DON'T RETRAIN, REMIX: ADAPTATION OF WEB AGENTS USING MIXTURE HYPERNETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

The web is not static or uniform. Websites that serve the same purpose can look and navigate very differently. Even within a single site, successive redesigns can change how familiar features are presented and accessed. Humans adapt to this by recognizing familiar interaction patterns and reusing what they already know. We give web agents a similar ability with WEBMIX, a mixture hypernetwork that adapts them to unseen interfaces by composing previously trained low-rank adapters. WEBMIX begins with a bank of adapters trained on source interfaces. At deployment, a scout agent explores the target interface and summarizes its observations into a compact manual. A lightweight router uses this manual to predict mixing coefficients over the adapter bank, which combine the existing adapters into a new one for inference. Adapting to a new interface takes only a scouting run and a forward pass, without gradient updates on the target. With Qwen and Llama models, we evaluate *domain transfer* on TimeWarp and WebArena-Lite-v2, and *version transfer* on TimeWarp. In all three settings, WEBMIX matches or outperforms in-context learning and LoRA baselines, often matching the average success of specialists trained directly on those interfaces. On TimeWarp domain transfer, it gains about 8 percentage points on average over the best baseline with both models. These gains do not come at the cost of seen interfaces, where WEBMIX keeps its lead, and extend to version transfer. Adapting to a new interface can therefore rely on exploration and composition instead of retraining.

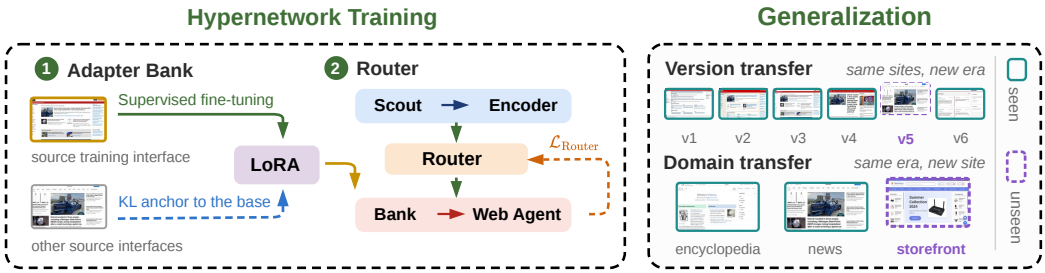


Figure 1: **WEBMIX at a glance.** We train a bank of interface-specific LoRA adapters, each KL-anchored to the base model on the other source interfaces. At deployment, a scout explores the unseen interface and writes a short manual. A router reads the encoded manual and mixes the bank into one adapter for the web agent. We test transfer to unseen domains and unseen versions.

1 INTRODUCTION

Humans compose and transfer skills to unfamiliar interfaces with little effort. A person who has booked flights on American Airlines' website can book the same flight through their credit card's travel portal, following the same search, filter, and checkout flow in a layout they have never seen. The same holds across time: when a site is redesigned, its regular users are slowed down for a few visits, not retaught. A fine-tuned web agent can do neither. Mind2Web (Deng et al., 2023) reports a steady drop in performance from its cross-task split to its cross-website and cross-domain splits. TimeWarp (Ishmam & Marino, 2026), which renders the same sites in the styles of six eras of web

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

design, shows that an agent trained on one era degrades on the next. Frontier models reduce this brittleness through scale (Zheng et al., 2024; He et al., 2024; Anthropic, 2024; AI, 2025; Müller & Žunič, 2024), but their closed weights and serving cost raise privacy and deployment concerns (Qi et al., 2024; Zhang et al., 2025).

Small open-weight models that run on commodity hardware are the practical alternative (Grattafiori et al., 2024; Qwen Team, 2026), and this brittleness hits them hardest. They become competent web agents only after fine-tuning on demonstrations, typically through a low-rank adapter (LoRA) (Hu et al., 2022) on a frozen base, and every new site or redesign calls for a new corpus and a new adapter. Their performance is therefore bounded by the demonstrations they are trained on, and web demonstrations are expensive to collect: each one requires a human operating a live site, and the site may be redesigned before the corpus is large enough. We therefore ask whether an agent can get more out of the demonstrations it already has. Facebook has altered its interface repeatedly since 2004, but it has always had a search bar. Amazon and Wikipedia serve different purposes entirely, yet browsing a catalog of products and browsing an encyclopedia of articles are the same operation: filtering, paging, and selecting from a list. If an adapter trained on one interface has already learned these operations, that knowledge ought to be transferable to the next.

Recent work adapts small web agents with less supervision. It reduces target-interface demonstrations to one or two examples (Verma et al., 2025), synthesizes trajectories through autonomous exploration of the target (Murty et al., 2024), transfers reusable action routines across websites (Wang et al., 2024), or learns lightweight alignment vectors from deployment-time interaction (Chen et al., 2025). All of these still treat each new interface as a separate adaptation problem. They reuse knowledge about websites, but none of them learns how to adapt, so the next interface starts from scratch again. A hypernetwork can learn such an adaptation rule, because it generates an adapter from a description of the target (Ha et al., 2017; Charakorn et al., 2025). For web interfaces, however, generating adapter weights directly is hard. A family of interfaces provides only a handful of conditionings from which to regress millions of weights, and LoRA factors are not identifiable, so adapters that behave alike can lie far apart in weight space (Charakorn et al., 2025).

We introduce WEBMIX, a mixture hypernetwork (Qu et al., 2023) that learns to adapt web agents by composing LoRA adapters trained on familiar interfaces. A scout briefly explores a new interface and summarizes its observations in a compact manual. Conditioned on this description, a small router predicts one mixing coefficient per adapter and combines their weight updates into a single adapter for the frozen base policy. We train the router on source demonstrations through the composed policy and keep the adapter bank fixed. Adaptation then requires only scouting and one forward pass, with no further gradient updates.

On TimeWarp (Ishmam & Marino, 2026), WEBMIX achieves 56.1% success on the unseen News domain with Qwen-3.5 4B, exceeding transferred LoRA by 10.6 percentage points. Across held-out domains, average gains reach 8.3 points for Qwen and 7.5 points for Llama-3.1 8B, nearly doubling Llama's success and matching the average performance of target-trained specialists for both models. Improvements extend to held-out TimeWarp versions and WebArena-Lite-v2 domains, where Qwen gains 3.2 and 4.6 points, respectively. The benefits also extend to seen interfaces: with target domains included in training, Qwen achieves 42.4% overall success on WebArena-Lite-v2, outperforming LoRA trained on all domains by 4.3 points.

Our contributions are the following:

- We introduce WEBMIX, a mixture hypernetwork that learns to compose existing LoRA adapters from a scouted interface description. The learned adaptation rule transfers to new websites and redesigns without parameter optimization on the target interface (§3).
- We evaluate domain transfer on TimeWarp and WebArena-Lite-v2 and version transfer on TimeWarp with two open-weight models (§4). On held-out TimeWarp domains, WEBMIX matches target-trained specialists on average for both models, reaches 56.1% success on unseen News with Qwen (+10.6 percentage points over transferred LoRA), and nearly doubles Llama's average domain-transfer success.
- We decompose held-out imitation risk into bank approximation and routing terms, clarifying the limits of adapter composition (§ D). Ablations compare composition with direct adapter generation and examine how scouting and adapter regularization contribute to transfer (§4.2).

2 Related Work

Web Agents and Interface Generalization. Web-agent benchmarks test generalization to unseen interfaces along different axes of variation. Along the cross-site axis, Mind2Web evaluates agents on static demonstrations split by task, website, and domain (Deng et al., 2023), while WebArena and WorkArena provide self-hosted, fully functional websites (Zhou et al., 2024; Drouin et al., 2024), and WebVoyager and Online-Mind2Web move evaluation to live websites (He et al., 2024; Xue et al., 2025). Along the temporal axis, TimeWarp evaluates agents on successive eras of web design (Ishmam & Marino, 2026). Complementary diagnostic suites isolate individual factors by perturbing the layout, interaction semantics, visual attributes, or network events of otherwise fixed tasks (Bai et al., 2026; Kara et al., 2025). On all of these benchmarks, agents lose accuracy on interfaces they were not trained on. Mind2Web reports substantially lower performance for fine-tuned agents on unseen websites and domains than on its cross-task split (Deng et al., 2023), and TimeWarp finds that text-based agents become brittle under interface changes after fine-tuning on a single version (Ishmam & Marino, 2026). A practical web agent should therefore adapt at test time to visually and functionally unfamiliar interfaces without relying on extensive demonstrations. We evaluate WebMix along both axes. TimeWarp provides both domain shifts and version shifts, and WebArena-Lite-v2 (Liu et al., 2024; 2026) provides domain shifts.

Adapting Web Agents to Unseen Interfaces. An agent facing an unfamiliar interface must learn something about it before it can act well. Prior work differs in where that knowledge comes from and where it is kept. The most direct source is a human: AdaptAgent adapts multimodal agents to new websites from up to two human demonstrations, using in-context demonstrations for proprietary models and meta-learned adaptation for open-weight models (Verma et al., 2025). Demonstrations, however, are costly to collect, so a second line of work lets the agent gather its own experience through guided or unguided exploration (Murty et al., 2024; Su et al., 2025; Pahuja et al., 2025; Sun et al., 2024). Beyond demonstrations and raw experience, an agent can also carry forward explicit knowledge. Non-parametric methods keep the weights fixed and store what the agent learns outside them: as reusable workflows (Wang et al., 2024) or episodic memories (Sarch et al., 2025), as an instruction manual written through interaction (Chen et al., 2024), as skills distilled from practice (Zheng et al., 2025; Wang et al., 2026b), or as summaries of environment dynamics learned at deployment time (Chen et al., 2025). Like AutoManual (Chen et al., 2024), WebMix constructs an explicit description of the environment from interaction. AutoManual uses this description to condition planning, whereas we use it to predict a composition of existing adapters. More generally, prior methods treat each new interface as a separate adaptation problem. WebMix instead learns an interface-conditioned rule for composing adapters on the source interfaces and applies it unchanged to the target interface. Target-side exploration supplies the conditioning information, without target demonstrations or test-time gradient updates.

