DUALTRACK: ACCESSIBLE KNOWLEDGE TRANSFER FOR CONTINUAL LEARNING UNDER LONG-TAILED DISTRIBUTIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-tailed class-incremental learning requires a model to acquire new classes from imbalanced task streams while retaining all previously observed classes without replaying historical images. The difficulty is not limited to catastrophic forgetting: few-shot classes provide unreliable estimates of their representation distributions, and the addition of new experts can make historical knowledge inaccessible even when its parameters remain unchanged. We present DUALTRACK, a replay-free framework that couples stable coordinates, statistical routing, protected adaptation, and tail-class transfer. A frozen ViT-B/16 backbone is augmented with Early Shared LoRA modules and late General/Specific LoRA branches. An independent routing stream maintains class-conditional diagonal Gaussian statistics and decides whether a new class should reuse an existing expert or receive additional capacity. When an existing expert absorbs a class, rank expansion preserves its function at initialization, freezes the old rank prefix, and trains only the newly allocated directions; historical gradient-direction projection further restricts eligible updates to directions compatible with prior adaptation. Finally, image-free class prototypes are transferred to tail classes using visual similarity, cross-layer route overlap, and a count-adaptive shrinkage coefficient. We evaluate on long-tailed class-incremental benchmarks and compare DUALTRACK with replay, regularization, prompt, expandable-expert, and prototype-based baselines under matched ViT backbones.

1 INTRODUCTION

Continual learning studies models that learn from a sequence of changing data distributions while retaining previously acquired knowledge. Surveys of the field distinguish replay, regularization, parameter isolation, and architecture-based strategies Parisi et al. (2019); De Lange et al. (2022). In the class-incremental setting, tasks introduce disjoint classes, and evaluation requires prediction over all observed classes without a task identifier. Replay, regularization, and parameter-isolation methods have substantially advanced this setting, but their assumptions are often mismatched with realistic streams. In particular, the number of observations per class can vary by orders of magnitude within the same task. A model must then learn a new class from limited evidence while preserving the decision boundaries and representations established by earlier tasks.

We study this setting as long-tailed class-incremental learning (LT-CIL). Long-tailed imbalance changes the nature of forgetting. A head class receives many updates and typically forms a relatively stable feature estimate, whereas a tail class is represented by a small and noisy sample set. Subsequent task updates can therefore damage tail classes before their representations have become reliable. In addition, class imbalance and task incrementality interact through the classifier: the frequent classes dominate gradient updates, while new classes alter the competition among all previously learned classes. A method that only reweights gradients, balances a replay buffer, or calibrates the final classifier addresses one part of this process but does not ensure that the correct historical representation is retrieved.

Pre-trained vision transformers provide a useful starting point because their representations can support parameter-efficient continual adaptation. Prompt and adapter methods reduce the number of updated parameters, while expandable mixture-of-experts methods allocate additional modules

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.

when a new distribution cannot be handled by existing capacity. However, expansion introduces a second failure mode: the model can preserve an old expert and still select a different expert for an old class after the expert pool grows. This distinction between *parameter preservation* and *knowledge accessibility* is particularly important for tail classes, whose weak evidence makes route selection more uncertain. Conversely, unrestricted sharing improves transfer but exposes historical paths to representation drift.

Our central hypothesis is that stability and accessibility should be coupled through a coordinate system that is shared by routing and tail transfer. We therefore design DUALTRACK around four dependent decisions. First, a frozen ViT-B/16 and Early Shared LoRA modules provide a common, economical coordinate system while retaining limited adaptation capacity. Second, late General and Specific LoRA branches maintain a main representation stream and an independent routing stream. The routing stream excludes preceding Specific residuals, so a temporary expert choice does not recursively redefine the coordinates used to make later choices. Third, a class-conditional hierarchical Gaussian router uses these coordinates to determine expert reuse, isolation, and capacity expansion. Rank expansion preserves the existing expert function at initialization, while historical gradient projection and old-rank protection constrain subsequent updates. Finally, image-free class prototypes transfer information to low-count classes only when visual compatibility and cross-layer route overlap support the transfer, with stronger shrinkage for classes with fewer observations.

This design closes a specific loop. The shared coordinates make class statistics comparable across tasks; the statistics determine which expert path is accessible to a new class; the resulting path assignments define which historical classes are plausible sources for a tail prototype; and the prototype posterior provides label-free evidence for route feedback at inference. No component stores historical image pixels. The method therefore uses the same persistent state for both preservation and transfer, while keeping the two roles mathematically distinct.

We evaluate DUALTRACK against representative replay-based, gradient-based, prompt-based, prototype-based, and expandable-adapter baselines. The benchmark matrix contains CUB-200, CIFAR-10, CIFAR-100, TinyImageNet, CORe50, and ImageNet-A under their prescribed task protocols and random seeds. We report final and average incremental accuracy, forgetting, backward transfer when defined, trainable parameters, peak GPU memory, training time, and batch-size-specific inference latency.

Our contributions are fourfold. We formulate replay-free LT-CIL as the coupled problem of representation stability, expert accessibility, and tail-estimation uncertainty. We introduce a dual-track ViT adapter whose neutral/general stream supplies class-conditional Gaussian routing coordinates, while the main stream applies selected Specific LoRA residuals. We combine absorb-or-isolate routing with function-preserving rank expansion, old-rank protection, and historical gradient-direction projection. Finally, we propose image-free, route-aware prototype transfer for low-count classes and evaluate accuracy, forgetting, routing, and resource cost.

2 RELATED WORK

Continual learning and parameter-efficient adaptation. Continual learning methods mainly rely on replay, regularization, parameter isolation, or their combinations. Replay methods retain representative examples or features from previous tasks Rolnick et al. (2019); Rebuffi et al. (2017); Yan et al. (2021), whereas regularization-based methods constrain changes to previously learned parameters or representations Kirkpatrick et al. (2017); Li & Hoiem (2018); Chaudhry et al. (2019); Cha et al. (2021). Parameter-efficient adaptation is particularly relevant when a pretrained vision transformer is used: vision-language pretraining provides transferable representations Radford et al. (2021), prompt-based methods learn reusable prompt pools Wang et al. (2022b;a); Bonicelli et al. (2024), and RanPAC reduces continual updates through frozen projections and prototype accumulation McDonnell et al. (2023). These methods motivate our use of frozen pretrained representations and image-free statistics, but they do not jointly determine whether a new class should reuse an existing expert or receive additional expert capacity.

Long-tailed continual learning. Long-tailed recognition addresses class-frequency imbalance through sampling, reweighting, calibrated prediction, or frequency-aware optimization Menon et al. (2021); Park et al. (2021; 2022); Kim et al. (2020); Belouadah et al. (2020); He (2024). DCE further studies frequency-aware collaborative experts and dynamic selection in imbalanced domain-incremental learning Li et al. (2025). These works show that imbalance affects both prediction

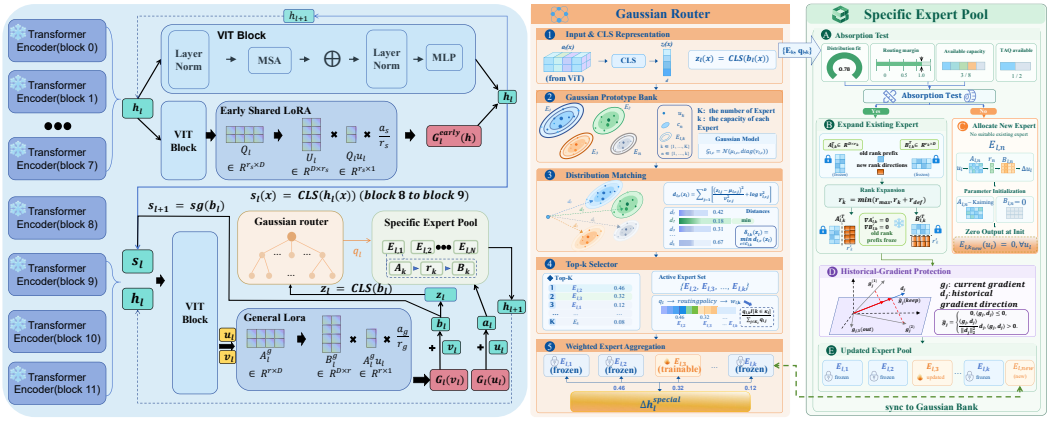


Figure 1: Distribution-guided expert reuse and protected adaptation in DualTrack. Left: shared LoRA adapts the frozen backbone, while an independent routing stream excludes Specific residuals from routing coordinates. Center: class-conditional Gaussian statistics in these coordinates support expert selection and provide distributional evidence for reuse. Right: the Specific Expert Pool reuses or expands an existing expert, or allocates a new one, based on the routing result, followed by protected updates.

and continual optimization. Our method uses tail frequency to control image-free prototype transfer and combines the transfer source with route overlap, rather than treating imbalance only through sampling or loss weighting.

Expandable experts and method position. Expandable architectures reduce direct interference by allocating new capacity. DER and recent modular adaptation methods dynamically expand or select expert-like components Yan et al. (2021); Yu et al. (2024); Wang et al. (2025); Ye et al. (2025). KeepLoRA further motivates protecting update directions associated with pretrained and previously learned representations Luo et al. (2026). The remaining gap is the interaction of these ideas under image-replay-free long-tailed class-incremental learning: a tail class may have an unreliable statistic, a reused expert may lack sufficient output directions, and a protected historical expert may remain inaccessible. DUALTRACK addresses these issues with a shared state design that combines Gaussian route assignment, class-conditional expert reuse or expansion, direction-protected updates, and count-adaptive prototype transfer. The extended discussion and method-level comparisons are provided in Appendix A.

3 METHOD

3.1 PROBLEM AND DESIGN RATIONALE

We learn from a sequence of class-disjoint tasks $\mathcal{D}^t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t}$, where class frequencies are imbalanced and $\mathcal{C}^t$ denotes all classes observed through task t. Only the current training images are accessible; historical information is retained as model parameters and image-free statistics. At inference, the model predicts over $\mathcal{C}^t$ without a task label.

Reusing historical experts requires both selecting an appropriate expert and preserving its usefulness as new classes arrive. These requirements interact: an expert update can change the features used for subsequent route selection, while a correct reuse decision can still overload a limited adaptation subspace. Class imbalance adds another difficulty, because a small class region can be obscured when several classes are represented by one pooled expert statistic.

DualTrack addresses these issues through distribution-guided expert reuse with protected adaptation. As shown in Fig. 1, an independent routing stream first removes Specific residuals from the coordinates on which route decisions depend. Class-conditional Gaussian statistics in those coordinates then support both sample-level expert selection and class-level reuse decisions. Accepted reuse can increase an expert's rank without changing its response at initialization; old-prefix masking and eligible historical-gradient protection constrain different forms of subsequent update. These mechanisms connect through the routing coordinates, class-to-expert assignments, and expert state, so capacity allocation is tied to the same class structure used for selection. Training supervision and prototype readout use the registered assignments to address remaining route ambiguity and limited tail-class evidence.

3.2 Separating Routing from Specific Adaptation

Our reuse criterion compares current features with stored class statistics. If Specific adaptation also changes the features used for routing, a class can move relative to its registered region even when the stored expert remains useful. We therefore retain shared feature adaptation while removing Specific residuals from the routing recurrence. This supplies the coordinate system for the Gaussian matching below.

We use a frozen CLIP-pretrained ViT-B/16 with hidden dimension $D = 768$. Early Shared LoRA is applied to blocks 0–7, block 8 remains a frozen bridge, and DualTrack modules follow blocks 9–11. Let F_l^0 denote a frozen block. Its early shared residual is

$$G_l^{\text{early}}(u) = \frac{\alpha_s}{r_s} U_l Q_l u, \tag{1}$$

where $Q_l \in \mathbb{R}^{r_s \times D}$ is fixed with $Q_l Q_l^{\mathsf{T}} = I_{r_s}$, and $U_l \in \mathbb{R}^{D \times r_s}$ is trainable and initialized to zero. Here, I_{r_s} denotes the $r_s \times r_s$ identity matrix. The residual is added to the frozen block output. This shared stage allows adaptation before expert selection, so shared adjustments need not be duplicated across Specific experts. Fixing the down-projection limits this adaptation to the up-projection; zero initialization preserves the initial frozen representation.

At a late layer, h_l and s_l denote the classification and routing token sequences. The classification branch first computes the frozen block output and then adds the General residual:

$$u_l = F_l^0(h_l), \tag{2}$$

$$a_l = u_l + G_l(u_l). \tag{3}$$

The routing branch applies the same frozen block and General module to its own input, without introducing a Specific residual:

$$v_l = F_l^0(s_l), \tag{4}$$

Prototype fusion

$$b_l = v_l + G_l(v_l). \tag{5}$$

The router reads the CLS token, while the complete token sequence is passed to the next routing layer with stopped gradients:

$$z_l = \text{CLS}(b_l), \tag{6}$$

$$s_{l+1} = \text{sg}(b_l). \tag{7}$$

Here $s_9 = h_9$ initializes both streams with the same token sequence. General LoRA acts tokenwise as

$$G_l(u) = \frac{\alpha_g}{r_g} B_l^g A_l^g u, \tag{8}$$

where $A_l^g \in \mathbb{R}^{r_g \times D}$ and $B_l^g \in \mathbb{R}^{D \times r_g}$ are shared between the two streams at layer l. This parameter sharing provides a common adaptation component without forcing the streams to share their evolving inputs. The classification stream additionally receives Specific residuals:

$$h_{l+1} = a_l + \xi \sum\nolimits_{k=1}^{K_l} w_{l,k} E_{l,k}(u_l), \tag{9}$$

where K_l is the expert count, $w_{l,k}$ is a routing weight, and ξ scales the Specific contribution. Each expert operates on u_l, whereas the router reads only z_l. In a single stream, an earlier Specific residual becomes part of the next block's input and can shift subsequent route decisions. Here, the routing recurrence excludes that residual, so changing Specific parameters alone leaves the route coordinates unchanged when shared parameters are fixed. Shared adapters may still evolve. The resulting separation removes one source of inconsistency between current coordinates and stored statistics; the next component defines how those statistics represent classes and select experts.

3.3 Class Structure for Selection and Reuse

Separating the streams does not determine which classes should share an expert. An expert may serve several class regions, and a pooled center can be dominated by its more frequent classes. Such a center can obscure a small class region during both routing and reuse assessment. We therefore preserve class-level distributions inside each expert and evaluate an expert through its best-matching registered class.

