REWARD-PROPORTIONAL ROUTING OF LORA EXPERTS FOR CONTINUAL LEARNING IN LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual learning (CL) of large language models (LLMs) commonly assigns task-specific low-rank adaptation (LoRA) experts to a frozen backbone and trains a router that selects experts for each input without task identity. Although the number of experts can vary for different inputs, it is inappropriately fixed in advance, as existing routers are only trained to identify the best experts. A small change in the loss can swap the entire selection, which degrades performance on previous tasks with their experts frozen. This paper proposes a method of reward-proportional routing (RPR) that trains the router toward a distribution over expert combinations that is proportional to their reward. The distribution changes continuously with the loss, and reading it up to a cumulative probability mass determines both the number and the composition of experts for each input. RPR learns the distribution with a generative flow network (GFlowNet) and freezes one expert per task. On three CL benchmarks with a frozen T5-large, RPR achieves the highest mean accuracy on all eight task orders, with 70.6% against 68.7% for the second-best method and the lowest average forgetting on the 15-task Long Sequence benchmark. On tasks never trained on, RPR reaches 58.2% against 62.1% for the unadapted backbone, whereas the compared methods produce labels of previous tasks and reach at most 1%.

1 INTRODUCTION

Continual learning (CL) of large language models (LLMs) is used to assign task-specific low-rank adaptation (LoRA) experts (Hu et al., 2022) to a frozen backbone and to train a router that selects experts for each input, because the task identity of an input is unavailable at inference (Zhao et al., 2024; Wang et al., 2024; 2025a; Liang et al., 2026). Freezing experts from previous tasks blocks interference between their parameters (Chen et al., 2023; Ge et al., 2025), while the router continues to be trained on every subsequent task.

Forgetting in this setting, therefore, also arises from the router's selection. When the router is updated for a later task, the experts it selects for an input from an earlier task can change, even though no parameter of those experts has moved. For overparameterized linear regression, the gating network of a mixture of experts (MoE) in CL must stop being updated after sufficient training for the model to converge (Li et al., 2025); in lifelong model editing, the routing of the same input is inconsistent between training and inference (Wang & Li, 2024); and the expert assigned to an input keeps changing during training even without a task stream (Dai et al., 2022). As the number and composition of the experts that an input requires also vary across inputs (Huang et al., 2024; Guo et al., 2025; Zhao et al., 2025), the router has to decide which and how many experts to apply for each input.

However, existing methods train the router to return a predetermined number of experts (Zhao et al., 2024; Liu et al., 2024; Wang et al., 2025a). Router scores trained in this way concentrate on the few experts that minimize the loss (Shazeer et al., 2017; Dou et al., 2024; Zhao et al., 2026). The number of selected experts is then set by a quantity fixed in advance, namely k, a probability threshold, or a sparsity coefficient, applied to scores that do not reflect how many combinations solve the input comparably well (Huang et al., 2024; Yang et al., 2024; Wang et al., 2025b; Zhao et al., 2025). A small change in the loss during later training can switch the entire selection (Proposition 1; Figure 1,

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.

