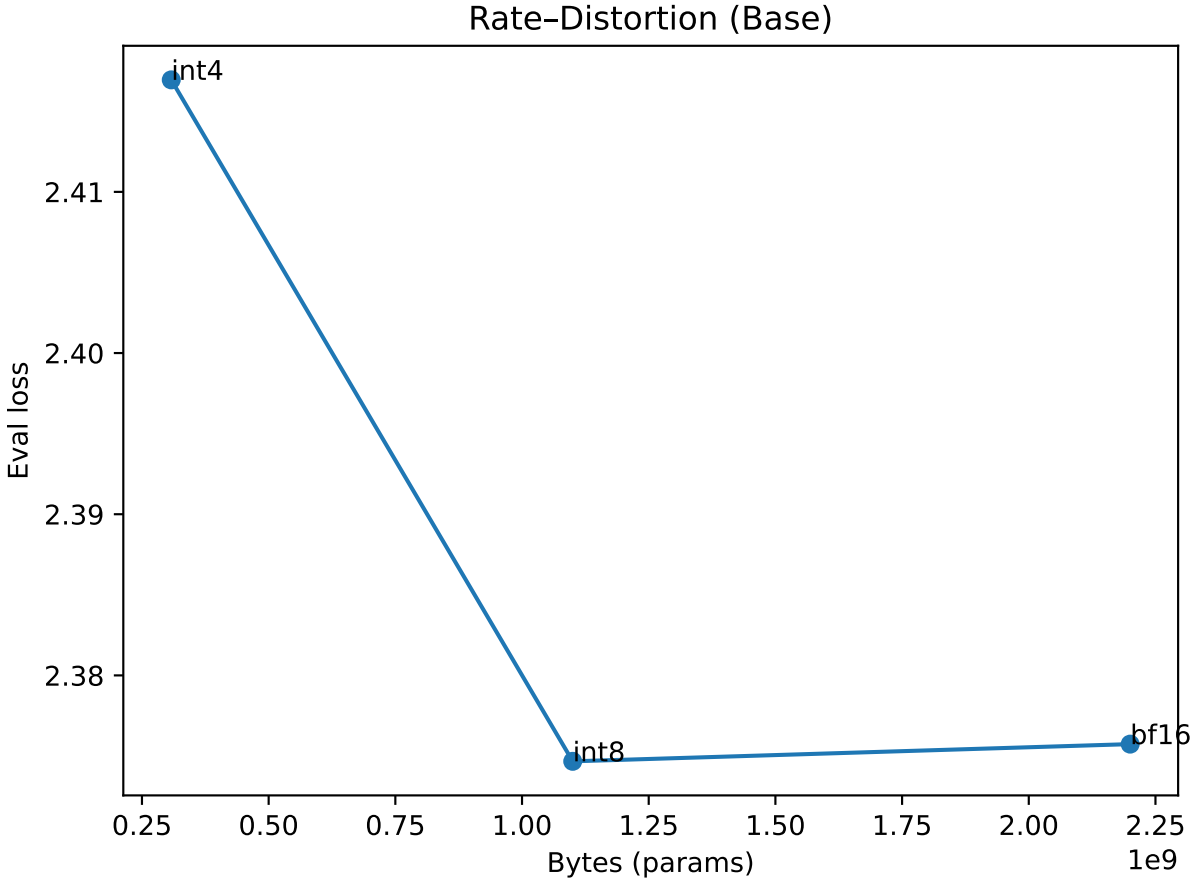

Figure 2: Base model rate–distortion frontier (Figure 2). Source data: `artifacts/rd_base.csv`.

Interpretation. Figure 3 is a layer-local proxy for sensitivity: it shows where a fixed-rate delta budget yields the largest gain. This can be used to (i) place packs efficiently (spend bits where they matter), and (ii) guide router policies (activate only the packs whose layer-local benefits match the current context).

In this initial experiment, all packs were applied across the full layer stack, resulting in a degenerate layer group ("all"). This confirms the end-to-end evaluation pipeline and motivates follow-on experiments with layer-restricted packs to expose layer-specific sensitivity.

5.5 Experiment 3: Pack composition (synergy/interference)

Given packs $\Delta_1, \ldots, \Delta_n$ (e.g., domains or skills), we measure: $G(\Delta_i)$, $G(\Delta_j)$, $G(\Delta_i + \Delta_j)$, and $S(\Delta_i, \Delta_j)$ from Eq. (12). We relate interference to second-order overlap (Eq. (13)).