Each layer stores a diagonal Gaussian for every observed class: $\mathcal{G}_{l,c} = \mathcal{N}(\mu_{l,c}, \mathrm{diag}(\nu_{l,c}))$, together with a count and a registered expert $m_l(c)$. The variance $\nu_{l,c}$ captures the dispersion of routing coordinates within class c. With $\bar{\nu}_{l,c,j} = \max(\nu_{l,c,j}, \epsilon)$, the class energy is

$$d_{l,c}(z) = \sum_{j=1}^{D} \left[\frac{(z_j - \mu_{l,c,j})^2}{\bar{\nu}_{l,c,j}} + \log \bar{\nu}_{l,c,j} \right]. \tag{10}$$

The squared-distance term measures mismatch relative to class dispersion, while the log-variance term accounts for the scale of that dispersion. A positive variance floor keeps the score finite even for classes with very little observed variation.

For the class set $\mathcal{C}_{l,k} = \{c : m_l(c) = k\}$, the expert energy is the best match among its registered classes:

$$\delta_{l,k}(z) = \min_{c \in \mathcal{C}_{l,k}} d_{l,c}(z). \tag{11}$$

We convert these energies into normalized routing probabilities:

$$q_{l,k}(z) = \frac{\exp[-\delta_{l,k}(z)/\tau_r]}{\sum_{j=1}^{K_l} \exp[-\delta_{l,j}(z)/\tau_r]}, \tag{12}$$

where τ_r is the routing temperature. The minimum in Eq. equation 11 lets a sample match a small registered class region even when that expert also contains larger classes. It does not make low-count distribution estimates exact; it preserves their separate contribution to expert selection. Expert-level Gaussian statistics provide a fallback when an expert has no registered class.

For the Top-K policy illustrated in the router, let $\mathcal{K}_l$ be the selected expert set. The aggregation weights are

$$w_{l,k} = \frac{q_{l,k} \mathbf{1}[k \in \mathcal{K}_l]}{\sum_{j \in \mathcal{K}_l} q_{l,j}}. \tag{13}$$

Gaussian statistics are stored as buffers, and the routing inputs and outputs are detached. The same class energies also supply the micro-coverage evidence used by the absorption test below. Thus the Gaussian Router and expert pool share a distributional description: one uses it to select experts for samples, while the other uses it to decide where new classes belong. The updated map $m_l(c)$ in turn determines the class sets used in future expert scores. Training-time path supervision and prediction-based route refinement are specified in Appendix B.6.

3.4 Protected Expert Reuse

Constrained absorption. The router identifies distributionally compatible experts, but compatibility alone does not establish how much new information an expert should receive. Repeatedly assigning compatible classes to the same expert concentrates adaptation in a limited parameter subspace. We therefore use its class-region coverage as evidence for reuse and apply explicit limits to the resulting assignments.

Before training each task, we scan its current images without gradients to collect class-level routing features. For each new class, the absorption test selects the historical expert with the highest micro-coverage, breaking ties by expert-level distance. Absorption requires sufficient coverage and separation margin, available class capacity, and an unused task assimilation quota (TAQ). TAQ limits how many new classes one expert can absorb within a task. Classes that fail the test are assigned to new experts subject to a separate allocation quota; when that quota is exhausted, a historical fallback preserves a valid assignment without automatically expanding its rank. Distribution fit and margin assess whether reuse is plausible; capacity and TAQ bound the number of classes assigned to that expert. These assignment limits do not increase its representational capacity. An accepted absorption therefore also supplies the structural event that can trigger the rank expansion described next. Appendix B.2 gives the complete rule.

Function-preserving rank expansion. Absorbing a class reuses an existing path, but its adaptation can require directions unavailable in the current low-rank factors. Keeping the rank fixed would require fitting the new residual with those same factors. Rank expansion is motivated by reducing the representational residual induced by the limited output subspace. We therefore retain the existing contribution and add trainable rank slices, using absorption as the expansion trigger rather than

assuming that distributional compatibility implies sufficient capacity. Each Specific expert has the form

$$E_{l,k}(u) = \frac{\alpha}{r_{l,k}} B_{l,k} A_{l,k} u, \tag{14}$$

where $A_{l,k} \in \mathbb{R}^{r_{l,k} \times D}$ and $B_{l,k} \in \mathbb{R}^{D \times r_{l,k}}$. A new expert uses Kaiming-initialized A and zero B, so its initial residual is zero. Ordinary absorption can expand an existing expert once per task. Suppressing layer and expert indices, the target rank is

$$r' = \min(r_{\max}, r + r_{\text{def}}). \tag{15}$$

When $r' > r$, the expanded factors are initialized as

$$A' = \begin{bmatrix} A \\ A^+ \end{bmatrix}, \tag{16}$$

$$B' = \begin{bmatrix} (r'/r)B & B^+ \end{bmatrix}. \tag{17}$$

The new A^+ rows use Kaiming initialization and $B^+ = 0$. Rescaling the old B columns compensates for the changed LoRA scale, giving $(\alpha/r')B'A' = (\alpha/r)BA$ at expansion. Old rank-prefix gradients are masked while the added slices receive gradients. The optimizer is rebuilt after the parameter tensors change. Thus capacity can grow without an immediate jump in the expert response. This connects reuse to capacity growth: the registered expert keeps its response at the expansion boundary, while the new factors provide directions for adapting to the absorbed classes. It also complements the dual-stream design. New Specific residuals can change classification features without entering the independent routing recurrence. Rank saturation retains the existing capacity and does not itself allocate another expert.

Historical-gradient protection. Function continuity controls the expansion instant, and prefix masking controls gradients on the retained slices. Neither specifies how to update a historical expert when that expert remains eligible for learning without being treated as newly expanded capacity. For this case, we retain a gradient-direction memory and use it to constrain subsequent historical updates.

For an eligible historical Specific parameter tensor, let g_j be its current gradient and d_j its stored normalized gradient direction. We replace the gradient by

$$\widehat{g}_j = \begin{cases} 0, & \langle g_j, d_j \rangle \le 0, \\ \dfrac{\langle g_j, d_j \rangle}{\|d_j\|_2^2} d_j, & \langle g_j, d_j \rangle > 0. \end{cases} \tag{18}$$

Inner products operate on vectorized tensors. The rule retains a positive multiple of the stored direction, removing orthogonal components and setting nonpositively aligned gradients to zero. New and just-expanded experts marked as current-task capacity bypass this projection. The two protections therefore address complementary update cases: prefix masking separates old and added slices after expansion, whereas direction memory constrains eligible historical tensors that continue to learn. Neither substitutes for the routing separation: protecting an expert's parameters alone does not ensure that later samples are routed to it. Direction accumulation and update conditions are detailed in Appendix B.3.

3.5 Using Registered Structure in Training and Prediction

The preceding components establish a registered mapping from classes to experts, but sample-level Gaussian matching can remain uncertain. During training, the current label provides the registered expert as a supervision signal. For samples that satisfy the confidence gate, a teacher distribution mixes the autonomous route with the registered path, transferring the class-level assignment to sample-level expert training. This mechanism is used only during training; inference requires neither task identities nor ground-truth labels. Correct routing and additional capacity do not eliminate the limited statistical evidence available for estimating the final representations of tail classes. At task boundaries, we retain final feature prototypes together with the routing statistics and class-to-expert assignments. For a low-count class, similar historical prototypes provide a bounded prior, weighted by visual similarity, path overlap under $m_l(c)$, confidence, and the class sample count. Prototype statistics remain separate from the routing Gaussians, so prototype refinement does not overwrite Gaussian moments or directly alter training routes. At inference, first-pass class posteriors and the

Table 1: Results on six long-tailed Class-IL benchmarks ($\rho = 10$). All values are final average Class-IL accuracy (%), reported as the mean and sample standard deviation over three seeds. The best result in each column is shown in **bold**, and the second-best result is underlined.

Method	CIFAR-10	CIFAR-100	Core50	CUB-200	TinyImageNet	ImageNet-A
CL-LoRA He et al. (2025)	$92.91_{\pm 1.02}$	$78.18_{\pm 0.78}$	$78.54_{\pm 1.26}$	$63.06_{\pm 0.29}$	$85.54_{\pm 0.53}$	$61.76_{\pm 0.66}$
DCE Li et al. (2025)	$94.46_{\pm 0.44}$	$77.81_{\pm 1.82}$	$79.19_{\pm 1.00}$	$45.42_{\pm 1.17}$	$81.99_{\pm 0.37}$	$34.61_{\pm 0.89}$
DER++ Yan et al. (2021)	$81.27_{\pm 6.71}$	$42.68_{\pm 7.33}$	$44.12_{\pm 3.40}$	$44.07_{\pm 2.51}$	$18.60_{\pm 0.72}$	$28.24_{\pm 4.42}$
DSEF Ye et al. (2025)	$69.40_{\pm 10.11}$	$37.94_{\pm 4.33}$	$31.01_{\pm 8.37}$	$21.09_{\pm 3.48}$	$18.41_{\pm 1.21}$	$7.60_{\pm 1.33}$
KeepLoRA Luo et al. (2026)	$92.74_{\pm 1.35}$	$79.67_{\pm 0.70}$	$\underline{82.17_{\pm 0.54}}$	$60.80_{\pm 1.20}$	$57.70_{\pm 0.39}$	$\mathbf{67.51}_{\pm 2.55}$
MoE-Adapters Yu et al. (2024)	$\underline{95.88_{\pm 0.93}}$	$83.34_{\pm 4.18}$	$37.62_{\pm 0.58}$	$61.80_{\pm 0.49}$	$27.91_{\pm 1.18}$	$45.30_{\pm 1.30}$
RanPAC McDonnell et al. (2023)	$94.73_{\pm 0.16}$	$80.82_{\pm 0.17}$	$72.44_{\pm 0.53}$	$73.23_{\pm 1.02}$	$83.39_{\pm 0.08}$	$42.35_{\pm 0.81}$
StarPrompt Bonicelli et al. (2024)	$92.07_{\pm 0.64}$	$\underline{83.39_{\pm 0.33}}$	$76.03_{\pm 1.90}$	$\underline{73.97_{\pm 0.64}}$	$\underline{86.09_{\pm 0.62}}$	$47.98_{\pm 0.26}$
DualTrack (Ours)	$\mathbf{95.90}_{\pm 0.32}$	$\mathbf{84.01}_{\pm 0.54}$	$\mathbf{83.61}_{\pm 1.04}$	$\mathbf{81.74}_{\pm 0.35}$	$\mathbf{87.27}_{\pm 1.22}$	$\underline{67.28_{\pm 0.87}}$

same class-to-expert mapping refine expert selection, while the final prediction combines the log probabilities from the linear head and the prototypes. In the neutral prototype-only mode, Specific outputs are disabled, so route refinement does not directly change the final features. Feature modes, prototype transfer, classifier protection, and routing policies are specified in Appendix B.4–B.6.

4 EXPERIMENTS

Datasets and protocol. We evaluate DUALTRACK on six long-tailed class-incremental learning benchmarks: CIFAR-10, CIFAR-100, Core50, CUB-200, TinyImageNet, and ImageNet-A. CIFAR-10 and Core50 contain five and eight tasks, respectively, while the other benchmarks contain ten tasks. All streams use a fixed long-tail ratio of $\rho = 10$ with the prescribed class orders and task partitions. For ImageNet-A, we use a fixed stratified split over 200 classes.

We follow the strict Class-IL protocol. After task t, a single model predicts over the union of all classes observed up to task t, denoted by $\mathcal{C}_{1:t}$. At inference, neither the task identity, the ground-truth label, nor a task-specific oracle is provided. Therefore, classifier, adapter, and expert selection must be determined from the input and the model's learned state. Task boundaries are used only for incremental updates during training. DUALTRACK is replay-free: images from previous tasks are neither stored nor replayed; only model parameters and image-free historical statistics are retained.

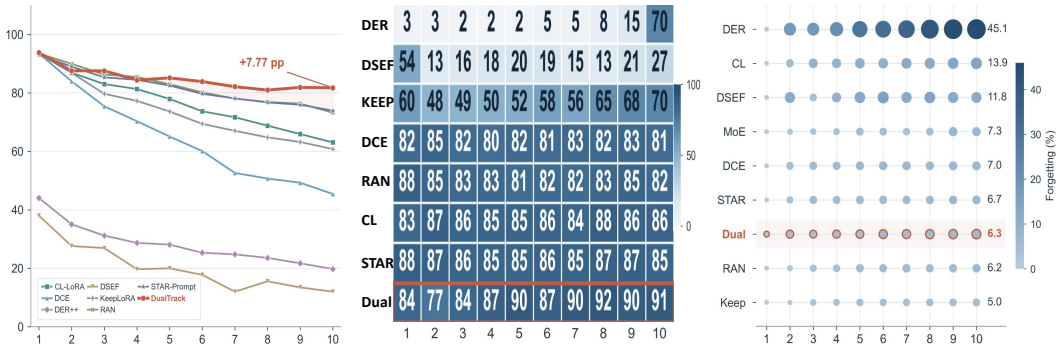
Implementation details. We use a frozen CLIP ViT-B/16 backbone and train the low-rank adapters, classifier, and routing/prototype states. ImageNet-A uses a batch size of 16; all other datasets use a batch and minibatch size of 32, four data-loading workers, and a learning rate of 0.01. CIFAR-10, CUB-200, and TinyImageNet use three epochs per task, CIFAR-100 uses five epochs, and ImageNet-A uses twelve epochs. Route-specific LoRA modules are inserted into the last three Transformer blocks with ranks 2, 4, and 8. At inference, prototype-logit fusion uses temperature 16 and weight 1.0, and routing considers the top two candidate experts. The classification stream uses General and Specific LoRA adaptation, whereas the routing stream uses the frozen backbone with General LoRA; the remaining routing and state-update details are given in the Method section.

4.1 MAIN RESULTS

Overall comparison. DUALTRACK obtains the highest average accuracy on CIFAR-10, CIFAR-100, Core50, CUB-200, and TinyImageNet. On CUB-200, it reaches 81.74%, outperforming the strongest baseline, StarPrompt ($73.97_{\pm 0.64}$%), by 7.77 percentage points. On TinyImageNet, it reaches 87.27%, exceeding StarPrompt ($86.09_{\pm 0.62}$%) by 1.18 points. On Core50, its 83.61% accuracy is 1.00 point higher than KeepLoRA ($82.17_{\pm 0.54}$%). These gains span both 8-task and 10-task sequences, indicating that the dual-stream adaptation and prototype decision mechanism remain effective across different class counts and task lengths.

On CIFAR-10, the $95.90_{\pm 0.32}$% result of DUALTRACK is essentially tied with the $95.88_{\pm 0.93}$% result of MoE-Adapters, with a mean difference of only 0.02 percentage points. On CIFAR-100, DUALTRACK reaches $84.01_{\pm 0.54}$%, outperforming StarPrompt ($83.39_{\pm 0.33}$%) and MoE-Adapters ($83.34_{\pm 4.18}$%) by 0.62 and 0.67 points, respectively. The larger standard deviation of MoE-Adapters on CIFAR-100 indicates greater sensitivity to the random seed.