Adapter Composition and Generation. Low-rank adaptation (Hu et al., 2022) lets many adapters share one frozen base model. A growing body of work reuses such collections instead of training a new model for each task (Yadav et al., 2024). The simplest form of reuse combines adapters in weight space. Merging methods add or average updates arithmetically (Ilharco et al., 2023; Yadav et al., 2023; Wortsman et al., 2022). Other approaches optimize composition coefficients on target examples, as in LoraHub (Huang et al., 2024), or compose adapters dynamically during inference through layer-wise gating in MoLE (Wu et al., 2024) and token-level routing in Arrow and PHATGOOSE (Ostapenko et al., 2024; Muqeeth et al., 2024). A more ambitious alternative skips the bank altogether and generates the adapter with a hypernetwork (Ha et al., 2017), conditioned on contexts and documents (Charakorn et al., 2026), task descriptions and prompts (Charakorn et al., 2025; Liang et al., 2025), whole code repositories (Hotsko et al., 2026), or per-sample memories in continual learning (Wang et al., 2026a). Generation is flexible, but it is hard to learn. The generator must regress every entry of every LoRA factor, and because LoRA factors are not identifiable, many different factors encode the same update. This is manageable with abundant conditionings but not with few: Drag-and-Drop LLMs, for example, improve little over their training adapters when trained on only a few datasets (Liang et al., 2025), and a family of web interfaces provides only a handful. WebMix sits between these two lines. Like a hypernetwork, it maps a description of the target to an adapter in one forward pass. Like merging, it builds that adapter from a frozen bank. It predicts only one coefficient per bank adapter and emits a single standard LoRA adapter that an

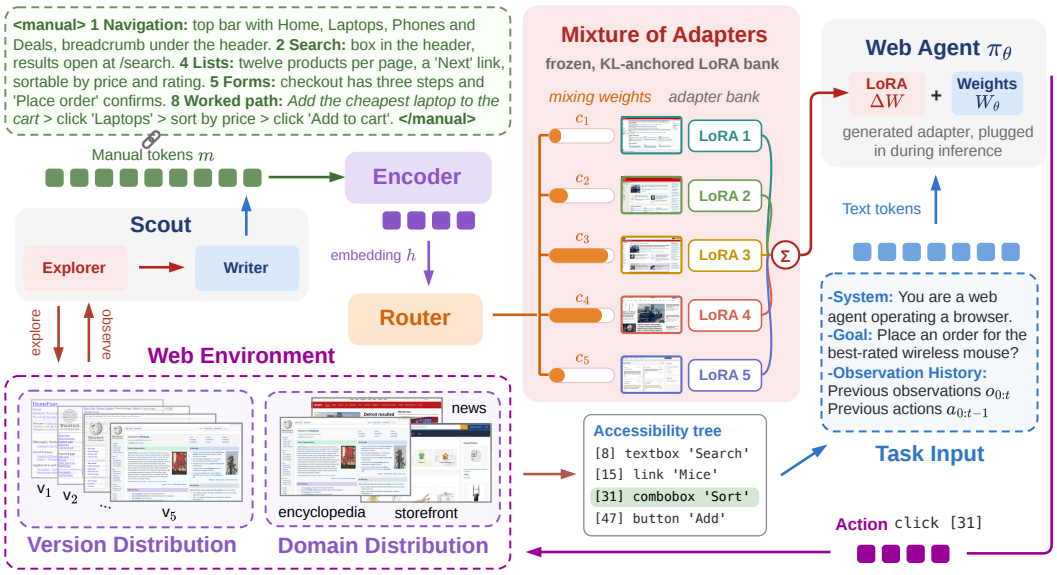


Figure 2: **Overview of WEBMIX.** A scout explores the target website. Its Explorer interacts with the environment, and its Writer turns the observations into a compact manual m. An encoder maps m to an embedding h, from which a router predicts mixing weights $c_1, \ldots, c_5$ over a frozen bank of KL-anchored LoRA adapters. Their weighted sum is a site-specific adapter ΔW, added to the base weights W_θ of the agent π_θ, which reads the accessibility tree and emits grounded actions (e.g., `click [31]`). Websites vary across versions (v_1–v_5) and domains.

unmodified inference stack can serve. We compare against Text-to-LoRA conditioned on the same interface representation, following Hotsko et al. (2026), and analyze formally in § D.2.

3 WEBMIX: ADAPTATION THROUGH COMPOSITION

Figure 2 gives an overview. §3.1 states the problem, §3.2 describes the mixture hypernetwork, and §3.3 describes its training and its adaptation to an unseen interface.

3.1 PRELIMINARIES

A web agent is a language model that completes a task on a website through a browser. It reads a natural-language goal and the current page, and it emits one browser action at a time, such as a click, a typed string, or a scroll, until the task ends. A task succeeds if a verifier accepts the final state of the website. A web environment consists of several *interfaces*. Each interface is either a website domain, such as a wiki or an online shop, or a version, *i.e.*, a design of the website. Interfaces differ in presentation, navigation structure, and content. Web agents are often trained using parameter-efficient methods, such as Low Rank Adaptation (LoRA) (Hu et al., 2022). We now formalize the web environment, agents, LoRA, and our problem setting in the following paragraphs.

Web environment and agents. We model the agent's interaction with a web interface as a partially observable Markov decision process (POMDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathbb{T}, \mathbb{T}_0, \mathbb{O}, R_g \rangle$ (Kaelbling et al., 1998). A state $s \in \mathcal{S}$ comprises the website's front-end (external) and back-end (internal) states. An episode starts from $s_0 \sim \mathbb{T}_0$, and browser actions $a \in \mathcal{A}$ change the state according to $s_{t+1} \sim \mathbb{T}(\cdot \mid s_t, a_t)$. The agent receives only an observation $o_t \sim \mathbb{O}(\cdot \mid s_t) \in \mathcal{O}$, in the form of an accessibility tree (Zhou et al., 2024). A task has a natural-language goal g and a verifier $V_g : \mathcal{S} \to \{0, 1\}$, which defines the terminal reward $R_g(s_T) = V_g(s_T)$ at the final step T. The agent is a policy π_θ with parameters θ that selects each action from the interaction history, $a_t \sim \pi_\theta(\cdot \mid H_t)$, where $H_t = (g, o_{0:t}, a_{0:t-1})$. Let $\mathcal{E}$ be the set of interfaces of the environment. We index the source interfaces as $\mathcal{E}_{\text{src}} = \{v_1, \ldots, v_K\} \subset \mathcal{E}$, so $K = |\mathcal{E}_{\text{src}}|$, and write an unseen interface as $u \in \mathcal{E} \setminus \mathcal{E}_{\text{src}}$. Each source interface v has a demonstration corpus $\mathcal{D}_v = \{(x, y)\}$, and we write $\mathcal{D}_k = \mathcal{D}_{v_k}$. Each input x is

a history H_t, rendered as a prompt that contains the goal, the accessibility tree of the current page, and a record of the earlier steps. Each target $y = (y_1, \ldots, y_{|y|})$ is the expert's response, a token sequence that contains its reasoning and one browser action.

Low-rank adaptation (LoRA). Fine-tuning all of θ for each interface would store one copy of the model per interface. Low-rank adaptation (LoRA) (Hu et al., 2022) keeps θ frozen and learns a low-rank update to selected weight matrices. Let Ω index the adapted weight matrices layer-wise, and let $W_{\theta,\ell} \in \mathbb{R}^{d_\ell^{\text{out}} \times d_\ell^{\text{in}}}$ be the frozen matrix at $\ell \in \Omega$. LoRA replaces it with

$$W_\ell = W_{\theta,\ell} + \Delta W_\ell, \qquad \Delta W_\ell = \frac{\alpha}{r} B_\ell A_\ell, \tag{1}$$

where $A_\ell \in \mathbb{R}^{r \times d_\ell^{\text{in}}}$ and $B_\ell \in \mathbb{R}^{d_\ell^{\text{out}} \times r}$ are trainable factors, r is the adaptation rank, and the scale α sets the size of the update independently of r. A_ℓ is initialized randomly and B_ℓ is set to zero, so training starts from the base policy. We call the collection of updates $\Delta\theta = (\Delta W_\ell)_{\ell \in \Omega}$ an *adapter*, and write $\pi_{\theta+\Delta\theta}$ for the policy with the adapter installed. Adapters can be held separate from the base weights and applied at serving time, so many adapters can share one frozen base model. We adapt the attention projections (query, key, value, output) and the MLP projections (gate, up, down), with $r = 16$ and $\alpha = 32$.

Problem setting. We start from a frozen base model and a set of *source* interfaces, each with a corpus of expert demonstrations. Our goal is for the agent to succeed on an *unseen* interface. In *domain transfer*, it is an unseen domain, and in *version transfer*, it is an unseen version. The unseen interface has no demonstrations, and no parameter is optimized on it. Before the agent starts its tasks, it may only make a brief, bounded visit to the unseen interface, which we call *scouting*. We ask whether the parameters the agent needs on the unseen interface can be assembled from what it learned on the source interfaces, using this visit alone.

3.2 The Mixture Hypernetwork

A hypernetwork (Ha et al., 2017) is a network that outputs the parameters of another network. WebMix uses a hypernetwork F_ϕ with trainable parameters ϕ to produce the adapter for an interface from a bounded exploration record of it. For an unseen interface u with record E_u, F_ϕ outputs one update per adapted matrix, which is added to the frozen base weights,

$$\{\Delta W_{u,\ell}\}_{\ell \in \Omega} = F_\phi(E_u), \qquad W_{u,\ell} = W_{\theta,\ell} + \Delta W_{u,\ell}. \tag{2}$$

The updates $\Delta\theta_u = (\Delta W_{u,\ell})_{\ell \in \Omega}$ produce the adapted policy $\pi_{\theta+\Delta\theta_u}$, with all other parameters unchanged. The objective is to maximize task success on u without target demonstrations or parameter optimization on the target interface. F_ϕ has three parts, described below: a frozen bank of adapters, a small router with the trainable parameters ϕ, and a frozen scout that maps the record E_v of an interface v to a representation $h_v \in \mathbb{R}^{d_c}$.

The adapter bank. The bank holds one LoRA adapter per source interface. For each v_k, the bank contains factors $\{(A_\ell^{(k)}, B_\ell^{(k)})\}_{\ell \in \Omega}$ of rank r and scale α, with updates $\Delta W_\ell^{(k)} = (\alpha/r)\, B_\ell^{(k)} A_\ell^{(k)}$. We write $\Delta\theta^{(k)} = (\Delta W_\ell^{(k)})_{\ell \in \Omega}$ for adapter k, and $\pi^{(k)} = \pi_{\theta+\Delta\theta^{(k)}}$ for the policy using it. The bank $\{\Delta\theta^{(1)}, \ldots, \Delta\theta^{(K)}\}$ is the source knowledge that WebMix transfers to an unseen interface. All bank adapters share the base model, the index set Ω, the rank r, and the scale α, so they can be combined matrix-wise. The bank is trained once (Stage 1, §3.3.1) and stays frozen afterwards.

What the hypernetwork predicts. A direct hypernetwork, such as Text-to-LoRA (Charakorn et al., 2025), predicts every entry of every LoRA factor. The router of WebMix predicts only K mixing coefficients, one per bank adapter. It is a two-layer perceptron with parameters $\phi = (U_1, b_1, U_2, b_2)$ that maps h_v to a vector of mixture weights over the K bank adapters,

$$c_v = \text{softmax}(U_2\ \text{SiLU}(U_1 h_v + b_1) + b_2) \in \mathbb{R}^K, \tag{3}$$

where $U_1 \in \mathbb{R}^{d_h \times d_c}$, $b_1 \in \mathbb{R}^{d_h}$, $U_2 \in \mathbb{R}^{K \times d_h}$, $b_2 \in \mathbb{R}^K$, and d_h is the hidden width ($d_h = 256$, Appendix A.3). The coefficient $c_{v,k}$ is the weight of bank adapter k in the adapter for v. A single coefficient vector is shared across all adapted matrices. We initialize U_2 and b_2 to zero, so the untrained router weights all bank adapters equally.