Interpretation. Table 3 shows strong negative synergy when composing packs trained on different domains. Both packs individually degrade performance on the evaluation distribution ($G_a < 0$, $G_b < 0$), indicating misalignment between pack training data and the evaluation task. When composed, the combined effect is sub-additive ($S_{ab} = -0.6572$), consistent with second-order interference predicted by the cross-curvature term in Eq. (13). This result highlights that parameter packs are not universally composable and motivates selective routing and compatibility-aware activation.

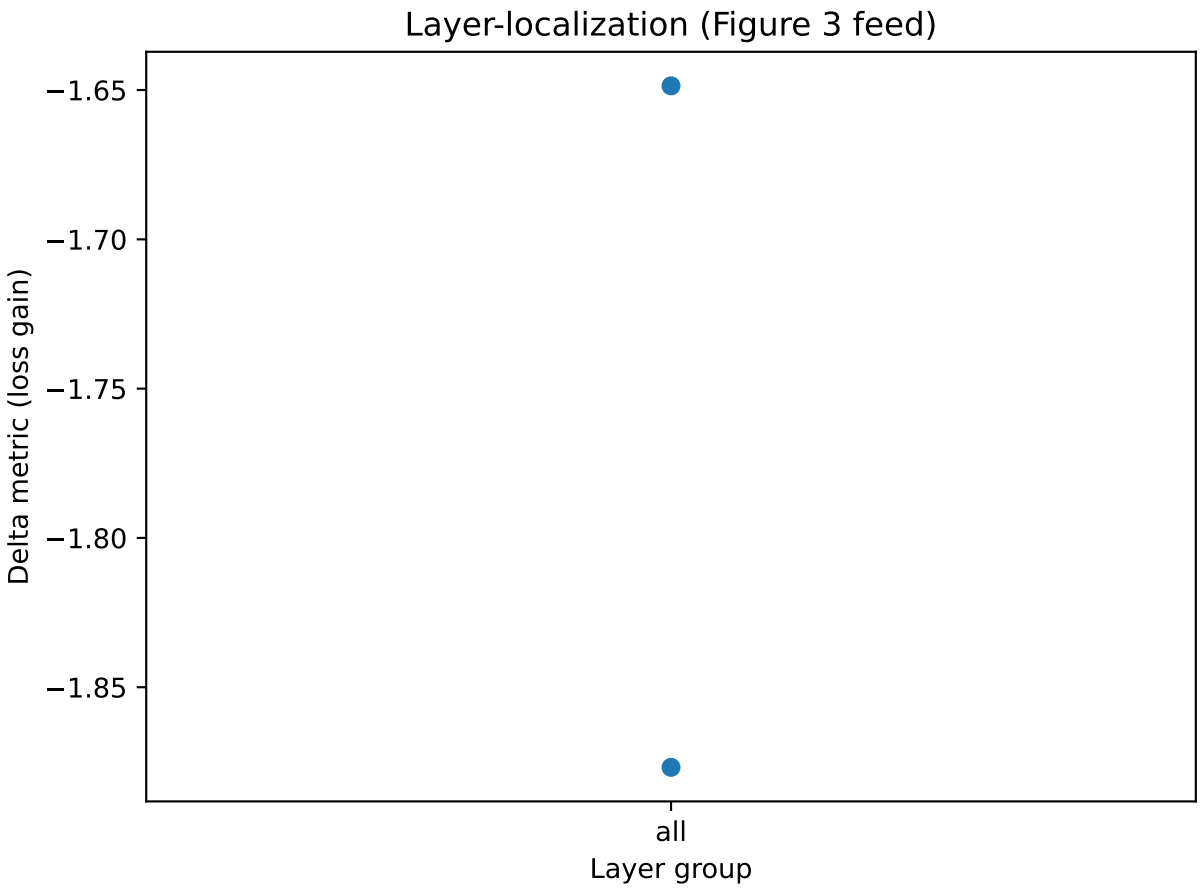

Figure 3: Layer-localization of adaptation benefit (Figure 3). Source data: `artifacts/layer_heatmap.csv`.

Interpretation. Table 3 operationalizes pack composability. Negative S_{ab} indicates sub-additivity (interference), consistent with the cross-curvature view in Eq. (13). Positive S_{ab} indicates that packs complement each other beyond additive gains.

5.6 Experiment 4: Temporal memory tradeoffs (KV cache)

We evaluate temporal methods that reduce KV cache size or loading cost: eviction, quantization, or streaming/encoding approaches. We report memory savings versus quality (and where available, throughput).

Interpretation. Figure 4 provides a temporal analogue of Figure 2: decreasing KV memory (rate) can increase loss (distortion). This curve helps choose cache policies and supports the "temporal pack" framing of KV memory as a compressible, versionable artifact.