On ImageNet-A, DUALTRACK achieves $67.28_{\pm 0.87}$%, ranking second behind KeepLoRA at $67.51_{\pm 2.55}$% by 0.23 points. It exceeds CL-LoRA ($61.76_{\pm 0.66}$%) by 5.52 points and StarPrompt



(a) CUB-200 **(b)** TinyImageNet **(c)** CIFAR-100

Figure 2: Task-level behavior on three ten-task Class-IL benchmarks. (a) Average accuracy over all observed classes after each CUB-200 task, where higher is better. (b) Per-task accuracy on TinyImageNet after completing task 10, where darker cells indicate higher accuracy. (c) Average forgetting on CIFAR-100 after each task, jointly encoded by bubble size and color, where lower is better. Baseline values are averaged over three seeds.

($47.98_{\pm 0.26}$%) by 19.30 points. Thus, DUALTRACK remains close to the strongest observed result under a long-tailed stream with substantial distribution shift. Overall, DUALTRACK ranks first on five of the six benchmarks and second on ImageNet-A.

Task-level dynamics. Figure 2 provides complementary views of the behavior behind the aggregate results. On CUB-200, the average accuracy of every method decreases as the number of tasks grows, but at different rates: DUALTRACK remains close to the strongest baselines over the first four tasks, leads from task 5 onward, and finishes at 81.74%, 7.77 points above StarPrompt at 73.97%. On TinyImageNet, DUALTRACK attains the highest accuracy among the plotted methods on tasks 4–10 after the full sequence, including 91.72% on task 8 and 90.98% on task 10; its lowest task-level value is 77.25% on task 2, showing that the gain is not uniform across historical tasks. On CIFAR-100, the separation in forgetting becomes more visible as additional tasks arrive. After task 10, DUALTRACK records 6.31% average forgetting, close to RanPAC at 6.23%, lower than StarPrompt at 6.71%, DCE at 7.00%, DSEF at 11.78%, CL-LoRA at 13.86%, and DER++ at 45.10%, but higher than KeepLoRA at 5.02%. Together with Table 1, these patterns indicate that the advantage of DUALTRACK reflects a favorable balance between retaining previous knowledge and learning new classes rather than uniformly minimizing forgetting at every stage.

4.2 TRAINING COST AND TRAINABLE PARAMETERS

Efficiency analysis. Table 2 shows that DUALTRACK completes the five-task CIFAR-10 stream in 35.55 minutes, comparable to RanPAC at 35.68 minutes and MoE-Adapters at 34.28 minutes. DSEF is the fastest method at 15.00 minutes, but its main-table accuracy is only $69.40_{\pm 10.11}$%, substantially below the $95.90_{\pm 0.32}$% of DUALTRACK. Training time must therefore be considered together with final accuracy; speed alone does not capture the overall performance–efficiency trade-off.

Table 2: Training cost on long-tailed CIFAR-10 over five tasks.

Method	Time (min) ↓	Memory (MiB) ↓	Params. (M) ↓
CL-LoRA	120.50	5,614.00	0.276
DCE	42.52	3,394.00	0.012
DER++	755.32	4,012.00	85.806
DSEF	15.00	4,405.33	21.264
KeepLoRA	59.11	6,825.33	0.983
MoE-Adapters	34.28	1,818.44	4.027
RanPAC	35.68	1,128.00	0.100
StarPrompt	350.04	4,750.00	0.166
DualTrack (Ours)	35.55	5,424	0.121

Time is measured over the complete task stream; memory is the peak GPU usage; and parameters are trainable parameters after training.

DUALTRACK is trainable-parameter efficient. It optimizes only 0.121M parameters, approximately 3.0% of the 4.027M trainable parameters used by MoE-Adapters, corresponding to a reduction of about 97.0%. Its trainable parameter count is also below those of CL-LoRA, KeepLoRA, StarPrompt, DSEF, and DER++, and only slightly above DCE at 0.012M and RanPAC at 0.100M. Its peak memory usage is 5,424 MiB, which is higher than that of MoE-Adapters and RanPAC, so parameter efficiency does not directly translate into the lowest runtime memory footprint. Overall, the main efficiency benefit of DUALTRACK is that it combines a small trainable parameter budget with competitive training time and high final accuracy.

4.3 COMPONENT AND STRUCTURAL ABLATIONS

The local component, core structural, and tail-transfer ablations are conducted on ten-task CUB-200. The local component study uses ten tasks, three epochs per task, a batch size of 32, and seed 42,

Table 3: Local component ablation on ten-task CUB-200.

Variant	$\mathcal{A}_{\text{avg}}$	Δ	RM	OM
Full model	81.74	0.00	72.36	25.91
w/o early shared LoRA	81.06	-0.68	71.77	26.99
w/o prototype fusion	61.38	**-20.36**	70.83	27.46

ΔA is the change from the full model in percentage points; RM and OM are the final-task route-match and old-class route-mismatch rates, respectively. All values are percentages.

Table 4: Core structural ablation on ten-task CUB-200.

Structural variant	$\mathcal{A}_{\text{avg}}$	RM	OM
w/o dual stream (single stream)	72.65	51.57	47.16
w/o rank expansion (fixed rank 2)	74.91	70.33	28.03

RM: task-9 route match; OM: old-class route mismatch. All values are percentages. The ten-task protocol is distinct from the four-task component study.

Table 5: Matched controls for tail transfer and rank adaptation. Each accuracy pair is reported as Variant 1 / Variant 2 under the corresponding dataset protocol.

Dataset	Variant 1	Variant 2	Accuracy (%)
CUB-200	Tail transfer off	Tail transfer on	79.68 / 81.74
CIFAR-10	Dynamic rank	Fixed Specific rank 8	96.02 / 96.08

removing one component at a time. The core structural study uses the same ten-task protocol and disables either the dual-stream design or rank expansion. The minimal tail-transfer comparison also follows the ten-task CUB-200 protocol and contains two variants: the full model with tail transfer and the same model without tail transfer.

On CIFAR-10, dynamic rank expansion reaches 96.02%, only 0.06 percentage points below the fixed Specific rank-8 variant at 96.08%. It achieves nearly identical accuracy with a smaller average Specific-rank budget, thereby reducing the number of trainable adapter parameters and the corresponding adapter computation. This result supports dynamic expansion as a parameter- and computation-efficient alternative to allocating rank 8 uniformly. We report final Class-IL accuracy together with routing diagnostics where available.

Ablation analysis. Prototype fusion has the largest effect in the local component study: removing it reduces final Class-IL accuracy from 81.74% to 61.38%, a drop of 20.36 percentage points, while decreasing route matching and increasing old-class route mismatch. Removing Early Shared LoRA reduces accuracy to 81.06%, a drop of 0.68 points, with a smaller but directionally consistent change in the routing diagnostics. These results support retaining both components in the full model.

Relative to the 81.74% full-model result in Table 1, disabling the dual-stream design and rank expansion reduces accuracy by 9.09 and 6.83 points, respectively. Without the dual stream, the task-9 route-match rate falls to 51.57% and the old-class route-mismatch rate rises to 47.16%. Without rank expansion, the corresponding values are 70.33% and 28.03%. These controlled ablations indicate that the independent routing stream mitigates interference from specific LoRA modules in historical route representations, while rank expansion provides additional representational capacity for absorbing new classes; together, they reduce cross-task conflicts over long task sequences.Taken together, the ablations distinguish local component contributions from long-horizon structural effects. Prototype fusion provides the dominant decision-level gain, while the dual-stream route and adaptive rank expansion preserve route stability and representational capacity as the task sequence grows. The two types of ablation therefore support complementary roles for decision refinement and continual representation protection.

The matched controls in Table 5 isolate the two mechanisms. On CUB-200, enabling tail transfer improves final Class-IL accuracy from 79.68% to 81.74% under the ten-task protocol. On CIFAR-10, dynamic rank expansion reaches 96.02%, closely matching the fixed rank-8 result of 96.08%, while using a smaller average Specific-rank budget. These results show that tail transfer provides a measurable benefit on CUB-200, whereas dynamic expansion preserves the fixed-rank performance level with lower rank allocation.

5 Conclusion

In this work, we study replay-free long-tailed class-incremental learning by jointly addressing representation stability, expert accessibility, and tail-class uncertainty. We propose DUALTRACK, a parameter-efficient dual-stream framework that enables distribution-guided expert reuse and protected adaptation through Gaussian routing, function-preserving rank expansion, gradient protection, and image-free prototype transfer.

AI USE STATEMENT

In this work, we used generative AI tools for language polishing and for limited assistance with code drafting, debugging, and organization. We have not used generative AI tools to generate or alter experimental data, fabricate citations, or replace author judgment in scientific decisions and conclusions; the remaining required-disclosure tasks are not applicable to this work. Additionally, we used generative AI tools for code-level troubleshooting and suggestions for improving the clarity of technical writing. All AI-assisted text and code were reviewed by the authors. Code suggestions were manually inspected, adapted when appropriate, and verified through compilation, testing, and experimental runs. The reported results, claims, citations, figures, tables, and conclusions were checked against the original sources and experiment records. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

REPRODUCIBILITY STATEMENT

The appendix specifies the architecture, update rules, hyperparameters, and evaluation protocol. The supplementary material will include anonymized source code, per-seed training scripts, environment information, and raw logs used to produce the reported results.

REFERENCES

Eden Belouadah, Adrian Popescu, Umang Aggarwal, and Leo Saci. Active class incremental learning for imbalanced datasets. In *European Conference on Computer Vision*, pp. 146–162, 2020.

Lorenzo Bonicelli, Matthias De Lange, et al. Semantic residual prompts for continual learning. In *European Conference on Computer Vision*, 2024.

Hyuntak Cha, Jaeho Lee, and Jinwoo Shin. Co2l: Contrastive continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9516–9525, 2021.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *International Conference on Learning Representations*, 2019.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022.

Jiangpeng He. Gradient reweighting: Towards imbalanced class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16668–16677, 2024.

Jiangpeng He, Zhihao Duan, and Fengqing Zhu. Cl-lora: Continual low-rank adaptation for rehearsal-free class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 30534–30544, 2025.

Chris Dongjoo Kim, Jinseo Jeong, and Gunhee Kim. Imbalanced continual learning with partitioning reservoir sampling. In *European Conference on Computer Vision*, pp. 411–428, 2020.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. 114(13):3521–3526, 2017.

Lan Li, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Addressing imbalanced domain-incremental learning through dual-balance collaborative experts. In *Proceedings of the International Conference on Machine Learning*, 2025.

Zhizhong Li and Derek Hoiem. Learning without forgetting. 40(12):2935–2947, 2018.

Mao-Lin Luo, Zi-Hao Zhou, Yi-Lin Zhang, Yuanyu Wan, Tong Wei, and Min-Ling Zhang. Keeplora: Continual learning with residual gradient adaptation. In *International Conference on Learning Representations*, 2026.

Mark D. McDonnell, Dong Gong, Amin Parveneh, Ehsan Abbasnejad, and Anton van den Hengel. Ranpac: Random projections and pre-trained models for continual learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. In *International Conference on Learning Representations*, 2021.

German I. Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

Seulki Park, Jongin Lim, Younghan Jeon, and Jin Young Choi. Influence-balanced loss for imbalanced visual classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 735–744, 2021.

Seulki Park, Youngkyu Hong, Byeongho Heo, Sangdoo Yun, and Jin Young Choi. The majority can help the minority: Context-rich minority oversampling for long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6887–6896, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pp. 8748–8763, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 348–358, 2019.

Huiyi Wang, Haodong Lu, Lina Yao, and Dong Gong. Self-expansion of pre-trained models with mixture of adapters for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pp. 631–648, 2022a.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, et al. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 139–149, 2022b.

Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3014–3023, 2021.

Fei Ye, Yulong Zhao, Qihe Liu, Junlin Chen, Adrian G. Bors, Jingling Sun, Rongyao Hu, and Shijie Zhou. Dynamic siamese expansion framework for improving robustness in online continual learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Ping Hu, Dong Wang, Huchuan Lu, and You He. Boosting continual learning of vision-language models via mixture-of-experts adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23219–23230, 2024.

A Extended Related Work

Continual learning. Continual learning methods are commonly organized around replay, regularization, parameter isolation, or combinations of these strategies. Experience Replay stores a bounded subset of previous examples and reuses them when learning new data Rolnick et al. (2019), whereas iCaRL combines exemplars with nearest-mean classification Rebuffi et al. (2017). Regularization methods constrain parameter or representation changes, including EWC Kirkpatrick et al. (2017), LwF Li & Hoiem (2018), A-GEM Chaudhry et al. (2019), and contrastive continual learning Cha et al. (2021). DER and DER++ preserve feature or logit information in a replay-based expandable framework Yan et al. (2021). These approaches establish strong mechanisms for stability, but they do not explicitly model the additional uncertainty caused by a long-tailed class distribution.

Parameter-efficient adaptation has become especially relevant with pretrained vision transformers. Vision-language pretraining provides transferable representations Radford et al. (2021). L2P Wang et al. (2022b) and DualPrompt Wang et al. (2022a) learn prompt pools on top of a frozen backbone, while STAR-Prompt uses semantic two-level prompt selection in a vision-language model Bonicelli et al. (2024). RanPAC reduces continual parameter updates by combining a frozen random projection with class-prototype accumulation McDonnell et al. (2023). These methods motivate our use of frozen ViT representations and image-free statistics. However, they do not provide a class-conditional mechanism that jointly decides whether a new class should reuse an expert or receive new capacity.

Long-tailed and imbalanced continual learning. Long-tailed learning addresses unequal class frequencies through sampling, reweighting, calibrated losses, or classifier adjustment. Logit adjustment corrects prior-induced classifier bias using class frequencies at prediction time Menon et al. (2021), while influence-balanced objectives and minority oversampling modify optimization or feature distributions Park et al. (2021; 2022). In continual learning, partitioning reservoir sampling maintains a more balanced replay allocation Kim et al. (2020), and active class-incremental learning explicitly considers imbalance when selecting examples and classes Belouadah et al. (2020). Gradient Reweighting addresses imbalanced class-incremental learning through frequency-aware gradients and distillation He (2024). DCE extends the problem to imbalanced domain-incremental learning with frequency-aware collaborative experts and a dynamic selector based on historical class statistics Li et al. (2025).