How the bank is combined. The adapter for interface v is the weighted sum of the bank's updates,

$$\Delta W_{v,\ell} = \sum_{k=1}^{K} c_{v,k}\, \Delta W_{\ell}^{(k)}. \tag{4}$$

We compute this sum by concatenating the bank factors along their rank dimension,

$$A_{v,\ell} = \begin{bmatrix} c_{v,1} A_{\ell}^{(1)} \\ \vdots \\ c_{v,K} A_{\ell}^{(K)} \end{bmatrix}, \qquad B_{v,\ell} = \begin{bmatrix} B_{\ell}^{(1)} & \cdots & B_{\ell}^{(K)} \end{bmatrix}. \tag{5}$$

The product satisfies $B_{v,\ell} A_{v,\ell} = \sum_k c_{v,k} B_{\ell}^{(k)} A_{\ell}^{(k)}$. Averaging the two factors separately would instead weight each adapter's own term by $c_{v,k}^2$ and add cross terms $B_{\ell}^{(j)} A_{\ell}^{(k)}$ with $j \neq k$, which by Eq. 9 (Appendix A.1) depends on how each adapter happens to be factored. The composed factors have inner dimension $r_{\text{out}} = Kr$, and we set $\alpha_{\text{out}} = K\alpha$, so that $\alpha_{\text{out}}/r_{\text{out}} = \alpha/r$. The result is a standard LoRA adapter of rank at most Kr and scale α_{out}, and installing it through Eq. 2 adds the same update as Eq. 4 to the base weights. It is served by an unmodified LoRA inference stack.

The scout. The scout turns a bounded visit to an interface v into the representation h_v that the router reads. It has three parts: an explorer collects an exploration record E_v, a writer summarizes E_v into a text *interface manual* m_v, and a frozen sentence encoder ψ embeds m_v. None of the three is trained at any stage. Appendix A.5 describes each part. The scout returns two objects for an interface v. The first is the manual m_v, a readable text summary of the interface. The second is the conditioning vector, the normalized encoding of the manual,

$$h_v = \frac{\psi(m_v)}{\|\psi(m_v)\|_2} \in \mathbb{R}^{d_c}. \tag{6}$$

Only h_v enters the hypernetwork, where the router maps it to the mixing coefficients c_v of Eq. 3.

3.3 Training and Adaptation

WebMix assembles the parameters from low-rank adapters. It trains one adapter per source interface, keeps the adapters in a frozen bank, and learns a small network, the router, that decides how much of each adapter to use on a given interface. Learning and adaptation proceed in three stages:

1. **Stage 1: Adapter Bank Training (§3.3.1).** For each source interface, we fine-tune one adapter on its demonstrations and freeze it in the bank.
2. **Stage 2: Router Training (§3.3.2).** The router learns to map the scout's encoded manual of each source interface to mixing weights over the bank, by imitating its demonstrations through the frozen base model and bank.
3. **Stage 3: Adaptation (§3.3.3).** The scout writes a manual of the unseen interface, and the router mixes the bank into the agent's adapter for it in one forward pass.

Both training stages use only source-interface data, so the scouting visit is the only information from the unseen interface that WebMix uses. Adding a source interface later adds one adapter to the bank and retrains the router, leaving the existing adapters unchanged.

3.3.1 Stage 1: Adapter Bank Training

Each adapter k is trained with supervised fine-tuning (SFT) to reproduce the expert on its own interface. An anchoring term also limits how far the adapter moves the base policy on the other source interfaces. We call $\mathcal{D}_k$ the adapter's *own corpus*, and $\mathcal{D}_{\setminus k} = \bigcup_{j \neq k} \mathcal{D}_j$ its *anchor corpus*, which pools the demonstrations of the other $K-1$ source interfaces. Neither corpus contains data from an unseen interface. We minimize the objective

$$\mathcal{L}_{\text{bank}}^{(k)} = \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_k}\left[-\frac{1}{|y|}\sum_{i=1}^{|y|}\log \pi^{(k)}(y_i \mid x, y_{<i})\right]}_{\text{SFT on training interface}} + \beta \underbrace{\mathbb{E}_{(x,y)\sim\mathcal{D}_{\setminus k}}\left[\frac{1}{|y|}\sum_{i=1}^{|y|}\kappa_i^{(k)}\right]}_{\text{anchor on other source interfaces}}, \tag{7}$$

where $\beta \geq 0$ controls the anchoring strength and $\beta = 0$ recovers plain SFT. The first term is the SFT loss on the adapter's own interface. The second term penalizes changes to the base policy's likelihood of the demonstrated tokens of the other source interfaces, where $\kappa_i^{(k)}$ is a clipped estimate of the KL divergence (Appendix A.2).

3.3.2 Stage 2: Training the Router

Stage 2 trains the router with the bank frozen. Before training, the scout visits each source interface v_k once and returns its representation h_{v_k}. From h_{v_k}, Eqs. 3 and 4 construct the composed update $\Delta\theta_\phi(h_{v_k}) = (\Delta W_{v_k,\ell})_{\ell\in\Omega}$. Let $\pi_{\phi,k} = \pi_{\theta+\Delta\theta_\phi(h_{v_k})}$ denote the resulting policy. We train the router with the behavior-cloning (BC) loss averaged over all response tokens of the pooled source corpus,

$$\mathcal{L}_{\text{router}}(\phi) = -\frac{1}{N}\sum_{k=1}^{K}\sum_{(x,y)\in\mathcal{D}_k}\sum_{i=1}^{|y|}\log \pi_{\phi,k}(y_i \mid x, y_{<i}), \qquad N = \sum_{k=1}^{K}\sum_{(x,y)\in\mathcal{D}_k}|y|. \tag{8}$$

Prompt tokens are masked, so only the expert's response tokens contribute. Gradients reach the router through the composed updates and the frozen base model, so the router learns which mixtures reproduce the expert on each source interface. Only ϕ is trained. The base model, the bank, and the scout stay frozen, and the representations h_{v_k} are fixed inputs.

3.3.3 Stage 3: Adaptation

In Stage 3, the scout visits the unseen interface u once, with the same procedure and budget as for the source interfaces, and produces m_u and h_u. The router maps h_u to c_u through Eq. 3, Eq. 5 composes the factors, and Eq. 2 adds the resulting update $\Delta W_{u,\ell} = \sum_k c_{u,k}\,\Delta W_\ell^{(k)}$ to the frozen base weights. For version transfer, the visit performs no tasks. For domain transfer, the base policy attempts n tasks from the training split of u, and the writer reads their intents and answer-stripped reference procedures. Adaptation therefore costs one scouting visit and one router forward pass. It uses no demonstrations from u and no gradient updates.

4 Experiments

We answer four research questions. **RQ1 (transfer).** Does WebMix transfer to unseen domains and versions better than in-context learning and LoRA? **RQ2 (retention).** Does it keep its performance on the seen interfaces? **RQ3 (design).** Which design choices matter: composing a bank instead of generating weights, the scout, the scouting budget, and the KL anchor? **RQ4 (analysis).** What do the bank adapters and the router's mixtures look like?

4.1 Experimental Setup

Benchmarks and evaluation. We evaluate on (i) TimeWarp (TW) (Ishmam & Marino, 2026), a containerized web benchmark with three domains (Wiki, News, Shop), each rendered in six versions, and (ii) WebArena-Lite-v2 (WAL) (Liu et al., 2026), an updated version of WebArena-Lite (Liu et al., 2024), itself a curated subset of WebArena (Zhou et al., 2024), with five domains (Reddit, GitLab, CMS, Map, OSS). We evaluate *domain transfer*, in which the unseen interface is a new domain, on both benchmarks, and *version transfer*, in which it is a new version, on TW, holding out v1 and v5 in turn. We conduct all the experiments with the BrowserUse harness (Müller & Žunič, 2024), with non-rationale tokens and memory enabled.

Baselines. We compare WebMix with three baselines on two open-weight models, Qwen-3.5 4B (Qwen Team, 2026) and Llama-3.1 8B (Grattafiori et al., 2024): the base model zero-shot, in-context learning (ICL) (Brown et al., 2020), and LoRA (Hu et al., 2022). We evaluate each method in two settings. In the **U**nseen setting, the evaluated domain or version is left out of the training data and the in-context examples. In the **S**een setting, it is included. In the seen setting, we report LoRA trained only on the evaluated interface (in), which is a target-trained specialist, and LoRA trained on all interfaces (all). Each number is the average of three runs. § A.4 gives the training details.

Model		Method	Domains				Versions		
			Wiki	News	Shop	Avg.	v1	v5	Avg.
Qwen-3.5 4B		Zero-shot	19.35	30.30	20.99	22.92	17.48	19.09	18.28
	U	In-context Learning	18.28	40.91	20.99	25.42	19.74	20.39	20.06
	U	LoRA	17.20	45.45	18.52	25.42	26.54	22.01	24.27
	U	**WEBMIX**	**24.73**	**56.06**	**25.93**	**33.75**	**30.42**	**24.60**	**27.51**
	S	In-context Learning	20.43	43.94	23.46	27.92	20.39	20.06	20.23
	S	LoRA (in)	27.96	48.48	28.40	33.75	27.51	23.30	25.40
	S	LoRA (all)	29.03	51.52	25.93	34.17	26.86	24.27	25.57
	S	**WEBMIX**	**34.41**	**59.09**	**32.10**	**40.42**	**30.74**	**25.24**	**28.00**
Llama-3.1 8B		Zero-shot	4.30	6.06	12.35	7.50	5.83	7.44	6.63
	U	In-context Learning	6.45	4.55	9.88	7.08	6.80	5.50	6.15
	U	LoRA	5.38	12.12	8.64	8.33	18.77	15.21	16.99
	U	**WEBMIX**	**15.05**	**18.18**	**14.81**	**15.83**	**20.39**	**17.15**	**18.77**
	S	In-context Learning	6.45	6.06	14.81	9.17	7.44	7.77	7.61
	S	LoRA (in)	13.98	16.67	17.28	15.83	19.42	15.86	17.64
	S	LoRA (all)	11.83	22.73	16.05	16.25	20.06	17.80	18.93
	S	**WEBMIX**	**17.20**	**27.27**	**19.75**	**20.83**	**21.36**	**20.06**	**20.71**

Table 1: **TimeWarp results (RQ1, RQ2).** Success rate (%) on held-out domains and held-out versions. U/S: unseen/seen setting. LoRA (in) is trained on the evaluated interface only, and LoRA (all) on all interfaces. Avg. is pooled over all tasks. Best result per model and setting in **bold**.