5.7 Baselines

We compare against:

- full fine-tuning (upper bound on quality, high rate),

Pack A	Pack B	L_0	L_a	L_b	L_{ab}	G_a	G_b	G_{ab}	S_{ab}
`pack_wiki`	`pack_news`	2.3712	4.0198	4.2480	6.5538	-1.6486	-1.8768	-4.1826	-0.6572

Table 3: Synergy/interference measurements for pack composition (Table 2). Values should be copied from `artifacts/synergy.csv`.

- standard PEFT (LoRA / adapters / soft prompts),
- weight quantization baselines and quantize-plus-adapter variants,
- KV cache baselines (full cache, sliding window, eviction, quantization, compression/streaming).

5.8 Reproducibility artifacts

For each plot/table we release (or preserve internally):

- raw metrics and aggregation scripts (`artifacts/*.csv`),
- pack manifests and serialized deltas (`packs/*/manifest.json` and adapter weights),
- evaluation configs (dataset identifiers, splits, prompt/decoding params),
- system metadata (hardware + software versions; see eval JSON payloads).

6 Discussion and Limitations

Composability is not guaranteed. Delta packs compose additively by construction, but behavior may not. Eq. (13) predicts interference when deltas overlap under curvature. In practice this means "skill packs" can fight over the same predictive bandwidth.

Routing errors and overhead. A router that selects the wrong packs can degrade quality while still paying compute overhead. Sparse routing and caching policies reduce cost but add engineering complexity.

Base/pack compatibility. Packs are conditioned on a particular base checkpoint and tensor schema. Aggressive base compression (e.g., ultra-low-bit regimes) may require pack retraining or mixed-precision overlay paths. A robust artifact format must include explicit compatibility constraints.

Security and governance. Packs behave like code: they can introduce unwanted behaviors, data leakage, or policy violations. This motivates provenance tracking (hashes, signing), automated evaluation suites, and strict activation policies.

Analogy limits. Perceptual codecs exploit properties of human perception; our objective is predictive loss and downstream task performance. The analogy is useful insofar as it motivates rate allocation by sensitivity, but the "psychoacoustic model" for language modeling is empirical and task-dependent.