These methods demonstrate that frequency imbalance changes continual optimization itself. Our method differs in using tail frequency to control image-free prototype transfer and in coupling the transfer source with route overlap. The resulting prior is applied at prediction time to classes with scarce observations, while Gaussian routing and expert assignment remain separate parts of the continual training state.

Expandable experts, adapters, and routing. Architectural expansion reduces direct parameter interference by allocating new capacity. DER++ dynamically expands representations while retaining historical information Yan et al. (2021). MoE-Adapters dynamically adds adapters to a pretrained CLIP model and uses a distribution-discriminative selector to choose between adapted and original paths Yu et al. (2024). SEMA learns distribution descriptors and an expandable router to reuse or add modular adapters on demand Wang et al. (2025). DSEF uses static and dynamic Siamese components, dynamic representation optimization, and feature fusion for online continual robustness Ye et al. (2025).

These methods highlight the value of separating transferable and task-specific computation. Their routing decisions, however, are not designed specifically for long-tailed class-conditional uncertainty. In our setting, the route assignment must also determine whether an existing expert has sufficient capacity for a new class and whether an additional expert or rank expansion is needed.

KeepLoRA projects new-task updates into residual directions that avoid principal subspaces associated with the pretrained model and prior task features Luo et al. (2026). Its stability-plasticity perspective motivates our gradient-direction protection. Our implementation applies this protection at the Specific-expert parameter level and combines it with function-preserving rank expansion and statistical expert assignment. The resulting mechanism addresses both the direction of parameter updates and the accessibility of the expanded expert pool.

Position of our method. Existing lines of work each address a useful projection of the problem: replay preserves examples, regularization preserves selected functions, prompt or adapter methods reduce trainable parameters, expandable experts provide new capacity, and long-tailed methods correct frequency bias. The remaining gap is their interaction under image-replay-free long-tailed class-incremental learning. A tail class may have an unreliable statistic, a new expert may alter route competition, and a protected historical expert may still be inaccessible.

DUALTRACK addresses this gap with one shared state design. Neutral coordinates support Gaussian route assignment; route assignments constrain capacity reuse and identify compatible historical sources; direction memory protects eligible historical updates; and count-adaptive prototype transfer supplies additional support when current evidence is weak. The experiments therefore evaluate not only final accuracy but also route agreement, tail accuracy, forgetting, and resource growth.

B How Class Structure Supports Continual Expert Reuse

The main method links expert selection to continual adaptation through a shared structure of registered classes. This appendix follows these dependencies: routing coordinates provide a reference for historical distributions, class distributions provide evidence for expert reuse, accepted reuse creates additional capacity requirements, and expert state determines the appropriate update protection. The same class-to-expert map also guides current-task learning and the use of historical feature statistics. Each mechanism has a defined scope: dual streams do not eliminate all feature drift, distributional compatibility does not imply unlimited capacity, and function continuity at expansion does not guarantee that every subsequent update preserves historical knowledge.

After the core mechanisms, we describe optional branches, the recorded reference configuration, and diagnostic definitions.

B.1 Why Routing Separation Precedes Statistical Matching

Providing a shared input while excluding direct Specific feedback. Stored distributions can guide reuse only if current samples are evaluated in coordinates that remain meaningful relative to those distributions. The frozen backbone uses twelve blocks, patch size 16, and input resolution 224, with zero-based indices. At an early adapted block, the frozen output is

$$u_l = F_l^0(h_l), \qquad l \in \{0, \dots, 7\}. \tag{19}$$

The shared residual supplies the next token sequence:

$$h_{l+1} = u_l + \frac{\alpha_s}{r_s} U_l Q_l u_l. \tag{20}$$

The orthogonal down-projection Q_l is fixed and the up-projection U_l starts at zero. Both later streams thus receive a common adapted input, with no initial change to the frozen representation. Block 8 has no such residual. The complete sequence initializes both late streams as $s_9 = h_9$; it is not reduced to a CLS vector at this boundary.

At a late layer, the two streams share the General LoRA parameters, while each updates its own input according to the recurrence in the main text. The router reads only $z_l = \mathrm{CLS}(b_l)$; the next routing layer receives the complete sequence $\mathrm{sg}(b_l)$, which contains no Specific residual. With the input and shared parameters fixed, changing Specific parameters alone therefore leaves routing coordinates unchanged. This separation allows the classification branch to learn class-specific residuals while isolating their direct effect on the reference used for expert selection.

From feature displacement to route consistency. Let $k^+ = m_l(c)$ be the registered expert of class c, and let $\delta_{l,k}$ denote the expert energy in the main text. Define

$$m_{l,c}(z) = \min_{k \neq k^+} [\delta_{l,k}(z) - \delta_{l,k^+}(z)]. \tag{21}$$

A positive margin means the registered expert has lower energy than every competitor. Class identity defines this diagnostic, not an inference input. For fixed statistics and assignments, suppose each competing energy difference is L-Lipschitz on a bounded neighborhood, with $L > 0$. A displacement within that neighborhood satisfies

$$m_{l,c}(z + \Delta z) \geq m_{l,c}(z) - L\|\Delta z\|_2. \tag{22}$$

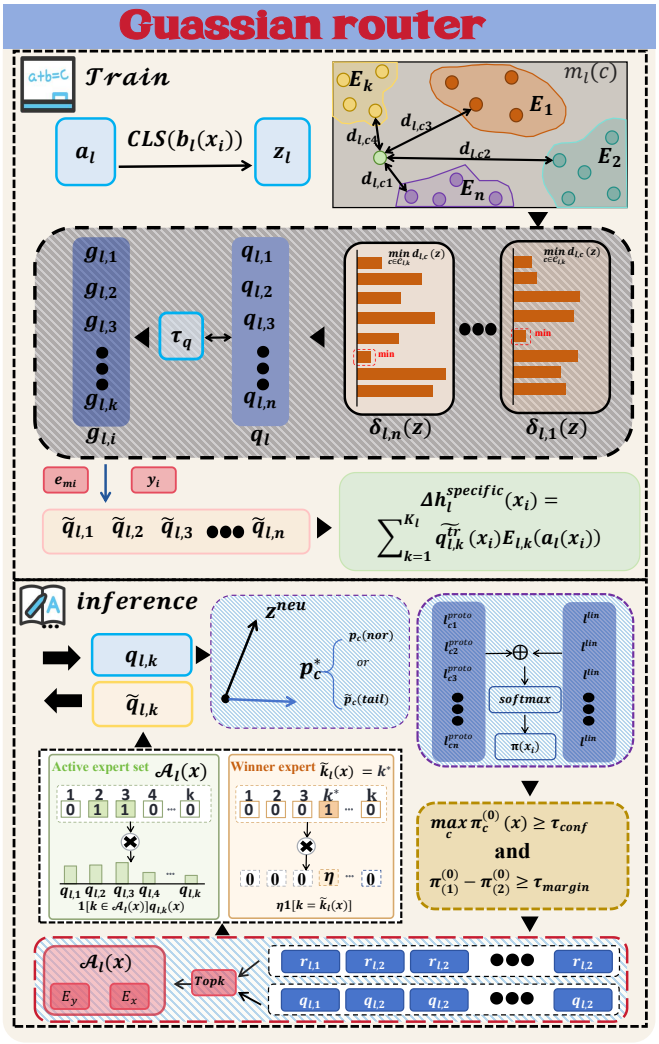


Figure 3: Detailed schematic of the Gaussian router used in DualTrack. The upper part summarizes the training-time feature and expert-statistics path; the lower part shows inference-time prototype readout, routing selection, and the final prediction path.

This follows by bounding each energy difference and taking the minimum. When $m_{l,c}(z) > 0$, a displacement smaller than $m_{l,c}(z)/L$ preserves the registered Top-1 route. Gaussian energies with positive variance floors admit a finite local bound on a bounded neighborhood, as does the minimum over finitely many class energies.

Dual streams exclude the direct effect of Specific updates from these sources of displacement. Shared adapters and statistics may still change, and Top-K weights need not remain constant. The analysis thus explains one source of routing consistency; it does not imply that routes remain unchanged throughout training.

Class regions connect sample routing to class allocation. Coordinate separation alone does not determine which expert fits a class. For each class, the router retains count $n_{l,c}$, mean $\mu_{l,c}$, diagonal variance $\nu_{l,c}$, and assignment $m_l(c)$. Variances in Eq. equation 10 are floored by ϵ. The class variance uses population variance of the collected feature batch; statistics are buffers rather than trainable neural parameters. Normal expert routing uses the minimum class energy, preserving distinct regions inside an expert instead of reducing them to one pooled center.

Expert-level moments provide a coarser view for candidate comparison and empty class-set fallback. For a nonempty registered set, its count is

$$n^E_{l,k} = \sum_{c \in \mathcal{C}_{l,k}} n_{l,c}. \tag{23}$$

The aggregated mean is

$$\mu^E_{l,k} = \frac{1}{n^E_{l,k}} \sum_{c \in \mathcal{C}_{l,k}} n_{l,c} \mu_{l,c}. \tag{24}$$

The diagonal variance follows the corresponding second moments:

$$\nu^E_{l,k} = \frac{\sum_{c \in \mathcal{C}_{l,k}} n_{l,c}(\nu_{l,c} + \mu^{\odot 2}_{l,c})}{n^E_{l,k}} - (\mu^E_{l,k})^{\odot 2}. \tag{25}$$

Here $\odot 2$ is elementwise squaring, and numerical variance flooring also applies to the aggregate. Updating an assignment changes which stored class moments contribute to an expert; historical statistics are not replaced by current-task samples alone. Sample routing and new-class allocation therefore share statistical evidence while making different decisions.

B.2 From Class Compatibility to Bounded Reuse

Allocation assesses an entire class. The router selects an expert for one sample, whereas allocation decides whether an expert can receive a new class. Before each task is optimized, a no-gradient scan collects current training features by class. The first task supports two initialization policies. With one class per route, each class is assigned directly to a separate route. Only the multi-class grouping policy forms initial groups using prototype proximity, sample confusion, Gaussian ambiguity, and capacity constraints. The recorded reference configuration uses one class per route. Subsequent tasks assess historical experts independently at each expert layer. Their per-task new-expert quota is neither a lifetime limit on the expert count nor a constraint on the separate first-task initialization branch.

For the scanned features $\mathcal{Z}_{l,c}$ of a new class, micro-coverage is

$$\rho_{l,c,k} = \frac{1}{|\mathcal{Z}_{l,c}|} \sum_{z \in \mathcal{Z}_{l,c}} \mathbf{1}\left[\min_{c' \in \mathcal{C}_{l,k}} d_{l,c'}(z) < \tau_{\mu,l}\right]. \tag{26}$$

This uses the same class energies as sample routing to measure how much of the new class falls inside existing regions. It is not the frequency of Top-1 wins: an expert can win against other poor candidates without adequately covering the class. The candidate k^* maximizes coverage; ties are resolved by the smaller macro distance $\bar{d}_{l,c,k}$, the mean new-sample energy under the expert-level Gaussian.

Suitability and assignment load require separate tests. Coverage provides evidence for reuse, but repeatedly assigning compatible classes to one expert can concentrate the learning demands of a new task. Ordinary absorption therefore requires four conditions to hold jointly. First, coverage must reach its threshold:

$$\rho_{l,c,k^*} \geq \eta_l. \tag{27}$$

The target must have available class capacity:

$$|\mathcal{C}_{l,k^*}| < C^{\max}_l. \tag{28}$$

Its task assimilation quota must also be available:

$$N^{\text{assim}}_{l,k^*,t} < M_l. \tag{29}$$

The macro-distance margin must satisfy

$$\bar{d}^{[2]}_{l,c} - \bar{d}_{l,c,k^*} \geq \gamma_l. \tag{30}$$

Here η_l is the coverage threshold, $C^{\max}_l$ the class capacity, M_l the per-task quota, and γ_l the margin threshold; $\bar{d}^{[2]}_{l,c}$ is the second-smallest macro distance. Since coverage selects the candidate, k^* need

not minimize macro distance, and the margin can be negative. With one candidate, the second distance is $+\infty$, so the margin test does not reject it. Capacity limits the registered class count, while TAQ limits new absorptions within a task. Neither directly measures the representational rank that a class requires, so satisfying these constraints does not establish that low-rank capacity is sufficient. If the coverage-selected candidate fails a test, the implementation does not search for a second feasible candidate. The optional reassignment mode instead skips ordinary absorption for unknown classes and prioritizes isolated routes; it is not an additional test in the ordinary policy.

Allocation sets the conditions for adaptation. Classes failing ordinary absorption form an isolated set $\mathcal{I}_{l,t}$. For a nonempty set and remaining new-expert quota $Q_{l,t}^{\mathrm{new}}$, the allocated expert count is

$$K_{l,t}^{\mathrm{new}} = \min\left(Q_{l,t}^{\mathrm{new}}, \max\left(1, \left\lceil \frac{|\mathcal{I}_{l,t}|}{C_l^{\max}} \right\rceil\right)\right). \tag{31}$$

Classes are ordered by increasing candidate coverage, then class index, and distributed round-robin among new routes. If allocation is disabled or its quota is exhausted, the isolated batch goes to the historical candidate associated with its highest-coverage class. This fallback preserves a valid assignment without automatically requesting rank growth. It is distinct from ordinary absorption and need not satisfy its tests.

The router and expert pool retain the same indices. Each eligible absorbing expert can undergo at most one rank-expansion event per task boundary, even if several classes are absorbed. Under the reference configuration, one expansion event increases the specific rank by the default increment of two, yielding $2 \to 4 \to 6 \to 8$, capped at $r_{\max} = 8$. Experts that are not selected for assimilation or grouping do not expand. Structural changes precede optimizer reconstruction and current-task training. The synchronized objects are expert identities and $m_l(c)$, not expert weights copied into Gaussian means or variances.

The fixed-rank ablation sets $r_{\mathrm{default}} = r_{\max} = 8$. Thus, every newly created specific expert starts at rank eight and no dynamic expansion is performed in that condition.

This map changes the class sets used by subsequent routing and supplies registered-path supervision for training. Ordinary absorption also triggers the capacity response in the next section, because distributional compatibility alone adds no new adaptation directions.

B.3 Why Reuse Needs Capacity and Update Protection

Compatible experts may lack the required output directions. Suppressing layer and expert indices, write $E(u) = (\alpha/r)BAu$, where $A \in \mathbb{R}^{r\times D}$ and $B \in \mathbb{R}^{D\times r}$. The output lies in $\mathcal{S}_r = \mathrm{col}(B)$. For a conceptual target update $\Delta W_c \in \mathbb{R}^{D\times D}$, let P_r project orthogonally onto this column space and define

$$R_c = (I - P_r)\Delta W_c. \tag{32}$$

The best output-subspace projection residual is

$$e_r(c) = \|R_c\|_F. \tag{33}$$

This target is explanatory: the algorithm does not estimate it to initialize new factors or decide when to expand.