Model		Method	Reddit	GitLab	CMS	Map	OSS	Overall
Qwen-3.5 4B		Zero-shot	19.30	22.22	22.86	14.10	26.52	21.86
	U	In-context Learning	22.81	25.56	21.90	15.38	33.33	24.89
	U	LoRA	26.32	30.00	27.62	17.95	37.88	29.22
	U	**WEBMIX**	**31.58**	**34.44**	**30.48**	**25.64**	**41.67**	**33.77**
	S	In-context Learning	29.82	31.11	27.62	17.95	34.09	28.79
	S	LoRA (in)	45.61	35.56	35.24	25.64	43.18	37.23
	S	LoRA (all)	40.35	38.89	38.10	24.36	44.70	38.10
	S	**WEBMIX**	**47.37**	**42.22**	**40.95**	**32.05**	**47.73**	**42.42**
Llama-3.1 8B		Zero-shot	5.26	12.22	6.67	0.00	6.06	6.28
	U	In-context Learning	1.75	10.00	3.81	0.00	6.82	4.98
	U	LoRA	5.26	17.78	13.33	0.00	17.42	12.12
	U	**WEBMIX**	**12.28**	**21.11**	**18.10**	0.00	**19.70**	**15.37**
	S	In-context Learning	3.51	8.89	7.62	2.56	9.85	7.14
	S	LoRA (in)	42.11	22.22	20.00	5.13	25.76	22.29
	S	LoRA (all)	36.84	25.56	21.90	6.41	**28.03**	23.59
	S	**WEBMIX**	**45.61**	**26.67**	**24.76**	**10.26**	**28.03**	**26.19**

Table 2: **WebArena-Lite-v2 results (RQ1, RQ2).** Success rate (%) on held-out domains. U/S: unseen/seen setting. Overall is pooled over all tasks. Best result per model and setting in **bold**.

4.2 RESULTS AND ANALYSIS

RQ1: Transfer to unseen interfaces. Tables 1 and 2 report transfer to **U**nseen domains and versions on TW and WAL, respectively. WEBMIX outperforms every baseline on both models and transfer types, often matching the performance of baselines trained on those interfaces. For example, it matches LoRA (in) on TW domains with both models. Gains are largest for domain transfer, at 7.5–8.3 percentage points (pp) over the best baseline on TW and 3.3–4.6pp on WAL. Version transfer gains are smaller (1.8–3.2pp), and LoRA is a strong baseline there. Transfer is strongest for domains that are similar to the training domains. With Qwen-3.5, News gains 26pp over zero-shot, compared to ∼5pp on Wiki and Shop. On WAL, OSS gains the most for both models (14–15pp).

HYPERNETWORK Design	DOMAIN			Avg
	Wiki	News	Shop	
T2L RECONS (B.2)	8.60	24.24	18.52	16.25
T2L SFT (B.3)	18.28	39.39	22.22	25.42
MIXTURE (3.2)	19.35	48.48	24.69	29.17
+SCOUT n=1	**24.73**	56.06	**25.93**	**33.75**
+SCOUT n=3	18.28	**57.58**	22.22	30.42
+SCOUT n=5	17.20	56.06	**25.93**	30.83

Table 3: **Hypernetwork design (RQ3).** Success rate (%) on unseen TimeWarp domains with Qwen-3.5 4B. T2L rows generate every adapter weight, MIXTURE composes the frozen bank, and +SCOUT conditions the router on the scout's manual built from n scouted tasks. Avg is pooled over all tasks. Best result per column in **bold**, second best underlined.

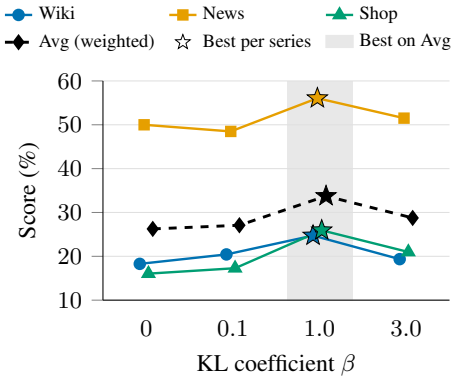


Figure 3: **KL anchor strength (RQ3).** Success rate (%) on unseen TimeWarp domains with Qwen-3.5 4B as the anchor weight β of Eq. 7 varies. The shaded column marks the best average, $\beta = 1.0$.

The one failure case is Map with Llama-3.1, where every unseen method scores zero. Composing adapters trained on other interfaces thus recovers most of the benefit of training on the target itself.

RQ2: Retention on seen interfaces. WEBMIX's transfer gains do not come at the cost of Seen interface performance. In Tables 1 and 2, WEBMIX is the best or tied-best method in every column. It improves over the strongest seen baseline, LoRA (all), by 4.6–6.3pp on TW domains, 1.8–2.4pp on TW versions, and 2.6–4.3pp on WAL. Training LoRA on all domains adds only 0.2–1.3pp over in-domain training, suggesting that WEBMIX's gains do not solely come from more training data.

RQ3: Design choices. *Hypernetwork design.* Table 3 isolates the contribution of each component. T2L RECONS (Appendix B.2), which regresses the stored adapter updates, is the weakest variant. T2L SFT (Appendix B.3), which trains the same generator through the policy, improves on it by 9.17pp. Replacing weight generation with the mixture hypernetwork (§3.2) adds a further 3.75pp, mostly from News (+9pp). Conditioning the router on the scout then adds another 4.58pp, for a total gain of 8.33pp over the strongest T2L variant. Composing a frozen bank is easier to learn from a handful of interfaces than generating weights, which is consistent with the analysis in Appendix D.2. *Scouting budget.* SCOUT outperforms MIXTURE on average for every number of scouted tasks n, but n=1 is best. Larger n gives a marginal gain on News (+1.5pp at n=3) but costs 6–8 points on Wiki, so we use n=1 throughout. *KL regularization.* Figure 3 varies the anchor weight β. β=1.0 is the best of the tested values on every domain and on average. Removing the KL term (β=0) lowers the average by over 7pp. A stronger penalty (β=3.0) also hurts. WEBMIX therefore benefits from moderate regularization.

RQ4: Analysis. The domain adapters of the bank are nearly orthogonal (cosine below 0.16, Figure 4a), and without the KL anchor they are about twice as similar (Figure 5). The router builds a held-out version from its nearest versions, e.g., v1 from v2 and v3 (Figure 4b). With v6 in the bank, it puts 96% of the weight on v6, so the mixture weights track interface similarity.

5 CONCLUSION

Experience with one website makes the next one easier to use, and WEBMIX gives small, open-weight web agents the same advantage. A scout writes a short manual of the new interface, and a router, trained only on source interfaces, maps the encoded manual to mixing weights over a frozen adapter bank. The result is a standard LoRA adapter, produced without gradient updates on the target. WEBMIX transfers to unseen domains and versions better than in-context learning and LoRA, often matches target-trained specialists. When the web changes, a web agent does not need retraining. It can simply *remix* what it already knows.

REFERENCES

Open AI. Computer using agent. https://openai.com/index/computer-using-agent/, 2025. [Accessed 20-09-2026].

Anthropic. Introducing computer use, a new claude 3.5 sonnet, and claude 3.5 haiku. https://www.anthropic.com/news/3-5-models-and-computer-use, 2024. [Accessed 20-09-2026].

Haoyue Bai, Dong Wang, Long Chen, Bingguang Hao, Pengyang Shao, Yonghui Yang, Yicheng He, and Chenyi Zhuang. Stressweb: A diagnostic benchmark for web agent robustness under realistic interaction variability, 2026. URL https://arxiv.org/abs/2604.16385.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Rujikorn Charakorn, Edoardo Cetin, Yujin Tang, and Robert Tjarko Lange. Text-to-LoRA: Instant transformer adaption. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Rujikorn Charakorn, Edoardo Cetin, Shinnosuke Uesaka, and Robert Tjarko Lange. Doc-to-lora: Learning to instantly internalize contexts, 2026. URL https://arxiv.org/abs/2602.15902.

Arthur Haonan Chen, Zuxin Liu, Jianguo Zhang, Akshara Prabhakar, Zhiwei Liu, Shelby Heinecke, Silvio Savarese, Victor Zhong, and Caiming Xiong. Grounded test-time adaptation for llm agents. *ArXiv*, abs/2511.04847, 2025. URL https://api.semanticscholar.org/CorpusID:286087642.

Minghao Chen, Yihang Li, Yanting Yang, Shiyu Yu, Binbin Lin, and Xiaofei He. Automanual: Constructing instruction manuals by llm agents via interactive environmental learning, 2024. URL https://arxiv.org/abs/2405.16247.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: towards a generalist agent for the web. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H. Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: how capable are web agents at solving common knowledge work tasks? In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala,

Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Ro-

driguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations (ICLR)*, 2017.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. *ArXiv*, abs/2401.13919, 2024. URL https://api.semanticscholar.org/CorpusID:267211622.

Liliana Hotsko, Yinxi Li, Yuntian Deng, and Pengyu Nie. Code2lora: Hypernetwork-generated adapters for code language models under software evolution, 2026. URL https://arxiv.org/abs/2606.06492.

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. In *International Conference on Learning Representations (ICLR)*, 2022.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition. In *First Conference on Language Modeling (COLM)*, 2024.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2212.04089.

Md Farhan Ishmam and Kenneth Marino. TimeWarp: Evaluating web agents by revisiting the past. In *Fortieth Conference on Neural Information Processing Systems Evaluations and Datasets Track*, 2026. URL https://arxiv.org/abs/2603.04949. To appear.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

Su Kara, Fazle Elahi Faisal, and Suman Nath. Warex: Web agent reliability evaluation on existing benchmarks. *ArXiv*, abs/2510.03285, 2025. URL https://api.semanticscholar.org/CorpusID:281843185.

Zhiyuan Liang, Dongwen Tang, Yuhao Zhou, Xuanlei Zhao, Mingjia Shi, Wangbo Zhao, Zekai Li, Peihao Wang, Konstantin Schürholt, Damian Borth, Michael M. Bronstein, Yang You, Zhangyang Wang, and Kai Wang. Drag-and-drop llms: Zero-shot prompt-to-weights. *arXiv preprint arXiv:2506.16406*, 2025. URL https://arxiv.org/abs/2506.16406.

Xiao Liu, Tianjie Zhang, Yu Gu, Iat Long Iong, Yifan Xu, Xixuan Song, Shudan Zhang, Hanyu Lai, Xinyi Liu, Hanlin Zhao, et al. Visualagentbench: Towards large multimodal models as visual foundation agents. *arXiv preprint arXiv:2408.06327*, 2024.

Zhaoyang Liu, JingJing Xie, Zichen Ding, Zehao Li, Bowen Yang, Zhenyu Wu, Xuehui Wang, Qiushi Sun, Shi Liu, Weiyun Wang, Shenglong Ye, Qingyun Li, Zeyue Tian, Gen Luo, Xiangyu Yue, Biqing Qi, Kai Chen, Bowen Zhou, Yu Qiao, Qifeng Chen, and Wenhai Wang. ScaleCUA: Scaling open-source computer use agents with cross-platform data. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yBFUqdJFZn.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. Learning to route among specialized experts for zero-shot generalization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Shikhar Murty, Dzmitry Bahdanau, and Christopher D. Manning. Nnetnav: Unsupervised learning of browser agents through environment interaction in the wild, 2024. URL https://api.semanticscholar.org/CorpusID:273162280.