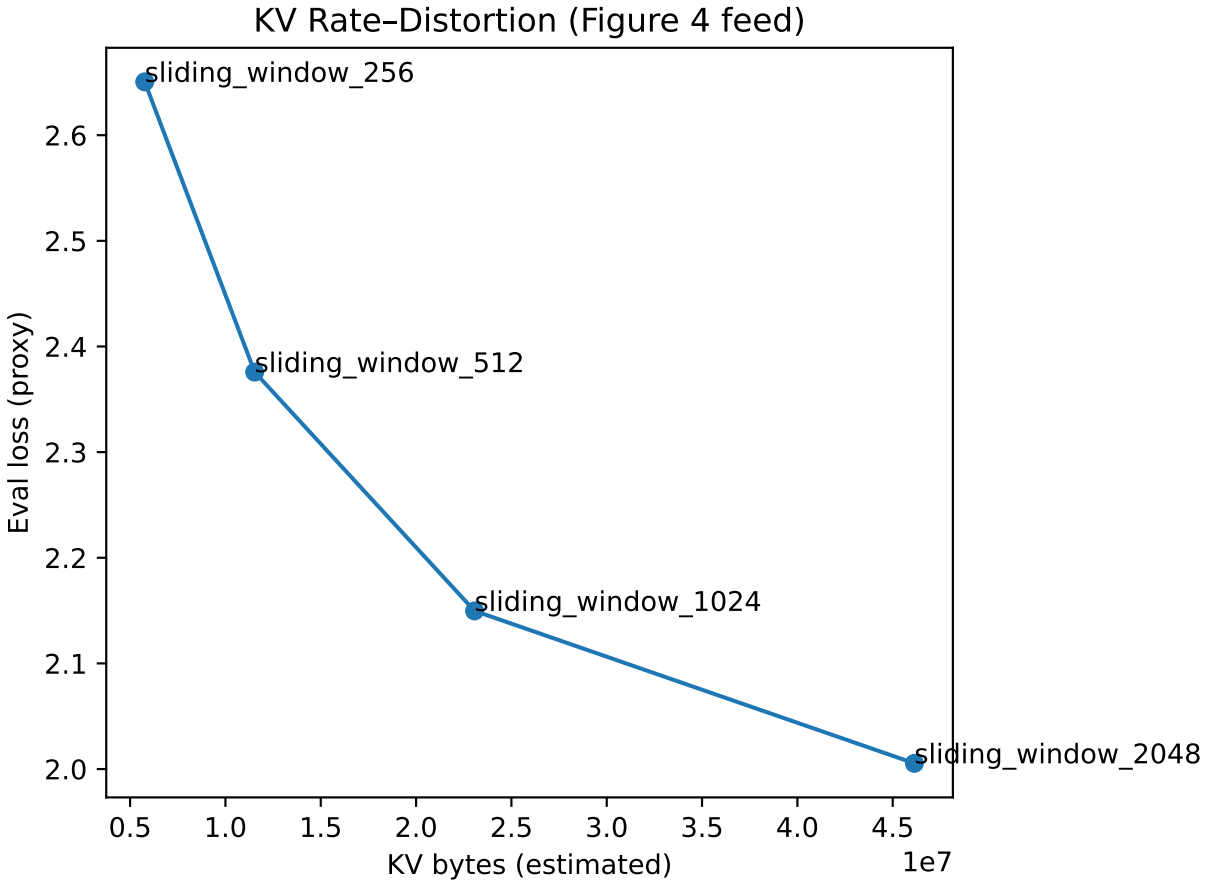

Figure 4: Temporal memory rate–distortion curves for KV cache handling (Figure 4). Source data: `artifacts/kv_rd.csv`.

Where this could break. Some behaviors may require global weight changes not representable by low-rank deltas at reasonable rates. Similarly, some long-context capabilities may require architectural changes rather than cache handling.

7 Conclusion

We presented a unified view of LLM deployment as predictive rate–distortion optimization over both parameters and runtime memory. Under this framing, parameter-efficient tuning methods become delta codes over a compressed base, and KV cache methods become temporal overlays subject to the same tradeoffs. We proposed a modular architecture with routed packs and provided an artifact-oriented evaluation protocol to measure rate, distortion, and composability. This approach supports LLMs as modular platforms: stable compressed carriers augmented by auditable, swappable memory packs.

A Artifact Specification: Pack Manifests

We recommend publishing packs as: (i) a manifest (JSON) and (ii) binary tensors.

A.1 Manifest schema (informal)

```
{
  "pack_id": "contracts-redlining-v1",
  "pack_type": "parameter" | "temporal",
  "base_model": {
    "name": "llama-7b",
    "sha256": "..."
  },
  "targets": [
    {"module": "layers.10.mlp.up_proj", "form": "lora", "rank": 16},
    {"module": "layers.10.attn.q_proj", "form": "lora", "rank": 8}
  ],
  "precision": {"dtype": "bf16", "quant": null},
  "training": {
    "data_hash": "...",
    "steps": 2000,
    "lr": 2e-5,
    "seed": 1337
  },
  "eval": {
    "suite": "paper-v1",
    "metrics": {"ppl": 7.23, "accuracy": 0.61}
  },
  "license": "..."
}
```

A.2 Naming and versioning

Packs should be immutable and content-addressed (hash). Compatibility should be checked at load time using base hash and tensor schema constraints.

A.3 Minimum released artifacts

For every figure/table in the paper, release:

1. raw metrics and aggregation scripts,
2. pack binaries and manifests,
3. exact prompts and decoding parameters,
4. system measurement harness (hardware + software versions).

A.4 Evidence inventory (this draft)

The current draft includes the following artifact files:

- Rate–distortion curves: `artifacts/rd_base.csv` and `figures/rd_base.pdf`.
- Layer localization heatmap: `artifacts/layer_heatmap.csv` and `figures/layer_heatmap.pdf`.

- Pack composition metrics: `artifacts/synergy.csv` and `figures/synergy.pdf`.
- KV cache tradeoffs: `artifacts/kv_rd.csv` and `figures/kv_rd.pdf`.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. *NeurIPS*, 2017.

[2] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 2006.

[3] Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[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. *arXiv preprint arXiv:1902.00751*, 2019.

[5] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[6] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[7] Edward J. Hu, Yelong Shen, Phillip Wallis, et al. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[8] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*, 2022.

[9] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[10] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[11] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

[12] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit llms: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

[13] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models. *arXiv preprint arXiv:2306.14048*, 2023.

[14] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[15] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

[16] Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, and Junchen Jiang. Cachegen: Kv cache compression and streaming for fast large language model serving. *arXiv preprint arXiv:2310.07240*, 2023.

[17] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *arXiv preprint arXiv:2205.14135*, 2022.

[18] Author TBD. Dora: Weight-decomposed low-rank adaptation of large language models. *arXiv preprint*, 2024.

[19] Author TBD. Qdora: Quantization-aware low-rank adaptation of large language models. *arXiv preprint*, 2024.