With the old column span retained, $q = r' - r$ learned additional columns give $\mathcal{S}_{r+q} = \mathrm{col}([B, B^+])$. Let $\mathcal{U} = \mathrm{col}((I - P_r)B^+)$ be the added span orthogonal to the old space. Orthogonal decomposition yields

$$e_{r+q}(c)^2 = e_r(c)^2 - \|P_{\mathcal{U}}\Delta W_c\|_F^2. \tag{34}$$

Consequently,

$$e_{r+q}(c) \le e_r(c). \tag{35}$$

The residual decreases strictly when the new directions capture a target component that was previously unrepresentable. Appending q columns adds at most q independent directions. This result assumes that the old column space is retained and concerns the best projection onto the available output directions. Actual adaptation also depends on optimization and the learned factor A, so the inequality does not imply that accuracy must improve.

Capacity is added without an initial function jump. The main text initializes $A' = [A; A^+]$ and $B' = [(r'/r)B\ B^+]$, with Kaiming-initialized A^+ and $B^+ = 0$. At the expansion instant, every input satisfies

$$\frac{\alpha}{r'}B'A'u = \frac{\alpha}{r'}\left[\frac{r'}{r}BAu + B^+A^+u\right] = \frac{\alpha}{r}BAu. \tag{36}$$

Rescaling compensates for the changed LoRA scale. The realized column space is initially unchanged because the new B columns are zero. Their later gradients need not vanish, since A^+u is generally nonzero. Thus expansion retains the response at the boundary and makes new capacity available for subsequent training. The learned Specific residual remains outside the routing recurrence, connecting this adaptation back to coordinate separation.

Learning is concentrated in the added slices by masking old A rows:

$$\nabla A'[1\!:\!r, :] = 0. \tag{37}$$

The old B columns are also masked:

$$\nabla B'[:, 1\!:\!r] = 0. \tag{38}$$

Diagrams using A^T or B^T transpose these slices. Zero gradients alone do not guarantee zero parameter displacement under arbitrary optimizer state or weight decay. The continuity identity concerns initialization; later updates follow the configured optimizer.

Continued learning by historical experts requires a different protection. Prefix masking handles newly expanded experts. An eligible historical expert that continues to learn without being treated as new or just-expanded capacity instead uses direction memory. For vectorized tensors, define

$$\mathrm{N}(v) = \frac{v}{\max(\|v\|_2, \epsilon_N)}, \tag{39}$$

where $\epsilon_N > 0$ and zero maps to zero. Valid gradients of current-task capacity accumulate as

$$d_j = \mathrm{N}\left(\frac{1}{S_j}\sum_{s=1}^{S_j}\mathrm{N}(g_j^{(s)})\right). \tag{40}$$

Here S_j counts valid accumulations, and the memory is committed at the task boundary. During a later task, Eq. equation 18 projects eligible historical gradients onto the nonnegative ray of a usable d_j. It retains positive alignment, removes orthogonal components, and zeros nonpositively aligned gradients. For nonzero d_j,

$$\langle \widehat{g}_j, d_j \rangle \geq 0. \tag{41}$$

The memory summarizes historical gradients rather than the current gradient of an unavailable historical loss. Alignment therefore does not prove monotonic historical-loss improvement; momentum, weight decay, and adaptive preconditioning can also alter the eventual step.

New and just-expanded experts marked as current-task capacity bypass this projection. Missing gradients and parameters without usable memory are skipped. Accumulation is restricted to current new-capacity training parameters, including eligible expanded experts, rather than all historical parameters. Projection occurs before the optimizer step, but actual adaptation also depends on the parameter group's learning rate. The parameter `old_lr_mult` is implemented as a learning-rate multiplier for the `old_specific_lora` optimizer group. It is not a coefficient in the training loss and does not apply to every historical expert: frozen historical experts are excluded, and only historical specific experts that remain trainable are placed in this group.

With `old_lr_mult`=0, gradients may still be computed and projected, but the optimizer update for these trainable historical experts is zero. Thus, executing projection alone does not make those experts adapt in the reference configuration. Prefix masking and direction projection therefore apply to different update cases, rather than constituting two operations that every expert must undergo in sequence. Both constrain parameter adaptation, whereas dual streams address whether later samples can retrieve the appropriate expert.

B.4 Using Assignments to Guide Current-Task Learning

Class-level allocation must reach sample-level training. Each class has a registered expert, but autonomous routing can remain ambiguous for an individual sample. The training teacher uses the availability of current labels to pass class-level allocation decisions to sample-level training. For (x_i, y_i), let $\chi_{l,i}$ indicate a valid mapping and autonomous confidence no greater than the configured threshold. The adjusted probabilities are

$$\widetilde{q}_l(x_i) = (1 - \chi_{l,i}\beta_q)q_l(x_i) + \chi_{l,i}\beta_q e_{m_l(y_i)}. \tag{42}$$

The one-hot vector $e_{m_l(y_i)}$ selects the registered expert, and $\beta_q \in [0, 1]$. The gate uses $\max_k q_{l,k}(x_i)$ as autonomous confidence. This is a forward-weight correction, not a separate KL objective or a substitute for historical-gradient protection. Disabling the confidence gate makes every valid mapping eligible. A minibatch with at least one teacher hit uses soft expert aggregation; non-hit samples retain their autonomous probabilities. A batch with no hits uses hard routing. Allocation thus guides which experts receive current-task training without supplying labels at inference.

Classifier and expert protection act at different locations. Let $\mathcal{B}$ be a current-task minibatch and $\ell(x) = Wf(x) + b$ the linear logits. For classification set $\mathcal{C}^t_{\mathrm{CE}}$,

$$\mathcal{L}_{\mathrm{CE}} = -\frac{1}{|\mathcal{B}|} \sum_{(x,y)\in\mathcal{B}} \log \frac{\exp \ell_y(x)}{\sum_{c\in\mathcal{C}^t_{\mathrm{CE}}} \exp \ell_c(x)}. \tag{43}$$

The seen-class setting uses $\mathcal{C}^t_{\mathrm{CE}} = \mathcal{C}^t$; the current-task option restricts the set to new classes. Unseen classes are masked. Classifier expansion copies historical rows, initializes new weights with Xavier initialization, and sets new biases to zero. Historical-row gradients can be scaled; when the scale is zero, historical weights and biases are restored after each optimizer step.

This operation directly protects historical rows of the decision head. Specific protection instead constrains the feature adaptation that supplies the head's inputs. The two operate at different locations and cannot substitute for one another.

Auxiliary objectives constrain quantities left unfrozen by dual streams. Shared coordinates can still change despite the removal of Specific feedback. The implementation exposes configurable objectives for shared features and stored class representatives:

$$\begin{aligned}\mathcal{L}_{\mathrm{base}} = \mathcal{L}_{\mathrm{CE}} &+ \beta_a\mathcal{L}_{\mathrm{anchor}} + \beta_s\mathcal{L}_{\mathrm{supcon}} \\ &+ \beta_b\mathcal{L}_{\mathrm{blockKD}} + \beta_g\mathcal{L}_{\mathrm{generalKD}} + \beta_p\mathcal{L}_{\mathrm{proto}}.\end{aligned} \tag{44}$$

A zero coefficient disables a term. These are configuration-controlled training objectives, not additional routing modules or a claim that every experiment activates every term with identical coefficients. We call this the base objective to distinguish it from separately enabled extensions in Appendix B.8.

For normalized block-8 features $\bar{f}^{(8)}_i$ and fixed normalized class anchors e_c, the anchor term is cross-entropy on $\tau_a(\bar{f}^{(8)}_i)^{\mathsf{T}}e_c$. More explicitly, with V containing these anchors as columns,

$$\mathcal{L}_{\mathrm{anchor}} = -\frac{1}{|\mathcal{B}|} \sum_i \log \mathrm{softmax}\left(\tau_a(\bar{f}^{(8)}_i)^{\mathsf{T}}V\right)_{y_i}. \tag{45}$$

It supervises the shared stage before stream separation. Anchor expansion preserves existing entries; joint orthogonality is not guaranteed when the class count exceeds the feature dimension. Supervised contrastive regularization uses same-class positives $\mathcal{P}(i) = \{p \neq i : y_p = y_i\}$ and similarities $\zeta_{ia} = (\bar{f}^{(8)}_i)^{\mathsf{T}}\bar{f}^{(8)}_a/\tau_s$:

$$\mathcal{L}_{\mathrm{supcon}} = -\frac{1}{|\mathcal{I}_+|} \sum_{\substack{i\in\mathcal{I}_+ \\ p\in\mathcal{P}(i)}} \frac{1}{|\mathcal{P}(i)|} \log \frac{\exp \zeta_{ip}}{\sum_{a\neq i} \exp \zeta_{ia}}. \tag{46}$$

Here $\mathcal{I}_+$ contains samples with at least one positive, with zero loss when empty and no historical positive-image queue. Let r_8 be the zero-index block-8 CLS output, i.e., the ninth block output,

and r_8^- its counterpart from previous-task early adapters on the same current image. Normalized block-feature distillation is

$$\mathcal{L}_{\text{blockKD}} = \text{MSE}\left(\text{N}(r_8), \text{N}(r_8^-)\right). \tag{47}$$

The normalization is switch-controlled. The pretrained transformer is shared and the previous adapters are frozen. For frozen previous-task General modules G_l^-, General distillation uses the same block input:

$$\mathcal{L}_{\text{generalKD}} = \frac{1}{|\mathcal{L}_e|} \sum_{l \in \mathcal{L}_e} \text{MSE}\left(G_l(u_l), G_l^-(\text{sg}(u_l))\right). \tag{48}$$

MSE averages all tensor elements. Both terms are zero without a previous-task snapshot. They constrain shared quantities that stream separation does not freeze, using current images only.

When enabled, prototype supervision applies the classifier to one detached stored final-feature prototype per valid historical class:

$$\mathcal{L}_{\text{proto}} = \frac{1}{|\mathcal{H}|} \sum_{c \in \mathcal{H}} \text{CE}(W p_c + b, c). \tag{49}$$

Here $\mathcal{H}$ contains historical classes with valid prototypes, and logits are restricted to seen classes. This supplies historical class-level feature evidence without replaying images or generating pixels. The estimation quality of these representatives motivates the treatment of limited observations in the next section. Because the stored inputs are detached, this loss directly trains the classifier rather than the backbone. Even when historical classifier rows are fixed, new rows can learn to suppress responses to old prototypes.

B.5 From Scarce Observations to an Image-Free Class Prior

Expert capacity cannot replace missing observations. Correct routing and sufficient expert capacity do not ensure a reliable final-feature representative for a class with few samples. Prototype transfer addresses this estimation problem. Routing Gaussians summarize intermediate coordinates; prediction prototypes summarize final features. The two banks serve distinct purposes and remain separate.

In neutral collection mode, Specific residuals are disabled, and

$$p_c = \text{N}\left(\frac{1}{n_c} \sum_{i: y_i = c} \text{N}(f^{\text{neu}}(x_i))\right). \tag{50}$$

The count n_c comes from the collection loader, and the images are not retained. Normalization is defined in Eq. equation 39. Prediction uses a feature mode matched to collection. In the main collection path, updating the prototype bank requires prototype fusion to be enabled and a collection loader to be available. Enabling a prototype-supervision or transfer coefficient alone does not create this bank.

Registered paths help select historical support. For current class c with $0 < n_c \leq n_{\text{tail}}$ and historical class h, visual similarity is

$$s_{c,h} = p_c^{\mathsf{T}} p_h. \tag{51}$$

Let $\mathcal{V}_{c,h}$ contain layers with valid assignments for both. Their path overlap is

$$o_{c,h} = \frac{\sum_{l \in \mathcal{V}_{c,h}} \mathbf{1}[m_l(c) = m_l(h)]}{\max(1, |\mathcal{V}_{c,h}|)}. \tag{52}$$

The allocation map thus supplies a second source-selection signal. Visual similarity remains necessary: sources below $s_{\min}$ are excluded. The top K_p remaining sources ranked by $s_{c,h} + \lambda_o o_{c,h}$ form $\mathcal{H}_c$, with weights

$$w_{c,h}^p = \frac{\exp[(s_{c,h} + \lambda_o o_{c,h})/\tau_p]}{\sum_{a \in \mathcal{H}_c} \exp[(s_{c,a} + \lambda_o o_{c,a})/\tau_p]}. \tag{53}$$

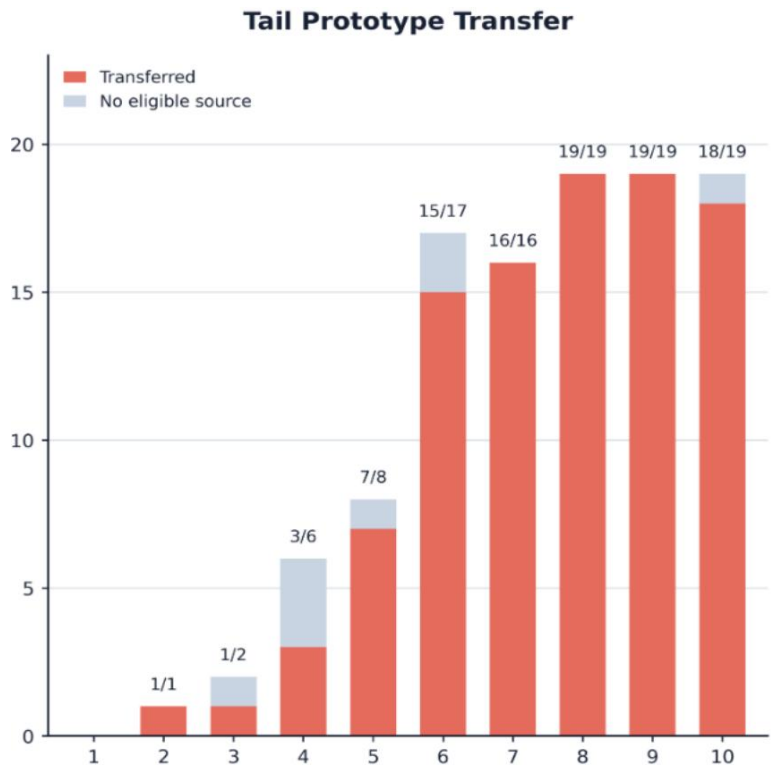


Figure 4: Tail transfer: supported/identified classes per task.

Their historical prior is

$$p_c^{\text{hist}} = \mathrm{N}\left(\sum_{h\in\mathcal{H}_c} w_{c,h}^{p} p_h\right). \tag{54}$$

Path overlap records shared assignments, not guaranteed semantic equivalence. Visual filtering and the following confidence weighting control how strongly this cue influences the estimate.