Magnus Müller and Gregor Žunič. Browser use: Enable ai to control your browser, 2024. URL https://github.com/browser-use/browser-use.

Oleksiy Ostapenko, Zhan Su, Edoardo Maria Ponti, Laurent Charlin, Nicolas Le Roux, Lucas Caccia, and Alessandro Sordoni. Towards modular llms by building and reusing a library of loras. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Awadallah. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents. In *Annual Meeting of the Association for Computational Linguistics*, 2025. URL https://api.semanticscholar.org/CorpusID:276408442.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Xinyue Yang, Jiadai Sun, Yu Yang, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning. *ArXiv*, abs/2411.02337, 2024. URL https://api.semanticscholar.org/CorpusID:273821797.

Jingang Qu, Thibault Faney, Ze Wang, Patrick Gallinari, Soleiman Yousef, and Jean-Charles de Hemptinne. Hmoe: Hypernetwork-based mixture of experts for domain generalization, 2023. URL https://arxiv.org/abs/2211.08253.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Gabriel Sarch, Lawrence Jang, Michael J. Tarr, William W. Cohen, Kenneth Marino, and Katerina Fragkiadaki. Vlm agents generate their own memories: Distilling experience into embodied programs of thought, 2025. URL https://arxiv.org/abs/2406.14596.

John Schulman. Approximating kl divergence, Mar 2020. URL http://joschu.net/blog/kl-approx.html.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö. Arik. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. *ArXiv*, abs/2501.10893, 2025. URL https://api.semanticscholar.org/CorpusID:275757520.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. Os-genesis: Automating gui agent trajectory construction via reverse task synthesis. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/CorpusID:275119524.

Gaurav Verma, Rachneet Kaur, Nishan Srishankar, Zhen Zeng, Tucker Balch, and Manuela Veloso. AdaptAgent: Adapting multimodal web agents with few-shot learning from human demonstrations. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*

(Volume 1: Long Papers), pp. 20635–20651, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1008. URL https://aclanthology.org/2025.acl-long.1008/.

Yiran Wang, Chenyi Xiong, Ziyue Qin, Miao Zhang, Kui Xiao, and Zhifei Li. Hylovqa: Dynamic hypernetwork-generated low-rank adaptation for continual visual question answering, 2026a. URL https://arxiv.org/abs/2605.22035.

Zhaoyang Wang, Qianhui Wu, Xuchao Zhang, Chaoyun Zhang, Wenlin Yao, Fazle Elahi Faisal, Baolin Peng, Si Qin, Suman Nath, Qingwei Lin, Chetan Bansal, Dongmei Zhang, Saravan Rajmohan, Jianfeng Gao, and Huaxiu Yao. Webxskill: Skill learning for autonomous web agents, 2026b. URL https://arxiv.org/abs/2604.13318.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. *ArXiv*, abs/2409.07429, 2024. URL https://api.semanticscholar.org/CorpusID:272592995.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2404.13628.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 641–649, 2024.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Xiaodong Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. *ArXiv*, abs/2504.01382, 2025. URL https://api.semanticscholar.org/CorpusID:277502135.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. TIES-merging: Resolving interference when merging models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xtaX3WyCj1.

Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. What matters for model merging at scale? *ArXiv*, abs/2410.03617, 2024. URL https://api.semanticscholar.org/CorpusID:273162841.

Shu-Ning Zhang, Yu-Tong Jiang, Rongjun Ma, Yu-Ting Yang, Mingyao Xu, Zhixin Huang, Xin Yi, and Hewu Li. Privweb: Unobtrusive and content-aware privacy protection for web agents. *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, 2025. URL https://api.semanticscholar.org/CorpusID:281315205.

Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded. *ArXiv*, abs/2401.01614, 2024. URL https://api.semanticscholar.org/CorpusID:266741821.

Boyuan Zheng, Michael Y. Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gao-Wen Liu, Graham Neubig, and Yu Su. Skillweaver: Web agents can self-improve by discovering and honing skills. *ArXiv*, abs/2504.07079, 2025. URL https://api.semanticscholar.org/CorpusID:277634081.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Yonatan Bisk, Daniel Fried, Uri Alon, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations (ICLR)*, 2024.

A METHOD DETAILS

A.1 ADAPTER FACTORIZATION

The factors of an adapter are not unique. For any invertible $Q \in \mathbb{R}^{r \times r}$,

$$(QA_\ell,\ B_\ell Q^{-1}) \quad \text{induces} \quad (B_\ell Q^{-1})(QA_\ell) = B_\ell A_\ell, \tag{9}$$

so different factor pairs give the same update and the same policy. Two adapters trained on similar data can therefore lie far apart in factor space and still behave alike. WEBMIX combines adapters through their updates ΔW_ℓ, which Eq. 9 leaves unchanged. Every comparison we report is either functional (task success) or computed on the updates ΔW_ℓ (Figures 4a and 5), never on the factors.

A.2 KL ANCHOR

$$\begin{aligned} \tilde{d}_i^{(k)} &:= \mathrm{clip}\Big(\log \pi_\theta(y_i \mid x, y_{<i}) - \log \pi^{(k)}(y_i \mid x, y_{<i}),\ \tau_{\min},\ \tau_{\max}\Big), \\ \kappa_i^{(k)} &:= \exp\big(\tilde{d}_i^{(k)}\big) - \tilde{d}_i^{(k)} - 1, \end{aligned} \tag{10}$$

where $\tau_{\min} < 0 < \tau_{\max}$ are clipping bounds, set to $\tau_{\min} = -20$ and $\tau_{\max} = 8$. This penalty is nonnegative and vanishes when the two policies assign the same probability to the demonstrated token. It has the form of the k_3 estimator of the KL divergence (Schulman, 2020), but it is evaluated on demonstrated tokens instead of samples from $\pi^{(k)}$. It therefore anchors the probabilities of those tokens without constraining the full output distribution. Own-corpus examples contribute only the SFT term, and anchor-corpus examples contribute only the anchoring term. We sample the anchor examples from $\mathcal{D}_{\setminus k}$ to match the size of $\mathcal{D}_k$. The objective encourages each adapter to specialize to its own interface with less interference when the adapters are mixed.

A.3 ROUTER DETAILS

The router is the two-layer perceptron of Eq. 3 with hidden width $d_h = 256$ and a SiLU nonlinearity. It has $d_c d_h + d_h + (d_h + 1)K$ parameters, about 0.26M for $d_c = 1024$ and any $K \leq 5$ used here. Its output layer starts at zero, so the initial mixture is uniform and the initial adapter is the mean of the bank. One coefficient vector is shared across all adapted matrices.

A.4 TRAINING HYPERPARAMETERS

Bank adapters use rank $r = 16$ and scale $\alpha = 32$ on the attention and MLP projections, and the KL anchor uses $\beta = 1.0$ (Figure 3). The router has hidden width $d_h = 256$ and trains for 500 steps with AdamW. The scout's encoder is BGE-large-en-v1.5 (Xiao et al., 2024) with $d_c = 1024$. For domain transfer, the scout attempts $n = 1$ task (§4.2, RQ3). Table 4 lists the hyperparameters of every component.

A.5 SCOUT DETAILS

Explorer. For version transfer, the explorer runs a scripted, task-free, breadth-first crawl of same-origin links from each site's landing page. It visits up to 25 pages per site at a maximum link depth of 3. For every page it records the title, headings, link labels and targets, buttons, and form fields. It also performs one search probe, which submits the label of the first suitable link on the landing page to the first text or search field the crawl finds. For domain transfer, the explorer is task-grounded instead. The frozen base policy, without an adapter, attempts n tasks drawn from the training split of the site, and the explorer replays the pages that the policy visited.

Writer. A frozen language model writes the manual m_v from the exploration record. The manual follows a fixed schema of seven sections, which cover purpose and top-level navigation, search, page anatomy, lists and pagination, forms and multi-step flows, interaction pitfalls, and visual style. The writer is instructed to use only labels and paths observed during exploration, and to omit comparisons with other interfaces and any assessment of task performance. For version transfer, an interface spans several sites, and its manual concatenates the per-site manuals. For domain transfer, the writer also receives each attempted task's intent and its reference procedure with the final answer removed.

Component	Setting	Value
Bank adapters	rank r / scale α / dropout	16 / 32 / 0
	adapted matrices Ω	attention q, k, v, o and MLP gate, up, down
	epochs	6
	learning rate and schedule	10^{-4}, cosine, 10% warmup
	effective batch size	4 (1 per device $\times$ 4 accumulation)
	precision and hardware	bf16, DeepSpeed ZeRO-3, one 80 GB GPU
KL anchor	weight β	1.0
	clipping $[\tau_{\min}, \tau_{\max}]$	$[-20, 8]$
	anchor examples	as many as own-corpus examples
Router	hidden width d_h	256
	optimizer	AdamW, learning rate 10^{-4}, no weight decay
	schedule	20 linear warmup steps, then cosine
	steps / batch size / gradient clip	500 / 16 / 1.0
	maximum sequence length	8192 tokens
Scout	explorer budget	25 pages, link depth 3
	scouted tasks n (domain transfer)	1
	writer	Qwen-3.5 9B, temperature 0
	encoder	BGE-large-en-v1.5, $d_c = 1024$, 512 tokens
Agent decoding	temperature	0.01
	maximum new tokens per step	2048

Table 4: Training and evaluation hyperparameters.

The manual then adds a deterministic eighth section that lists, for each attempted task, its intent and the sequence of actions with the visible label of each target.

Encoder. A frozen sentence encoder ψ (`BGE-large-en-v1.5` (Xiao et al., 2024)) reads the tokens of the manual and returns their mean-pooled representation $\psi(m_v) \in \mathbb{R}^{d_c}$, with $d_c = 1024$.

B THE T2L BASELINE IN DETAIL

T2L (Charakorn et al., 2025) is a direct hypernetwork. It maps the same conditioning h_v that WEBMIX uses to the two LoRA factors of every adapted matrix. We index the adapted matrices by $s \in \Omega$ in this appendix and the next.

B.1 ARCHITECTURE

A query addresses one factor of one adapted matrix through three learnable embedding tables, indexed by the matrix's layer index l_s, its module type m_s, and the *role* ρ of the requested factor,

$$z_{s,\rho} \;=\; T_\phi\Big(\big[\, h_v;\; E_{\text{lyr}}(l_s);\; E_{\text{mod}}(m_s);\; E_{\text{role}}(\rho) \,\big]\Big), \qquad \rho \in \{\text{A}, \text{B}\}, \tag{11}$$

where each embedding has width 32 and the trunk T_ϕ is a linear projection to width $d_T = 128$ followed by a SiLU nonlinearity and two residual blocks. Output heads are shared at the coarsest granularity the target shapes allow. There is one linear head P_κ per distinct weight shape $(d^{\text{in}}, d^{\text{out}})$ realized in the target set, shared across layers and across both roles. The head emits $\max(r\, d^{\text{in}},\, r\, d^{\text{out}})$ values, and the leading block is reshaped into the requested factor,

$$A_s \;=\; \text{mat}_{r \times d_s^{\text{in}}}\big(P_{\kappa(s)}\, z_{s,\text{A}}\big), \qquad B_s \;=\; \text{mat}_{d_s^{\text{out}} \times r}\big(P_{\kappa(s)}\, z_{s,\text{B}}\big). \tag{12}$$

Because layer, module, and role enter as embeddings, all $2|\Omega|$ queries are batched and the full adapter is emitted in a single forward pass.