Prior strength depends on both sample scarcity and source relevance. For nonempty $\mathcal{H}_c$, visual confidence is

$$\gamma_c^{\text{vis}} = \text{clip}_{[0,1]}\left(\frac{\max_{h\in\mathcal{H}_c} s_{c,h} - s_{\min}}{\max(\epsilon, 1 - s_{\min})}\right). \tag{55}$$

The route factor is

$$\gamma_c^{\text{route}} = \frac{1}{2} + \frac{1}{2}\max_{h\in\mathcal{H}_c} o_{c,h}. \tag{56}$$

Without route metadata it is set to one. With metadata, the factor is at least one half: zero overlap alone does not veto transfer. For prior strength κ and maximum mixing weight $\lambda_{\max}$,

$$\omega_c = \lambda_{\max}\frac{\kappa}{n_c + \kappa}\gamma_c^{\text{vis}}\gamma_c^{\text{route}}. \tag{57}$$

The refined representative is

$$\widetilde{p}_c = \mathrm{N}[(1 - \omega_c)p_c + \omega_c p_c^{\text{hist}}]. \tag{58}$$

The count factor reduces dependence on the prior as observations grow, while confidence reduces transfer from weak historical support. Transfer is skipped without eligible sources or outside the configured low-count range. Only the refined current-class row is written back to the common buffer, allowing later tasks to reuse a previously refined source. This changes prediction prototypes, not Gaussian moments or training routing weights.

Historical support becomes available to more tail classes.

Figure 4 records prototype transfer in a completed ImageNet-A run with ten long-tailed Class-IL tasks and seed 42. As the historical prototype bank grows, eligible sources support most identified tail classes in later tasks: all 19 tail classes receive transfer at each of Tasks 8 and 9, followed by 18 of 19 at Task 10. This illustrates the operation of source selection in Eqs. equation 53–equation 58: historical evidence can supplement scarce observations, while the eligibility test permits a class to retain its empirical prototype when no suitable source is available. Coverage measures how often eligible historical support is available. Counts are reported per task rather than cumulatively, and the gray bar segment denotes classes without an eligible historical source. The matched ablation in Table 5 shows a 2.06-point gain on CUB-200 when tail transfer is enabled, indicating that the selected prior can improve final accuracy under the evaluated protocol. We therefore use tail transfer as a targeted prior for scarce classes.

Readout must match the feature mode of historical evidence. For a matching feature f, prototype logits are

$$\ell_c^p = \tau_{\text{proto}}\, \mathrm{N}(f)^\top \widetilde{p}_c. \tag{59}$$

An optional count bias addresses frequency effects at the score level:

$$\Delta\ell_c = \lambda_{\text{count}} \log(n_{\max}/n_c), \quad 0 < n_c < n_{\text{tail}}. \tag{60}$$

Here $n_{\max}$ is the largest valid collected count and λ_{count} controls the bias. This strict threshold differs from the inclusive transfer threshold $n_c \le n_{\text{tail}}$. The bias is added only when enabled, before prototype probabilities are normalized; it does not change the stored class representative. Linear and prototype probabilities are normalized over seen classes. Their score is

$$S_c(x) = (1-\lambda_f)\log P_c^{\text{lin}}(x) + \lambda_f \log P_c^{\text{proto}}(x). \tag{61}$$

The score is not already a normalized log posterior. If probabilities are needed, normalize it over observed classes:

$$P_c^{\text{fused}}(x) = \frac{\exp S_c(x)}{\sum_{j\in\mathcal{C}^t} \exp S_j(x)}. \tag{62}$$

This normalization leaves the predicted class unchanged. For $\lambda_f \in [0,1]$, prediction follows

$$\widehat{y} = \arg\max_{c\in\mathcal{C}^t} S_c(x). \tag{63}$$

At $\lambda_f = 1$, only prototypes contribute. Neutral prototype-only prediction multiplies Specific outputs by zero, so route changes do not directly change that final feature branch. Shared metadata connects the mechanisms, but not every output mode uses every inference branch.

B.6 From Class Predictions to Label-Free Routing

The teacher in Appendix B.4 uses current labels to identify registered paths during training. At inference, refinement instead uses first-pass class posteriors $\pi^{(0)}(x)$, aggregated through the same map:

$$P_l(k \mid x) = \sum_{c:m_l(c)=k} \pi_c^{(0)}(x). \tag{64}$$

The predicted expert is

$$\widehat{k}_l = \arg\max_k P_l(k \mid x). \tag{65}$$

Several plausible classes can support one expert, so marginalization differs from taking the route of the single most likely class. It converts class-level predictive evidence into an expert-level decision through the structure learned during allocation.

Incorrect predictions can also mislead routing. Refinement therefore requires the configured confidence and margin gates. The default evaluation distribution w_l^0 can be a one-hot argmax, normalized Gaussian Top-K weights, uniform Top-K weights, or the full q_l. It need not equal q_l. If candidate restriction is enabled, the autonomous top-K_a experts and $\widehat{k}_l$ form $\mathcal{A}_l(x)$. For positive restricted mass, the base weights are

$$w_{l,k}^{\text{base}} = \frac{\mathbf{1}[k\in\mathcal{A}_l(x)]w_{l,k}^0}{\sum_j \mathbf{1}[j\in\mathcal{A}_l(x)]w_{l,j}^0}. \tag{66}$$

If the restricted mass is zero, restriction retains w_l^0 instead; with restriction disabled, $w_l^{\text{base}} = w_l^0$. For normalized w_l^{base}, a nonnegative boost b_q gives

$$w_{l,k}^{\text{boost}} = \frac{w_{l,k}^{\text{base}} + b_q \mathbf{1}[k = \widehat{k}_l]}{1 + b_q}. \tag{67}$$

Samples failing the gate retain the autonomous policy. Neither training labels nor test task identities enter this step. Its effect on the final prediction depends on whether the readout uses Specific features, as distinguished in Appendix B.5.

Layer-posterior gates use the minimum across layers of the winning expert probability and of the top-two expert margin. Class-posterior mode instead tests class confidence; full-path mode first aggregates probability over classes sharing an entire route tuple and tests path confidence. These are alternative confidence conventions. Hard override and additional refinement forwards are also optional and change inference cost. Their confidence thresholds are not interchangeable.

B.7 How These Dependencies Determine the State Retained across Tasks

These dependencies determine the task-boundary order. A current-data scan supplies allocation evidence; allocation updates assignments and requests ordinary-absorption expansions; changed parameters are synchronized with the optimizer before training. Assignments provide gated path supervision, and expert state selects the applicable update protection. Task-boundary statistics and direction memories supply the next task's historical information. Prototype refinement uses path metadata without rewriting routing Gaussians. When a current-statistics loader is available, the local training path can refresh current-class routing statistics at task end. This does not re-estimate all historical class moments after shared updates. Optional prototype transport precedes prototype updating and uses current images; shared teacher snapshots are also committed for the next task. The modules are connected through persistent state, rather than by applying every operation to every sample.

For C_t observed classes and expert layers $\mathcal{L}_e$, final-feature prototypes, Gaussian moments, counts, and assignments require

$$\mathcal{M}_{\text{stats}} = O(C_t D) + O\left(\sum_{l \in \mathcal{L}_e} (C_t + K_l) D\right). \tag{68}$$

This counts statistics rather than total memory. Specific parameters and direction buffers additionally scale with $\sum_{l,k} 2Dr_{l,k}$; optimizer state and teacher snapshots are separate. Expanding one linear LoRA expert by Δr adds $2D\Delta r$ parameters. TAQ and allocation quota constrain structural decisions, rank limits bound individual expert size, and stored statistics replace retained historical images. Replay-free learning thus avoids historical image storage without implying constant parameter or statistical memory.

Gaussian and prototype buffers form part of model state, but gradient-direction dictionaries and nonpersistent random anchors need explicit checkpoint handling. Saving an ordinary model state alone should not be assumed to restore all dynamic-training state. The reference schedule and diagnostics below specify how this state is used and how its effects should be measured.

A compact trainable budget accompanies retained capacity.

The growing state described above should be distinguished from the parameters eligible for training in an individual task. Figure 5 reports this task-wise budget in the ImageNet-A run analyzed in Figure 4. The logged trainable count ranges from 175K to 283K, or 0.20%–0.33% of the final 86.81M-parameter model. The growing state described above should be distinguished from the parameters trained for an individual task. Figure 5 reports the task-wise trainable parameter count in the ImageNet-A run analyzed in Figure 4. The logged count ranges from 175K to 283K, or 0.20%–0.33% of the final 86.81M-parameter model. At each task, DualTrack trains only a small subset of the accumulated model parameters, corresponding to the adapters and experts used along the current task's training paths. This focuses optimization on components assigned to the current task while retaining the remaining expert capacity for future tasks.

The five-task long-tailed CIFAR-10 comparison in Table 2 provides a direct measurement of training cost: DualTrack takes 35.55 minutes, uses 5,424 MiB of peak GPU memory, and has

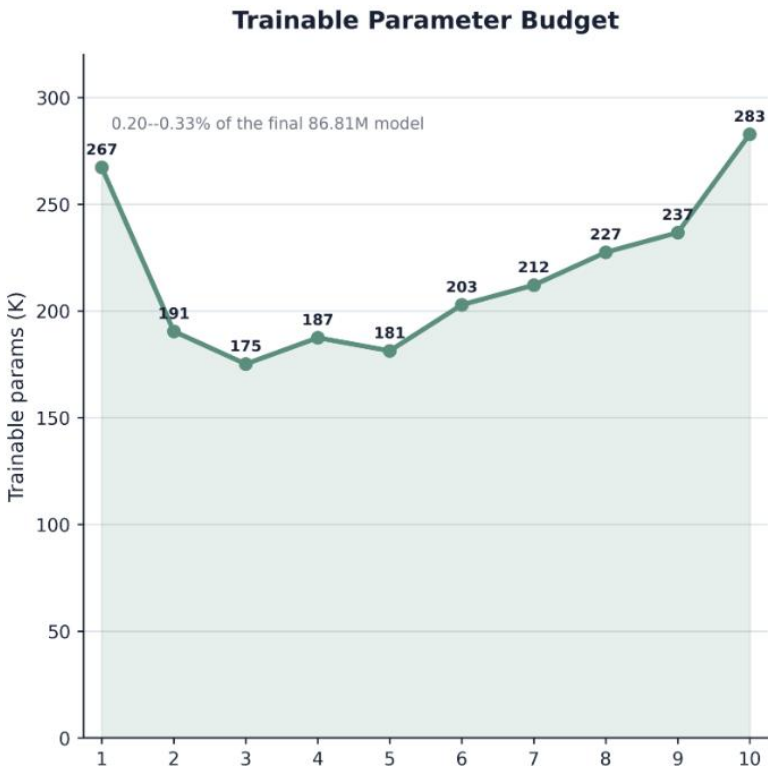


Figure 5: Trainable parameters per task (K).

0.121M trainable parameters. Its training time is lower than that of several compared methods and close to RanPAC, while its peak memory is below KeepLoRA and CL-LoRA and higher than the other listed methods. Together with the classification results in the main table, these measurements show that DualTrack can achieve strong performance while training a small, task-relevant parameter subset, with competitive training time and a measurable GPU-memory cost. The run uses ten long-tailed Class-IL tasks and seed 42. Point labels are in thousands, and all percentages use the final model as the denominator.

B.8 Optional Branches and Their Dependencies

The core mechanism does not prescribe every representation or inference option exposed by the implementation. The following branches address additional consistency, coordinate-drift, or readout questions and are documented for reproducibility and ablation. They are enabled separately and must not be attributed to a run merely because the code supports them. For enabled extra terms with available dependencies, the full objective is

$$\mathcal{L}_{\text{active}} = \mathcal{L}_{\text{base}} + \sum_{j \in \mathcal{J}_{\text{extra}}} \lambda_j \mathcal{L}_j. \tag{69}$$

Here $\mathcal{J}_{\text{extra}}$ excludes terms already included in the base objective. Generic logits/feature KD compatibility arguments are zero terms in this DualTrack path; General KD, block KD, and prototype supervision are separately implemented options.

Neutral classification and consistency. Neutral CE evaluates the classifier after setting the Specific residual scale to zero. A neutral/routed consistency term uses the symmetric KL form

$$\begin{aligned}\mathcal{L}_{\text{symKL}} = {} & \tfrac{1}{2}\,\mathrm{KL}(\mathrm{sg}(\pi^{\text{neu}}) \| \pi^{\text{route}}) \\ & + \tfrac{1}{2}\,\mathrm{KL}(\mathrm{sg}(\pi^{\text{route}}) \| \pi^{\text{neu}}).\end{aligned} \tag{70}$$

It is not Jensen–Shannon divergence. The extra forward can refresh cached features used by later optional losses. Here π^{neu} and π^{route} are class probabilities in the neutral and routed branches; the KL terms are averaged over the batch. Both optional losses require the neutral feature mode.

Optional heads. The metric head applies a normalized projection before classification. The neutral head is an independent linear classifier. The neutral residual head is a $D \to r_h \to \mathrm{GELU} \to C$ logit correction whose final projection is zero-initialized. These heads are auxiliary branches, not additional Specific experts or Gaussian updates. Here r_h is the residual-head bottleneck size and C is the output class count. The neutral head can add old-prototype CE and previous-head logit KD; the latter requires a teacher, whose construction currently depends on the prototype-transport switch. The residual head detaches input features and base logits, learns a correction using current CE and optional prototype CE/KD, and can mask historical output columns. It does not directly update the backbone.

An optional route-level constraint. For current classes assigned to experts that contain historical classes, a route-anchor term compares macro Gaussian energies. Let $\mathcal{I}_l^a$ contain eligible samples at layer l and $\mathcal{L}_a = \{l : |\mathcal{I}_l^a| > 0\}$. Define the sample penalty

$$a_{l,i} = \left[d^{\mathrm{macro}}_{l,m_l(y_i)}(z_{l,i}) - \min_{k \neq m_l(y_i)} d^{\mathrm{macro}}_{l,k}(z_{l,i}) + \Delta_a \right]_+ . \tag{71}$$

The implemented reduction averages within each eligible layer and then across those layers:

$$\mathcal{L}_{\mathrm{RA}} = \frac{1}{|\mathcal{L}_a|} \sum_{l \in \mathcal{L}_a} \frac{1}{|\mathcal{I}_l^a|} \sum_{i \in \mathcal{I}_l^a} a_{l,i}. \tag{72}$$

Here $[v]_+ = \max(0, v)$ and Δ_a is the margin. Layer and interval switches control evaluation. Detached later routing streams may leave no trainable gradient path for this term; its measured nonzero value alone does not show that it updates those coordinates.

Aligning historical prototype coordinates. Stream separation excludes Specific feedback but allows shared features to evolve. Optional prototype transport estimates this remaining coordinate change without historical images. Collect neutral feature matrices X^- and X^+ from previous and current models on current images, with rows denoting samples. Centered alignment uses

$$(X^- - \mathbf{1}\bar{x}^-)^{\mathsf{T}}(X^+ - \mathbf{1}\bar{x}^+) = U\Sigma V^{\mathsf{T}}. \tag{73}$$

The right-acting orthogonal transform is

$$R = UV^{\mathsf{T}}. \tag{74}$$

Here $\bar{x}^-$ and $\bar{x}^+$ are row means, $\mathbf{1}$ broadcasts them, and U, Σ, V are SVD factors. For a stored column prototype, define the transported candidate

$$p_c^{\mathrm{align}} = R^{\mathsf{T}}(p_c - (\bar{x}^-)^{\mathsf{T}}) + (\bar{x}^+)^{\mathsf{T}}. \tag{75}$$

With interpolation strength $\zeta \in [0, 1]$, its update is

$$p_c^{\mathrm{new}} = \mathrm{N}\left((1-\zeta)p_c + \zeta p_c^{\mathrm{align}}\right). \tag{76}$$

The fit uses unnormalized sample features, whereas stored prototypes are normalized. Transferring current-class alignment to historical classes is an approximation rather than a guarantee of distribution recovery. This branch changes prototype coordinates, not the historical Gaussian bank.

Cosine subcenters. An optional class-specific clustering procedure stores multiple normalized centers $p_{c,j}$ instead of relying exclusively on one center. Prediction uses $\tau_{\mathrm{proto}} \max_j \mathrm{N}(f)^{\mathsf{T}} p_{c,j}$, where j indexes valid subcenters. Tail transfer updates the single-center buffer, not these subcenters, so its direct influence must not be assumed when the subcenter classifier is selected.

Frozen prototype ensemble. A separate optional memory stores centers in the unadapted ViT space and combines its predictions with adapted predictions. This adds storage and a frozen-feature forward. It is distinct from neutral features, which retain shared adaptation.

Additional inference evidence and cost. Route-prediction TTA averages route evidence from the original and horizontally flipped image. Selective TTA is triggered by the first-pass entropy or top-class margin; it does not use an original/flip JS score. Prototype TTA is a separate switch that averages final prototype probabilities. Entropy uses natural logarithms. Selective evaluation flips only the chosen subset; the subset threshold has meaning only with the same entropy normalization and confidence convention.

The optional calibration head is trained against the TTA teacher:

$$\mathcal{L}_{\text{cal}} = \text{KL}\left(\bar{\pi}^T(x) \| \,\text{softmax}(g_\psi(f(x)))\right). \tag{77}$$

Here $\bar{\pi}^T$ is the teacher probability, g_ψ is a lightweight calibration head, and ψ are its parameters. The current forward path does not use this head as a proven single-pass replacement for TTA. The teacher uses original/flip routed predictions by default. Student features are not detached, so the calibration objective can also update feature parameters.

An optional CLIP text branch projects visual features into a fixed text-aligned space and adds cosine text logits. Its entropy and tail gates are evaluation options; it is separate from the visual prototype transfer described above. The optional JS gate compares neutral and routed predictions of the same image, not original and flipped images. Text vectors, class ordering, templates and the visual projection must be aligned. Training does not automatically apply the same tail/entropy gates as evaluation.

Alternative routers are separate ablations. Attention and InfoMax routers are alternatives to hierarchical Gaussian routing, not simultaneous components. An InfoMax router can optimize conditional entropy minus marginal entropy, whereas the Gaussian router does not supply that differentiable objective. Their task-boundary interfaces require compatibility checks before substitution. Likewise, a residual adapter is an alternative to Specific LoRA factorization, not an additional expert component.

B.9 Reference Configuration and Update Order

We retain the script defaults for imbalanced TinyImageNet recorded in the earlier implementation audit. These are not the formal experimental settings for every dataset. The reference uses the local legacy-timm CLIP ViT-B/16 checkpoint, three late expert layers, and eight early shared-adapter layers. Early shared rank and scale are 8 and 16; General rank is 4; Specific initial rank is 2 with rank ceiling 8. First-task initialization uses one class per route.

The global Gaussian policy uses temperature 1, micro-energy threshold 1000, coverage threshold 0.60, a target capacity of ten classes per expert, one ordinary assimilation per expert per task, and eight newly created experts per layer per subsequent task. These are separate limits: the last quantity is not a lifetime global expert cap.

The script defaults to ten tasks, three epochs per task, seed 43, batch size 32, and base learning rate 0.01. General and early shared learning-rate multipliers are 0.1, while the multiplier for non-new historical experts is zero. Historical classifier rows are restored after each step. Shared adapters are not frozen after the first task in this configuration. Task-specific environment overrides must be recorded with the resulting experiment.

The enabled auxiliary coefficients are 1 for General distillation, 0.5 for bridge/block KD, 0.1 for fixed anchors, 0.1 for supervised contrastive learning, and 1 for prototype supervision. Anchor logits have scale 16 and the contrastive temperature is 0.2. The registered-path teacher uses $\beta_q = 0.5$ and confidence ceiling 0.85. Thus the reference objective is

$$\begin{aligned}\mathcal{L}_{\text{ref}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{generalKD}} + 0.5\mathcal{L}_{\text{blockKD}} \\ + 0.1\mathcal{L}_{\text{anchor}} + 0.1\mathcal{L}_{\text{supcon}} + \mathcal{L}_{\text{proto}}.\end{aligned} \tag{78}$$

The teacher changes forward weights rather than adding a loss to this sum. Eligible historical gradients are projected before the optimizer step. With a zero historical learning-rate multiplier, this projection alone does not make historical expert parameters adaptive.

Neutral prototype prediction uses cosine scale 16 and fusion weight 1. Tail transfer uses count threshold 10, prior strength 8, maximum blend 0.35, three sources, cosine cutoff 0.20, route weight

0.50, and source temperature 0.10. Route evidence uses per-layer posterior aggregation, a default top-four autonomous route, a candidate set formed from the top two autonomous experts plus the predicted expert, and boost 1. The inference confidence and margin thresholds are 0.55 and 0.05. Route-evidence horizontal flipping is enabled.

Prototype transport, additional neutral/metric/residual heads, text logits, route-anchor training, and the calibration head are not enabled by this reference launcher. Final prototype TTA is a different option from route-evidence flipping and is not implied by it. These optional mechanisms are documented above for ablation completeness, not as claims about the reference result.

Before each task's optimization, a current-data pass collects class Gaussian statistics and finalizes assignments. Training uses those registered routes. At task end, current prototypes are collected and refined, and the model commits gradient-direction memory and shared teacher snapshots. Gaussian moments are not automatically re-estimated for all historical classes after these shared updates. The current-data refresh described in Appendix B.7 is not an all-class historical refresh. Optional transport, when enabled, precedes prototype updating and uses current images only.

B.10 Route Drift, Classification Forgetting, and Execution Cost

Coordinate separation and update protection address different failure sources. Their evaluation should therefore distinguish route movement from classification forgetting instead of treating them as the same event. Let D_l indicate that a sample's selected expert at layer l differs from the preceding evaluation, and let F indicate a transition from correct to incorrect classification. The diagnostic reports

$$\widehat{P}(F \mid D_l) = \frac{\#(F \cap D_l)}{\#D_l}. \tag{79}$$

For samples whose routes do not change, the corresponding diagnostic is

$$\widehat{P}(F \mid \neg D_l) = \frac{\#(F \cap \neg D_l)}{\#(\neg D_l)}. \tag{80}$$

Here $\#$ denotes sample counts, $\cap$ intersection, and $\neg$ negation. The implementation expresses the rates as percentages. Empty conditioning sets have no statistically meaningful rate even when a numerical guard returns zero; denominators should accompany reported results. The comparison identifies association rather than proving that route changes cause forgetting.

Sample matching currently uses loader, batch, and within-batch indices. Consequently, dataset order, batch partitioning and transformations must remain stable across evaluations. The supplementary single-sample probe compares stored outputs and route paths for the reference loader's first sample; it does not replace dataset-level analysis.

Further diagnostics include registered versus selected experts, top-K coverage, class-to-expert confusion, complete route frequencies, and per-class feature moments. Registered-route disagreement is not by itself proof of a classification error or evidence that the registered route is optimal. Evaluation labels are used to measure these quantities, not to choose experts in the deployed classifier.

Registered-path agreement becomes harder to maintain.

Figure 6 measures path agreement in the same ImageNet-A run as Figures 4 and 5. Exact path match falls from 76.68% after Task 1 to 39.93% after Task 10; old-class path mismatch rises overall from 42.81% after Task 2 to 58.23% after Task 10. These observations quantify the increasing difficulty of reproducing registered paths as the class and expert spaces expand. They are consistent with the scope of the analysis in Appendix B.1: removing direct Specific feedback does not freeze shared coordinates or guarantee unchanged routing. Agreement with a registered path is distinct from a change in a sample's path between evaluations, so these curves do not estimate Eqs. equation 79 and equation 80. They serve as operational diagnostics, not evidence of superiority over a baseline or proof that routing disagreement causes forgetting. This is the ten-task long-tailed Class-IL run with seed 42. Path match uses all evaluated samples, while mismatch uses old-class samples only; their denominators differ. Task 1 has no old classes, so its logged zero is a placeholder.

Measure the executed prediction path. Setting the Specific scale to zero suppresses its residual after its computation in the current implementation; it does not generally skip the expert and router

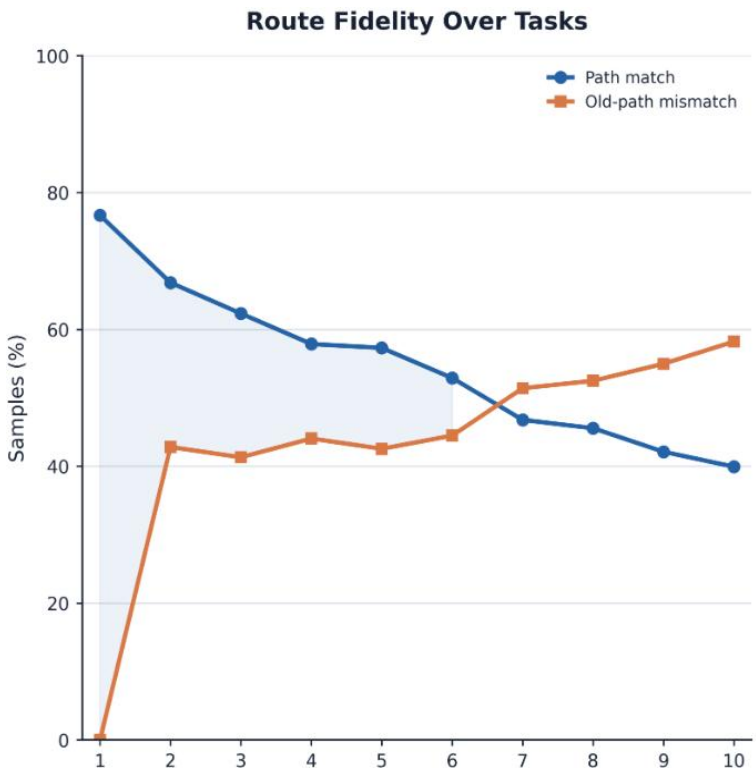


Figure 6: Registered-path agreement over tasks.

Table A1: Sensitivity to key hyperparameters on CUB-200. Experiments use ten tasks, three epochs per task, a batch size of 32, and seed 1993. Each run varies one hyperparameter while keeping all other settings fixed. Values are final average Class-IL accuracy (%) after task 10, and the best result in each row is shown in **bold**.

Hyperparameter	Candidate 1	Candidate 2	Candidate 3
Prototype-fusion weight	0.5: 74.35	**1.0: 80.98**	1.5: 76.13
Route-absorption threshold	0.5: 79.77	0.6: 80.16	**0.7: 80.98**
Early-anchor weight	0.0: 81.66	**0.1: 81.74**	0.2: 81.04
Registered-path teacher weight	0.0: 79.55	**0.1: 80.98**	0.2: 79.11

execution. The detached routing stream can require additional frozen-block computation. Route-evidence TTA, prototype TTA, iterative feedback, and frozen-feature ensembles further change the number of forwards. Resource reporting should therefore measure the complete inference call, with warm-up and device synchronization, rather than equating cost to a single nominal ViT pass.

With neutral prototype-only classification, a predicted route cannot directly change a Specific residual that is subsequently multiplied by zero. Any benefit attributed to expert routing in that setting must distinguish its training-time effect from its final inference contribution. The reference script can still incur route-evidence computation in this setting.

Training-time profiling records selected stages, but a profiler entry point alone does not establish that every reported metric was collected. Parameter counts, statistical buffers, optimizer memory, CPU memory, peak allocated/reserved GPU memory, throughput, and end-to-end wall time have different accounting scopes.

C HYPERPARAMETER SENSITIVITY ANALYSIS

Sensitivity analysis. Table A1 shows that prototype fusion is most effective at weight 1.0, achieving 80.98% accuracy, compared with 74.35% at 0.5 and 76.13% at 1.5. The fusion rule is $(1-w)\log p_{\text{lin}} + w \log p_{\text{proto}}$: weight 0.5 retains substantial linear-head influence, whereas weight 1.5 assigns the linear head a negative coefficient and may amplify disagreement between the two heads. The same table shows that increasing the route-absorption threshold from 0.5 to 0.7 improves accuracy from 79.77% to 80.98%. A stricter threshold may prevent weakly matched new classes from entering existing experts, although route-event statistics are needed to verify the resulting assignments.

Table A1 also indicates that the auxiliary mechanisms benefit from moderate weights. Setting the early-anchor weight to 0.1 yields 81.74%, close to 81.66% when its loss is disabled, while increasing the weight to 0.2 reduces accuracy to 81.04%. Moderate alignment with class anchors may help organize intermediate features, but a stronger constraint may interfere with fine-grained discrimination. For the registered-path teacher, weight 0.1 improves accuracy from 79.55% without teacher mixing to 80.98%; weight 0.2 instead lowers it to 79.11%. This suggests that limited guidance can stabilize expert selection, whereas stronger guidance may restrict adaptation when registered paths and evolving routing evidence differ.

D ADDITIONAL EXPERIMENTAL RESULTS

The following supplementary experiments distinguish discrimination within the observed class set from detection of inputs outside that set. Both use the same CUB200-Imbalance reference run, connecting the posterior and prototype readout described above to measured prediction behavior.