B.2 RECONSTRUCTOR

T2L RECONS follows the original training recipe of T2L (Charakorn et al., 2025). It regresses the stored bank adapters: for each source interface v_k, it minimizes the ℓ_1 distance between the

generated update $B_s A_s$ and the stored update $\Delta W_s^{(k)}$, scaled by 10^4, over 16 randomly drawn matrices per step. It trains for 2000 steps with AdamW (learning rate 10^{-4}, 100 warmup steps, cosine decay, gradient clipping at 1.0) from the initialization described below.

B.3 SFT

T2L SFT trains the same generator end to end through the frozen policy with the router objective of Eq. 8, so it receives exactly the training signal of WEBMIX. It uses the router's optimizer settings (Appendix A.4). Its output heads start at zero for B and at a LoRA-style random bias for A, so training starts at the base policy.

B.4 INITIALIZATION AT THE ORACLE'S SCALE

A generator that writes weights must write them at the right order of magnitude from step 0. Our trained per-version adapters have mean update magnitude $|\Delta W| \approx 5 \times 10^{-5}$. With every emitted entry i.i.d. $\mathcal{N}(0, \sigma_0^2)$, an entry of $B_s A_s$ sums r products of independent pairs, so its typical scale is $\sqrt{r}\,\sigma_0^2$. This is *quadratic* in σ_0, which is why an innocuous-looking initialization can emit updates orders of magnitude too large and swamp the frozen computation. We therefore set $\sigma_0 = (\varepsilon/\sqrt{r})^{1/2}$ with $\varepsilon = 10^{-6}$, which places the generated update well below the oracle scale at initialization. It is near a no-op, but without the exactly-zero factor that would stall the other factor's gradient through the product. Where WEBMIX starts at the mean trained adapter, T2L starts from almost nothing.

C ANALYSIS OF THE BANK AND THE ROUTER

This appendix answers RQ4 of §4: what the bank adapters and the router's mixtures look like.

The bank. Figure 4a shows that the domain adapters of the WAL bank are close to orthogonal. Every pairwise cosine similarity of their updates is below 0.16, so each adapter contributes a distinct direction to the mixture. Figure 5 measures the same similarity per layer. Adapters trained on the same domain become more similar with depth, reaching a cosine of about 0.5 (Qwen) and 0.6 (Llama) in the last layers. Adapters of different domains stay below 0.14 with the KL anchor, and are about twice as similar without it. The anchor therefore keeps the bank's adapters distinct, which is consistent with its gain in Figure 3.

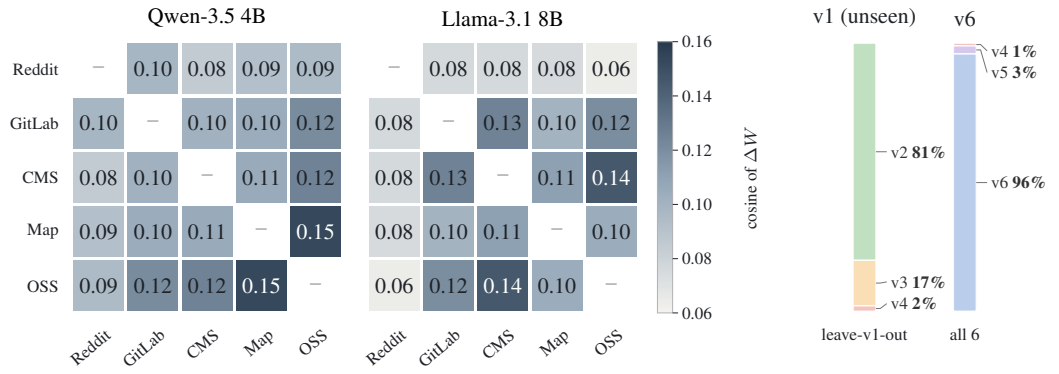
The router. Figure 4b shows the weights the router assigns on TW versions. With v1 held out, the router builds its adapter mostly from v2 (81%) and v3 (17%), so the weights fall with distance from v1 in time. With v6 in the bank, it puts 96% of the weight on the v6 adapter. The coefficients can thus be read as the model's judgment of which interfaces are similar (Appendix D.3).

D THEORETICAL ANALYSIS

We analyze the one design choice that separates WEBMIX from direct generation. The emitted adapter is confined to the convex hull of a frozen bank, and the interface chooses K coefficients rather than every weight. We ask what this confinement costs, what it buys, and how direct generation behaves in a tractable idealization. Full proofs are in Appendix E.

Setting. In this appendix, $s \in \Omega$ indexes the adapted matrices (written ℓ in §3, where ℓ is kept free for the logits in Appendix E), and $p_{\theta,\Delta W}$ is the policy $\pi_{\theta+\Delta\theta}$ written as a function of the update. We identify an adapter with its induced update $\Delta W = (\Delta W_s)_{s\in\Omega}$ in the product space $\mathcal{W} := \prod_{s\in\Omega} \mathbb{R}^{d_s^{\text{out}} \times d_s^{\text{in}}}$, with norm $\|\Delta W\| := (\sum_s \|\Delta W_s\|_F^2)^{1/2}$ and distance $\operatorname{dist}(\Delta W, \mathcal{H}) := \min_{W' \in \mathcal{H}} \|\Delta W - W'\|$ to a compact set $\mathcal{H}$. The bank is $\{\Delta W^{(k)}\}_{k=1}^K \subset \mathcal{W}$ with $\Delta W_s^{(k)} = \frac{\alpha}{r} B_s^{(k)} A_s^{(k)}$. For $c \in \Delta^{K-1}$ the mixture and its reachable set are

$$M(c) := \sum_{k=1}^{K} c_k\, \Delta W^{(k)}, \qquad \mathcal{H}_K := M(\Delta^{K-1}) = \operatorname{conv}\{\Delta W^{(1)}, \ldots, \Delta W^{(K)}\}, \tag{13}$$



(a) Pairwise cosine of ΔW (b) Router mixture weights

Figure 4: **The bank's domain adapters are close to orthogonal, and the router mixes nearby versions (RQ4). (a)** Pairwise cosine similarity of the weight updates $\Delta W = \frac{\alpha}{r} BA$ between the five domain adapters of the KL-anchored WebArena-Lite-v2 bank, computed over the whole model. All pairs stay below 0.16. Reddit is the least aligned with the other domains in both models, while OSS overlaps most with Map (Qwen) and CMS (Llama). **(b)** Mixture weights that the router assigns on TimeWarp versions. With v1 held out, it composes v1 mostly from the adjacent version v2 and then v3. With all six versions in the bank, it puts almost all weight on the matching v6 adapter.

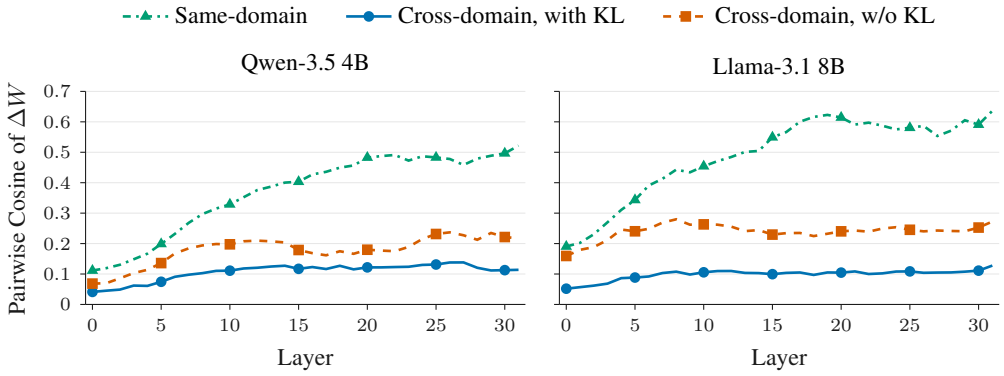


Figure 5: **The KL anchor keeps adapters of different domains apart (RQ4).** Cosine similarity of the weight updates ΔW per layer, averaged over pairs of adapters trained on the same domain, on different domains with the KL anchor, and on different domains without it.

and by Eq. 4, $M(c)$ is exactly the update WEBMIX serves under coefficients c. For an interface v let $p_{\theta,\Delta W}$ denote the likelihood of the frozen base with update ΔW installed, and let

$$\mathcal{R}_v(\Delta W) := \mathbb{E}_{(x,y)\sim\mathcal{D}_v}\big[-\log p_{\theta,\Delta W}(y \mid x)\big] \tag{14}$$

be the behavior-cloning risk of Eq. 8 on that interface. $\mathcal{R}_v$ is a finite average of smooth functions of the weights, so it is continuous on $\mathcal{W}$. The held-out interface is u, and its oracle $\Delta W^{(u)}$ is an adapter trained on the demonstrations of u itself, the LoRA (in) specialist of §4. We write $\hat{c} := c_u$ for the router output of Eq. 3 on the held-out embedding.

D.1 WHAT THE HULL CAN REACH

Assumption 1 (Lipschitz risk)**.** Let $\mathcal{K}_u := \mathrm{conv}\big(\mathcal{H}_K \cup \{\Delta W^{(u)}\}\big)$. There is a constant $\Lambda_u < \infty$ such that $|\mathcal{R}_u(\Delta W) - \mathcal{R}_u(\Delta W')| \le \Lambda_u \|\Delta W - \Delta W'\|$ for all $\Delta W, \Delta W' \in \mathcal{K}_u$.

The set $\mathcal{K}_u$ is convex and compact, and the base network is continuously differentiable in its weights, so by the mean value inequality the assumption holds with Λ_u equal to the largest gradient norm of $\mathcal{R}_u$ on $\mathcal{K}_u$. The constant can be large. We use the bound for its direction, not its value.

Theorem 1 (Hull decomposition of the held-out risk)**.** *Under Assumption 1, a minimizer* $c^\star \in \arg\min_{c\in\Delta^{K-1}} \mathcal{R}_u(M(c))$ *exists, and*

$$\mathcal{R}_u(M(\hat{c})) - \mathcal{R}_u(\Delta W^{(u)}) \;=\; \underbrace{\mathcal{R}_u(M(\hat{c})) - \mathcal{R}_u(M(c^\star))}_{\textit{routing}\,\geq 0} \;+\; \underbrace{\mathcal{R}_u(M(c^\star)) - \mathcal{R}_u(\Delta W^{(u)})}_{\textit{approximation}}, \quad (15)$$

where the approximation term satisfies

$$\begin{aligned} \mathcal{R}_u(M(c^\star)) - \mathcal{R}_u(\Delta W^{(u)}) \;&\leq\; \Lambda_u \operatorname{dist}\big(\Delta W^{(u)}, \mathcal{H}_K\big), \\ \mathcal{R}_u(M(c^\star)) \;&\leq\; \min_{k\leq K} \mathcal{R}_u(\Delta W^{(k)}). \end{aligned} \quad (16)$$

Takeaway: the excess held-out risk of WEBMIX *splits into a term set by the geometry of the bank and the target, which the head cannot change, and a term set by the head and the conditioning. The best mixture is never worse than the best single bank adapter transferred to the held-out version.*

Proof sketch. The decomposition is an identity. For the first bound, project $\Delta W^{(u)}$ onto the compact convex set $\mathcal{H}_K$. The projection is some $M(c_\Pi)$, and Lipschitz continuity on $\mathcal{K}_u$ gives $\mathcal{R}_u(M(c^\star)) \leq \mathcal{R}_u(M(c_\Pi)) \leq \mathcal{R}_u(\Delta W^{(u)}) + \Lambda_u \operatorname{dist}(\Delta W^{(u)}, \mathcal{H}_K)$. For the second, every vertex e_k lies in Δ^{K-1} and $M(e_k) = \Delta W^{(k)}$. □

Remark 1 (What the theorem predicts). The theorem is stated in behavior-cloning risk, and the benchmark reports task success. The bridge assumes the two move together. Three predictions follow. First, column u of the transfer matrix (whose entry (j, u) is the success of the adapter trained on interface j when evaluated on u) lists the performance of every per-version adapter on u, and its off-diagonal entries are exactly the bank adapters, so the gap between its best off-diagonal entry and its diagonal entry is an observable ceiling on the approximation term. If WEBMIX on u beats every off-diagonal entry of that column, the head is using a point of the hull that is not a vertex. Second, the uniform mixture $c = \mathbf{1}/K$ is a fixed element of $\mathcal{H}_K$, so its gap to WEBMIX isolates the routing term's gain over conditioning-blind routing. Third, $\operatorname{dist}(\Delta W^{(u)}, \mathcal{H}_K)$ is computable from the stored oracles alone as a K-variable least-squares problem over the simplex. The theorem predicts a small gap to the oracle when this distance is small. The eras are temporally ordered, so the oracle of an interior era should lie closer to the hull of the others than the oracle of a boundary era. A leave-one-out panel over all six versions, plotting the gap to the oracle against the hull distance, would test this. The experiments of §4 do not include the transfer matrix, the uniform mixture, or the hull-distance panel, so these three predictions remain untested.

Corollary 2 (Growing the bank)**.** *Let a bank of* $K+1$ *adapters extend the bank of* K *by one adapter* $\Delta W^{(K+1)}$*, and write* $M_K, \mathcal{H}_K$ *and* $M_{K+1}, \mathcal{H}_{K+1}$ *for the two mixtures and reachable sets. Then* $M_{K+1}((c, 0)) = M_K(c)$ *for every* $c \in \Delta^{K-1}$*, hence* $\mathcal{H}_K \subseteq \mathcal{H}_{K+1}$*, and for every version* v *and every* $\Delta W \in \mathcal{W}$*,*

$$\begin{aligned} &\operatorname{dist}(\Delta W, \mathcal{H}_{K+1}) \leq \operatorname{dist}(\Delta W, \mathcal{H}_K), \\ &\mathcal{R}_v^\star(K+1) \leq \mathcal{R}_v^\star(K), \qquad \textit{where} \quad \mathcal{R}_v^\star(K) := \min_{c\in\Delta^{K-1}} \mathcal{R}_v(M_K(c)). \end{aligned} \quad (17)$$

Takeaway: appending an adapter enlarges what the mixture can reach and can only lower the best achievable risk on every version, old or new. Any degradation on an old version after the bank grows is routing error, not lost capacity.

Remark 2 (Continual reading). This is the precise form of the zero-forgetting claim in §3.3. Under sequential arrival, report both $\mathcal{R}_v^\star(K)$, obtained by minimizing over c on $\mathcal{D}_v$ with the stored bank, and the risk of the realized head. Their difference is the forgetting the head is responsible for. The corollary also predicts a bank-size curve. With nested banks of size $2, \ldots, 5$ drawn from the training eras, each extending the previous one by one adapter, the best-mixture gap on the held-out era is non-increasing in K. The realized gap need not be monotone.

Corollary 3 (Convex twin)**.** *If* $\mathcal{R}_u$ *is convex on* $\mathcal{K}_u$*, then for every* $c \in \Delta^{K-1}$*,*

$$\mathcal{R}_u(M(c)) \;\leq\; \sum_{k=1}^{K} c_k\, \mathcal{R}_u(\Delta W^{(k)}), \quad \textit{in particular} \quad \mathcal{R}_u(M(\mathbf{1}/K)) \;\leq\; \frac{1}{K}\sum_{k=1}^{K} \mathcal{R}_u(\Delta W^{(k)}). \quad (18)$$

Takeaway: in the convex twin, any mixture is at least as good as the corresponding average of transferred bank adapters, and the uniform mixture is at least as good as the mean of the off-diagonal entries of the transfer-matrix column.

Remark 3 (Scope of the twin). Convexity holds whenever the adapted matrices feed the logits affinely, for example when only an output-side projection is adapted, because the negative log-likelihood is convex in the logits; the converse is not claimed. The 128-matrix adapter of Eq. 1 adapts matrices deep in the network, so for the paper's configuration the corollary is a simplification and its prediction is directional. It predicts that the uniform mixture on u performs at least as well as the mean off-diagonal entry of column u of the transfer matrix.

D.2 Direct generation in a frozen-trunk twin

The T2L baseline trains a trunk and per-shape heads and regresses the oracle updates. We analyze an idealization that keeps the regression and the heads and drops the rest.

Assumption 2 (Frozen-trunk twin)**.** A direct generator consists of a fixed feature map $\xi : \mathbb{R}^{d_c} \to \mathbb{R}^{d_\xi}$ and a linear head $\Gamma \in \mathbb{R}^{D_\Omega \times d_\xi}$ that emits the vectorized update directly, $\operatorname{vec} \Delta W_\Gamma(h) = \Gamma\, \xi(h)$, where $D_\Omega := \sum_{s\in\Omega} d_s^{\text{out}} d_s^{\text{in}}$. Write $\xi_k := \xi(h_{v_k})$, $\Xi := [\xi_1, \dots, \xi_K] \in \mathbb{R}^{d_\xi \times K}$, and $\mathcal{B} := [\operatorname{vec} \Delta W^{(1)}, \dots, \operatorname{vec} \Delta W^{(K)}] \in \mathbb{R}^{D_\Omega \times K}$. The features of the training versions are linearly independent, $\operatorname{rank} \Xi = K$, and the head is trained by gradient descent from $\Gamma = 0$ on the squared reconstruction loss $\sum_k \|\Gamma \xi_k - \operatorname{vec} \Delta W^{(k)}\|^2$.

Proposition 4 (Direct generation is unnormalized routing)**.** *Under Assumption 2, gradient descent with a sufficiently small step converges to $\hat{\Gamma} = \mathcal{B}\,\Xi^+$ with $\Xi^+ = (\Xi^\top \Xi)^{-1} \Xi^\top$, the unique minimum-norm interpolant of the training oracles. For every conditioning h,*

$$\operatorname{vec} \Delta W_{\hat{\Gamma}}(h) \;=\; \mathcal{B}\,\eta(h) \;=\; \sum_{k=1}^{K} \eta_k(h) \operatorname{vec} \Delta W^{(k)}, \qquad \eta(h) := \Xi^+ \xi(h) \in \mathbb{R}^K. \tag{19}$$

Hence (a) the output at any conditioning lies in the linear span of the bank, (b) the coefficients $\eta(h)$ depend on the feature geometry $(\Xi, \xi(h))$ alone and not on the bank or on the policy, and (c) $\eta(h)$ need not lie in Δ^{K-1}. For orthonormal ξ_1, ξ_2 and $\xi(h) = \xi_1 + \xi_2$ the output is $\Delta W^{(1)} + \Delta W^{(2)}$, twice the scale of the mean of the two, and for $\xi(h) = \xi_1 - \xi_2$ it is $\Delta W^{(1)} - \Delta W^{(2)}$.

Takeaway: in the twin, direct generation at a new interface is routing over the same bank, with coefficients that are neither normalized nor chosen through the policy. Routing is not the less expressive choice at zero-shot time. It is the same output family with a constraint on the coefficients.

Corollary 5 (Scale control)**.** *For every $c \in \Delta^{K-1}$ and every adapted matrix s,*

$$\|M(c)\| \le \max_{k\le K} \|\Delta W^{(k)}\| \qquad \text{and} \qquad \|M(c)_s\|_F \le \max_{k\le K} \|\Delta W_s^{(k)}\|_F. \tag{20}$$

No such bound holds in the twin. For any $t > 0$ and any k, the feature vector $\xi(h) = t\,\xi_k$ yields the output $t\,\Delta W^{(k)}$.

Takeaway: the mixture never emits an update larger than the largest adapter in the bank, at any conditioning and with no tuning, whereas the scale of a directly generated update is set by the feature geometry.

Remark 4 (Bridge and limits). T2L differs from the twin in three ways. It trains the trunk, it emits the factors A_s, B_s whose product is the update, and it uses an ℓ_1 loss on the updates. The twin idealizes the heads, which hold most of T2L's parameters (Appendix B), and it starts from near zero as T2L does (Appendix B). It predicts two diagnostics on the generated held-out adapter. The fraction of $\|\Delta W_{\text{T2L}}(h_u)\|^2$ that lies in $\operatorname{span}\{\Delta W^{(k)}\}$ should be close to one. The scale ratio $\|\Delta W_{\text{T2L}}(h_u)\| / \max_k \|\Delta W^{(k)}\|$ can fall on either side of one with no mechanism in the objective to correct it, whereas Corollary 5 fixes the ratio at or below one for WebMix. This is also why T2L required the initialization of Appendix B and WebMix required none. If the representer step fails for the factored architecture, the span prediction is the one to test first.

D.3 Well-posedness of the mixture

Proposition 6 (Gauge invariance and coefficient identifiability)**.** *(a) The objective $\mathcal{L}_{\text{router}}$ of Eq. 8 depends on $F_\phi(h_v)$ only through the induced updates $\{B_s A_s\}_{s\in\Omega}$. In particular, replacing bank*

member k by $(Q_k A_s^{(k)},\ B_s^{(k)} Q_k^{-1})$ for any invertible Q_k leaves $M(c)$ and $\mathcal{L}_{\text{router}}$ unchanged for every c. (b) If $\Delta W^{(1)}, \ldots, \Delta W^{(K)}$ are linearly independent in $\mathcal{W}$, the map $c \mapsto M(c)$ is injective on Δ^{K-1}. (c) A loss on factors, such as $\|A - A^*\|_1 + \|B - B^*\|_1$ against a stored oracle (A^*, B^*) with $A^* \neq 0$, takes different values against the targets (QA^*, B^*Q^{-1}) for different Q, although all of them are the same model.