D.1 EVALUATION PROTOCOL FOR CLASS DISCRIMINATION AND OOD DETECTION

We supplement the classification results with two diagnostics on CUB200-Imbalance: discrimination among observed classes and detection of inputs outside the currently observed class set. These diagnostics examine the predictions of one trained model; they introduce no additional training objective or learned rejection head.

Training and readout. The run uses ten tasks with twenty classes per task in a fixed class order, three epochs per task, seed 42, batch size 32, and SGD with learning rate 0.01. Training uses the existing linear subsampling schedule, with the class-dependent retention fraction decreasing from 1.0 to 0.1. The backbone is the local legacy-timm CLIP ViT-B/16. Evaluation uses neutral features and prototype-only prediction, with fusion weight $\lambda_f = 1$ and cosine scale $\tau_{\text{proto}} = 16$; visual tail transfer is enabled and the text branch is disabled. This is an independent reference run, not a validation of the candidate configuration suggested by the preceding sensitivity study: its early-anchor coefficient is 0.1 and registered-path teacher weight is $\beta_q = 0.5$, with confidence ceiling 0.85. Final prototype TTA is disabled. No test labels enter the prediction call, and OOD images do not update model parameters, prototypes, or routing statistics.

D.2 CLASS DISCRIMINATION ACROSS THE LONG-TAILED STREAM

For each observed class, we treat its test images as positive and all other observed-class images as negative, using its final posterior as the score. Macro one-vs-rest (OvR) AUROC averages class-wise AUROCs, whereas micro AUROC pools all sample–class decisions. We report top-1 accuracy alongside AUROC because binary ranking and multiclass identification measure different properties.

Table A2 shows that macro OvR AUROC remains above 99.7% at the measured stages, while final micro top-1 accuracy is 81.17%. Strong OvR ranking can coexist with multiclass errors: the correct class may outrank most alternatives yet be exceeded by a competing class. These measurements establish neither perfect class separation nor the absence of forgetting; the observed label space also grows between stages.

For frequency-stratified analysis, we sort classes by their actual retained training counts and divide them into approximately equal-sized groups, breaking ties by class index. The final head,

Table A2: Class discrimination on the complete observed-class CUB test set. All performance values are percentages; one seed is used. Micro top-1 weights images equally, while macro top-1 weights classes equally.

Task	Classes	Images	Top-1 accuracy		OvR AUROC	
			Micro	Macro	Micro	Macro
1	20	515	93.98	93.68	99.82	99.77
5	100	2864	85.34	85.41	99.84	99.75
9	180	5206	80.87	81.01	99.78	99.73
10	200	5794	81.17	81.30	99.79	99.74

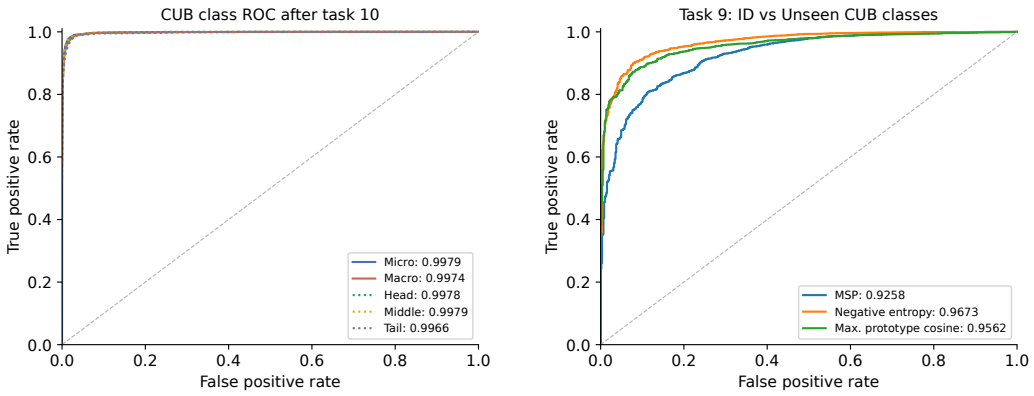


Figure 7: Supplementary ROC diagnostics on CUB200-Imbalance. Left: classification after task 10, including frequency-stratified macro curves. Right: detection after task 9, comparing 5,206 test images from the 180 observed classes with 588 images from the 20 remaining classes. ID is the positive class in the right panel. Legend values are AUROCs on a $[0, 1]$ scale. Both panels use seed 42; they evaluate different discrimination problems and their AUROCs are not interchangeable.

middle, and tail groups contain 66, 67, and 67 classes, respectively. Their macro top-1 accuracies are 82.31%, 81.91%, and 79.70%, and their macro OvR AUROCs are 99.78%, 99.79%, and 99.66%. These groups differ from the count threshold that activates prototype transfer. The head–tail accuracy gap is 2.62 percentage points, but this descriptive comparison does not isolate transfer effectiveness: class identity and position in the fixed task sequence also differ between groups.

D.3 Task-Relative and External OOD Detection

OOD populations. After tasks 1, 5, and 9, observed CUB classes define in-distribution (ID) inputs, and the remaining CUB classes define task-relative near-OOD inputs. These classes are unseen during incremental training, not necessarily during backbone pretraining. After task 10, all 200 CUB classes are observed. We therefore use an external far-OOD set consisting of 900 CIFAR-10 test images: 100 sampled without replacement from each of its nine non-bird classes, with sampling seed 20260921. All inputs use the same 224×224 resizing and CUB normalization. The external set differs substantially in semantics and original image resolution.

Detection scores and metrics. We evaluate three fixed scores using the existing posterior and prototype bank. Let $\mathcal{C}^t$ be the observed class set after task t, $P_c^{\text{fused}}(x)$ the normalized final posterior for image x, $f^{\text{neu}}(x)$ its neutral feature, and $\widetilde{p}_c$ the stored normalized prototype, including transfer

Table A3: OOD detection using three scores from the same CUB200-Imbalance model. AUROC and FPR95 are percentages. Near-OOD denotes remaining CUB classes; far-OOD denotes non-bird CIFAR-10 images. Counts report ID/OOD test images. Extra decimal places distinguish the nearly saturated far-OOD results from exact perfect detection.

Task	OOD (ID/OOD counts)	Score	AUROC ↑	FPR95 ↓
1	Near (515/5279)	MSP	93.17	35.22
		Negative entropy	96.85	13.43
		Max. cosine	98.01	7.39
5	Near (2864/2930)	MSP	90.28	46.42
		Negative entropy	94.85	17.68
		Max. cosine	95.56	18.63
9	Near (5206/588)	MSP	92.58	35.88
		Negative entropy	96.73	18.71
		Max. cosine	95.62	26.19
10	Far (5794/900)	MSP	99.9920	0.11
		Negative entropy	99.9987	0.00
		Max. cosine	> 99.9999	0.00

where applicable. With N denoting vector normalization, the scores are

$$\begin{aligned} s_{\mathrm{MSP}}(x) &= \max_{c \in \mathcal{C}^t} P_c^{\mathrm{fused}}(x), \\ s_{\mathrm{NE}}(x) &= \sum_{c \in \mathcal{C}^t} P_c^{\mathrm{fused}}(x) \log P_c^{\mathrm{fused}}(x), \\ s_{\cos}(x) &= \max_{c \in \mathcal{C}^t} \mathrm{N}(f^{\mathrm{neu}}(x))^{\mathsf{T}} \widetilde{p}_c. \end{aligned} \tag{81}$$

MSP is maximum softmax probability, NE is negative entropy with natural logarithms, and $s_{\cos}$ is maximum prototype cosine similarity. Higher scores indicate ID membership. OOD ROC treats ID as the positive class. FPR95 is the fraction of OOD inputs accepted as ID at the first empirical threshold attaining at least 95% ID true-positive rate; this is a metric threshold, not a validated deployment threshold. Scores and temperatures are fixed before evaluation, without fitting to OOD test labels.

Task-relative detection. Across tasks 1, 5, and 9, negative entropy and maximum prototype cosine similarity consistently outperform MSP in both AUROC and FPR95 (Table A3). After task 9, when 180 bird classes have been observed, negative entropy achieves 96.73% AUROC and 18.71% FPR95, compared with 92.58% and 35.88% for MSP (Figure 7, right). These results show that the learned posterior and prototype geometry retain useful evidence for identifying task-relative unseen classes as the observed class set grows. The best score varies by stage and metric, indicating complementary detection signals rather than a uniformly dominant scoring rule.

External OOD detection. After all ten tasks, the three scores also separate CUB test images from non-bird CIFAR-10 images with near-saturated AUROC (Table A3). We treat this as a supplementary external-shift check; the unseen CUB classes provide the more relevant fine-grained test because the CIFAR-10 comparison also differs in semantics and original image resolution. The seed results characterize the detection behavior of the trained model, while the component ablations assess the contributions of its mechanisms.

D.4 Robustness under Extreme Long-Tailed Imbalance

We next test whether the end-of-stream performance is stable when the class-frequency ratio is reduced from 1:10 to 1:100. The comparison uses ten CIFAR-100 tasks, seed (1993), and the corrected Class-IL evaluator, which exposes the same set of classes to every test task at a given training stage. Figure 8 reports average accuracy after tasks 3, 7, and 10. The decrease from 1:10 to 1:100 at task 10 is 2.15 percentage points for CL-LoRA, 1.64 points for KeepLoRA, and only 0.12 points for

Table A4: Robustness to long-tail imbalance and tail-stage task retention on CIFAR-100. Average accuracy is measured after learning tasks 3, 7, and 10. Tail-stage accuracy reports the final Class-IL accuracy of stream tasks 8–10, and Tail avg. is their mean. Because the protocol permutes class order, these are the final three stream tasks rather than a pure class-frequency partition. All results use seed 1993 and ten tasks.

Method	Ratio	$\mathcal{A}_{@3}$ (%)	$\mathcal{A}_{@7}$ (%)	$\mathcal{A}_{@10}$ (%)	Tail T8 (%)	Tail T9 (%)	Tail T10 (%)	Tail avg. (%)
CL-LoRA	1:10	87.77	81.59	78.18	87.40	86.80	88.90	87.70
CL-LoRA	1:100	86.70	78.56	76.03	84.00	86.50	87.80	86.10
KeepLoRA	1:10	87.63	83.01	80.42	82.30	76.90	80.40	79.87
KeepLoRA	1:100	87.37	81.86	78.78	81.10	73.90	79.00	78.00
DualTrack	1:10	87.97	85.63	83.85	89.70	83.20	87.60	86.83
DualTrack	1:100	87.50	85.07	83.73	89.20	83.00	87.20	86.47

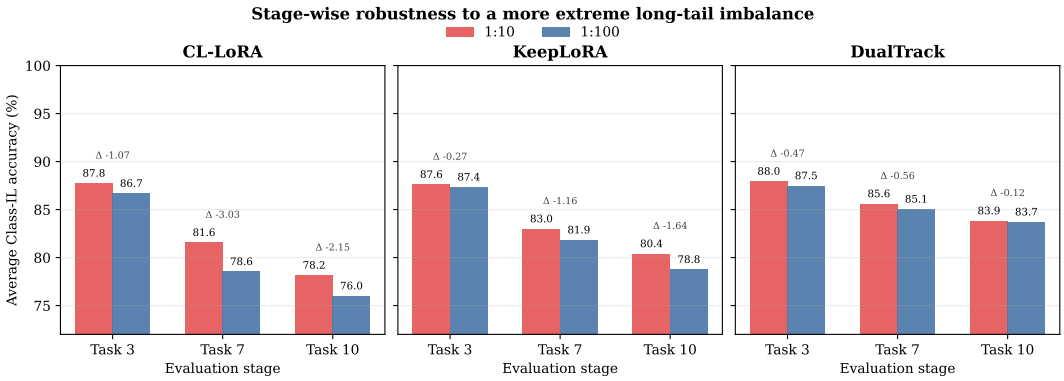


Figure 8: Stage-wise robustness on long-tailed CIFAR-100. Bars report average Class-IL accuracy after tasks 3, 7, and 10 for the 1:10 and 1:100 training streams. Δ denotes the 1:100 minus 1:10 difference in percentage points.

DUALTRACK. The smaller change indicates greater robustness to the more extreme tail under this protocol.

Figure 9 reports standard average forgetting: for each previously learned task, we subtract its current accuracy from the best accuracy observed before the current stage and average over previous tasks. At the final stage, CL-LoRA reaches 14.57/15.81 percentage points, KeepLoRA 5.38/5.19 points, and DUALTRACK 5.46/5.17 points under 1:10/1:100, respectively. Thus, DUALTRACK maintains a forgetting level close to KeepLoRA while retaining higher final average accuracy.

The final three stream tasks are reported as tail-stage tasks in Table A4. Their mean accuracy for DUALTRACK is 86.83% under 1:10 and 86.47% under 1:100. Because the benchmark permutes class order, tasks 8–10 measure end-of-stream tail-stage retention rather than an isolated class-frequency transfer test.

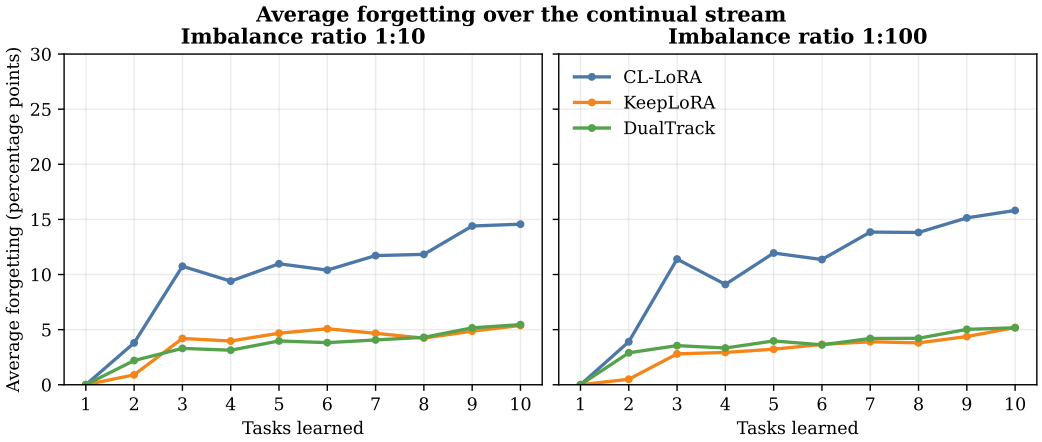


Figure 9: Average forgetting during the ten-task CIFAR-100 stream. Each point averages forgetting over previously learned tasks, using each task's best accuracy before the current stage.