Part (c) is a general remark about objectives on factors. Our T2L RECONS baseline does not have this problem, because it regresses the product $B_s A_s$ (Appendix B.2), which is gauge-invariant by Eq. 9.

Takeaway: the training objective sees the model and not its coordinates, and the coefficients WEBMIX *emits are a well-defined function of the adapter it serves. A factor-space objective is not a function of the target model.*

Remark 5 (Reading the coefficients). Part (b) is a generic condition, checkable from the stored bank as nonsingularity of the $K \times K$ Gram matrix of vectorized updates. Under it, two generated adapters differ if and only if their coefficient vectors differ, so distances between WEBMIX outputs reduce to distances between coefficient vectors. This is the sense in which comparisons among generated adapters are meaningful while comparisons among independently trained adapters are not (§A.1), and it is what licenses reading c as the model's judgment of similarity between interfaces.

E PROOFS

E.1 PROOF OF THEOREM 1

Proof. Existence. Δ^{K-1} is compact, M is linear and hence continuous, and $\mathcal{R}_u$ is continuous on $\mathcal{W}$, so $c \mapsto \mathcal{R}_u(M(c))$ is continuous on a compact set and attains its minimum at some $c^\star$.

Decomposition. Eq. 15 follows from adding and subtracting $\mathcal{R}_u(M(c^\star))$. The routing term is non-negative because $\hat{c} \in \Delta^{K-1}$ and $c^\star$ minimizes over Δ^{K-1}.

First bound. $\mathcal{H}_K = M(\Delta^{K-1})$ is the image of a compact convex set under a linear map, so it is compact and convex. The function $W' \mapsto \|\Delta W^{(u)} - W'\|$ is continuous, so it attains its minimum on $\mathcal{H}_K$ at some point Π, and by definition $\|\Delta W^{(u)} - \Pi\| = \operatorname{dist}(\Delta W^{(u)}, \mathcal{H}_K)$. Since $\Pi \in \mathcal{H}_K$ there is $c_\Pi \in \Delta^{K-1}$ with $\Pi = M(c_\Pi)$. Both Π and $\Delta W^{(u)}$ lie in $\mathcal{K}_u$, so Assumption 1 applies and

$$\begin{aligned}\mathcal{R}_u(M(c^\star)) \;&\leq\; \mathcal{R}_u(M(c_\Pi)) \;=\; \mathcal{R}_u(\Pi)\\ &\leq\; \mathcal{R}_u(\Delta W^{(u)}) + \Lambda_u \|\Pi - \Delta W^{(u)}\|\\ &=\; \mathcal{R}_u(\Delta W^{(u)}) + \Lambda_u \operatorname{dist}(\Delta W^{(u)}, \mathcal{H}_K).\end{aligned}$$

Second bound. For each k, the vertex e_k lies in Δ^{K-1} and $M(e_k) = \Delta W^{(k)}$, so $\mathcal{R}_u(M(c^\star)) \leq \mathcal{R}_u(M(e_k)) = \mathcal{R}_u(\Delta W^{(k)})$. Taking the minimum over k gives Eq. 16. □

E.2 PROOF OF COROLLARY 2

Proof. For $c \in \Delta^{K-1}$, the vector $(c, 0)$ lies in Δ^K and $M_{K+1}((c,0)) = \sum_{k \leq K} c_k \Delta W^{(k)} + 0 \cdot \Delta W^{(K+1)} = M_K(c)$. Hence $\mathcal{H}_K = M_K(\Delta^{K-1}) \subseteq M_{K+1}(\Delta^K) = \mathcal{H}_{K+1}$. A minimum over a superset is no larger than the minimum over the subset. Applied to $W' \mapsto \|\Delta W - W'\|$ this gives the distance inequality, and applied to $\mathcal{R}_v \circ M_{K+1}$ over $\{(c,0) : c \in \Delta^{K-1}\} \subseteq \Delta^K$ it gives $\mathcal{R}_v^\star(K+1) \leq \mathcal{R}_v^\star(K)$. Both minima exist by the continuity and compactness argument in the proof of Theorem 1, which uses only continuity of $\mathcal{R}_v$. □

E.3 PROOF OF COROLLARY 3

Proof. $M(c) = \sum_k c_k \Delta W^{(k)}$ is a convex combination of points of $\mathcal{K}_u$, and $\mathcal{R}_u$ is convex on $\mathcal{K}_u$, so Jensen's inequality gives $\mathcal{R}_u(M(c)) \leq \sum_k c_k \mathcal{R}_u(\Delta W^{(k)})$. The second statement is the case $c = \mathbf{1}/K$.

For the scope stated in Remark 3: if every adapted matrix feeds the output logits affinely, the logits for a fixed input are $\ell(\Delta W) = \ell_0 + \mathcal{A}(\Delta W)$ for a fixed vector ℓ_0 and a fixed linear map $\mathcal{A}$. The

per-token negative log-likelihood $\ell \mapsto \log \sum_j \exp(\ell_j) - \ell_y$ is convex in ℓ, the composition of a convex function with an affine map is convex, and a finite average of convex functions is convex. Hence $\mathcal{R}_u$ is convex on all of $\mathcal{W}$ in that configuration. □

E.4 Proof of Proposition 4

Proof. Write $L(\Gamma) := \|\Gamma\Xi - \mathcal{B}\|_F^2$ for the training loss. Since $\operatorname{rank}\Xi = K \le d_\xi$, the matrix $\Xi^\top\Xi \in \mathbb{R}^{K\times K}$ is invertible and $\Xi^+ := (\Xi^\top\Xi)^{-1}\Xi^\top$ satisfies $\Xi^+\Xi = I_K$. Hence $\hat\Gamma := \mathcal{B}\Xi^+$ obeys $\hat\Gamma\Xi = \mathcal{B}$ and $L(\hat\Gamma) = 0$, so $\hat\Gamma$ is a minimizer and the minimizer set is the affine space $\{\Gamma : \Gamma\Xi = \mathcal{B}\} = \hat\Gamma + \{N : N\Xi = 0\}$.

Minimum norm. Every row of $\hat\Gamma = \mathcal{B}\Xi^+$ is a linear combination of the rows of $\Xi^\top$, that is, of the vectors $\xi_1^\top, \dots, \xi_K^\top$. Every row of an N with $N\Xi = 0$ is orthogonal to each ξ_k. Therefore $\langle \hat\Gamma, N\rangle_F = 0$ and $\|\hat\Gamma + N\|_F^2 = \|\hat\Gamma\|_F^2 + \|N\|_F^2 \ge \|\hat\Gamma\|_F^2$, with equality only at $N = 0$. So $\hat\Gamma$ is the unique minimum-norm minimizer.

Gradient descent. $\nabla L(\Gamma) = 2(\Gamma\Xi - \mathcal{B})\Xi^\top$, whose rows lie in $\operatorname{span}\{\xi_1, \dots, \xi_K\}$ for every Γ. Starting from $\Gamma_0 = 0$, every iterate therefore has rows in that span. L is a convex quadratic whose minimizer set is non-empty, so for any step size below $1/\lambda_{\max}(\Xi\Xi^\top)$ gradient descent converges to a minimizer. A minimizer whose rows lie in $\operatorname{span}\{\xi_k\}$ differs from $\hat\Gamma$ by an N with $N\Xi = 0$ and rows in that span, which forces $N = 0$. Hence the limit is $\hat\Gamma$.

Output. For any h, $\operatorname{vec}\Delta W_{\hat\Gamma}(h) = \hat\Gamma\xi(h) = \mathcal{B}\,\Xi^+\xi(h) = \mathcal{B}\,\eta(h)$, which is Eq. 19. Statements (a) and (b) read off this expression. For (c), if ξ_1, ξ_2 are orthonormal then $\Xi^\top\Xi = I_2$ and $\eta(h) = \Xi^\top\xi(h) = (\langle\xi_1, \xi(h)\rangle, \langle\xi_2, \xi(h)\rangle)$, which equals $(1, 1)$ for $\xi(h) = \xi_1 + \xi_2$ and $(1, -1)$ for $\xi(h) = \xi_1 - \xi_2$. Neither lies in Δ^1. □

E.5 Proof of Corollary 5

Proof. For $c \in \Delta^{K-1}$, the triangle inequality and $c_k \ge 0$ give $\|M(c)\| \le \sum_k c_k\|\Delta W^{(k)}\| \le \big(\sum_k c_k\big)\max_k \|\Delta W^{(k)}\| = \max_k \|\Delta W^{(k)}\|$. The per-matrix statement is the same argument applied to $M(c)_s = \sum_k c_k \Delta W_s^{(k)}$ in the Frobenius norm. For the twin, $\xi(h) = t\,\xi_k$ gives $\eta(h) = \Xi^+(t\,\xi_k) = t\,\Xi^+\Xi e_k = t\,e_k$ and hence $\operatorname{vec}\Delta W_{\hat\Gamma}(h) = t\operatorname{vec}\Delta W^{(k)}$, whose norm is $t\|\Delta W^{(k)}\|$ and is unbounded in t. □

E.6 Proof of Proposition 6

Proof. (a) In Eq. 8, $p_{\theta, F_\phi(h_v)}$ is the likelihood of the network whose weight at matrix s is $W_s + \frac{\alpha}{r} B_s A_s$. Replacing (A_s, B_s) by $(Q_s A_s, B_s Q_s^{-1})$ leaves this weight, hence the likelihood and the loss, unchanged. For WebMix, re-gauging bank member k changes the concatenated factors of Eq. 5 but not the product $\sum_k c_k B_s^{(k)} Q_k^{-1} Q_k A_s^{(k)} = \sum_k c_k B_s^{(k)} A_s^{(k)}$ of Eq. 4, so $M(c)$ is unchanged for every c and so is $\mathcal{L}_{\text{router}}$.

(b) If $M(c) = M(c')$ then $\sum_k (c_k - c'_k)\Delta W^{(k)} = 0$, and linear independence forces $c_k = c'_k$ for every k.

(c) Fix a generator output (A, B) and take $Q = tI_r$ with $t > 0$, so the re-gauged target is $(tA^*, B^*/t)$. The loss $\|A - tA^*\|_1 + \|B - B^*/t\|_1$ is finite at $t = 1$ and tends to infinity as $t \to \infty$ because $A^* \neq 0$. It is therefore not constant in t, while every $(tA^*, B^*/t)$ induces the same update B^*A^*. □

Declaration of LLM Usage

The authors acknowledge the use of LLMs to polish the draft, to brainstorm ideas and research directions, to orchestrate experiments, to verify every proof in Lean, and to review code. We take full responsibility for the paper's content and correctness.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure our results are reproducible. The complete source code is included in the supplementary material and will be released publicly upon acceptance. It comes with a README describing environment setup, dependencies, and each component. All proofs are formalized in Lean 4, and the formalization is included in the supplementary material, so every theorem can be machine-checked. All datasets are publicly available and have been cited.