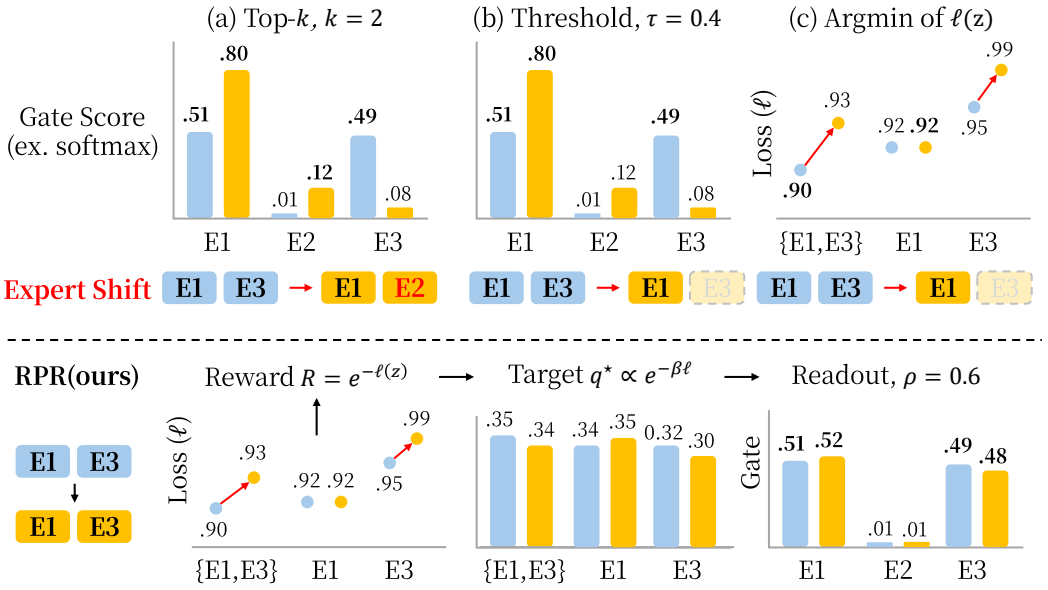


Figure 1: Routing of one input before (■) and after (■) a later task that updates the not-yet-frozen expert E3, while E1 is unchanged. Top: routers trained to minimize the loss change their selection. (a) The top-k gate moves from $\{\mathrm{E1}, \mathrm{E3}\}$ to $\{\mathrm{E1}, \mathrm{E2}\}$, and (b) the threshold gate, which applies every expert whose score exceeds τ, moves to $\{\mathrm{E1}\}$; gate scores are illustrative. (c) The minimizer of $\ell(z)$ also moves from $\{\mathrm{E1}, \mathrm{E3}\}$ to $\{\mathrm{E1}\}$. Bottom: RPR defines the reward $R = e^{-\ell(z)}$ from the same losses and routes toward the target $q^\star \propto e^{-\beta\ell}$, which moves by a total variation of 0.016. The readout applies experts in decreasing order of the first-step marginal of $q^\star$ until the cumulative mass reaches ρ, so the selection $\{\mathrm{E1}, \mathrm{E3}\}$ is kept.

top). As a result, too few, too many, or wrong experts can be selected for inputs of earlier tasks, and performance on those tasks degrades even though their experts are frozen.

This paper proposes a method of reward-proportional routing (RPR) that trains the router toward a probability distribution over expert combinations, proportional to how well each combination solves the input, that is, its reward. The probability is divided among combinations that solve an input comparably well and concentrates on a combination that solves it clearly best. Reading the distribution up to one cumulative probability mass selects a number and composition of experts that vary with the input. As the probability also changes continuously with the loss (Proposition 2), a small change in the loss during later training moves the distribution by a correspondingly small amount instead of switching it entirely (Figure 1, bottom). RPR learns this distribution with a generative flow network (GFlowNet) router trained by trajectory balance (TB) (Bengio et al., 2021; Malkin et al., 2022) and freezes one expert per task.

The contributions of this paper are as follows.

- **A routing target and readout that existing CL routers do not use.** RPR replaces the training target of the router rather than its architecture: a GFlowNet whose backward policy counts the orderings of each combination learns a reward-proportional distribution over variable-size combinations of LoRA experts, and the experts for each input are read deterministically from its first-step probabilities instead of being sampled.

- **Guarantees against both causes of mis-routing.** The target moves by at most $\tanh(\beta\epsilon/2)$ in total variation (TV) distance under a loss change of size ϵ, independently of the number of combinations, whereas the minimizer of the expected loss can move by a maximum distance of one (Propositions 1–2). At a global minimizer of TB, the first-step probability equals the per-expert marginal of the target (Lemma 1), so that one cumulative-mass rule determines the number and composition of experts for both expert updates and inference.

- **Consistent gains on various CL benchmarks.** RPR achieves the highest mean accuracy on all eight task orders and the lowest average forgetting on Long Sequence. Replacing its objective with REINFORCE lowers the average accuracy on Long Sequence from 70.6% to 30.4%. No fixed number of experts reaches the accuracy of its readout, and on tasks never trained on, RPR reaches 58.2% against 62.1% for the unadapted backbone, whereas the compared methods stay at or below 1%.

2 Related Work

Continual learning of language models with modular experts. Parameter-efficient CL of language models stores task knowledge in modules on a frozen backbone, and methods differ in how a module is chosen at inference. Isolation methods constrain each new LoRA module to a subspace orthogonal to, or free of interference with, those of previous tasks (Wang et al., 2023a; Liang & Li, 2024), identify the task of an input before selecting its module (Wang et al., 2023b), or concatenate task prompts (Razdaibiedina et al., 2023). Routing methods compose modules without task identity. SAPT aligns module learning and selection through shared attention and replays generated pseudo-samples to keep the selection from forgetting (Zhao et al., 2024); MoCL composes modules by matching task vectors (Wang et al., 2024); GainLoRA integrates task-specific LoRA branches through gates trained to suppress new branches on old tasks (Liang et al., 2026); AM-LoRA mixes LoRA modules through attention (Liu et al., 2024); and mixtures of LoRA experts have been applied to continual information extraction (Wang et al., 2025a) and continual multimodal instruction tuning (Ge et al., 2025). The routers of these methods are trained by loss minimization, and Li et al. (2025) show, for overparameterized linear regression, that the gating network of an MoE in CL must stop being updated after sufficient training for the model to converge. RPR keeps the router trainable throughout the stream and changes its training target instead.

Adaptive numbers of experts. MoE models commonly activate a fixed number of experts per token (Shazeer et al., 2017; Fedus et al., 2022), and their load-balancing losses act on the utilization of experts averaged over inputs, which leaves the routing of each input free to concentrate on a single combination (Lemma 3 in Appendix A). Several methods vary the number of experts. Top-p routing adds experts in decreasing probability until their cumulative probability exceeds a threshold, together with an entropy-minimizing loss that keeps the router from spreading probability (Huang et al., 2024); XMoE thresholds the cumulative probability of small experts (Yang et al., 2024); AdaMoE adds null experts (Zeng et al., 2024); DynMoE compares gate scores with trainable per-expert thresholds (Guo et al., 2025); ReMoE controls the number of active ReLU gates with an adaptive L_1 penalty (Wang et al., 2025b); and Ada-K samples the number of experts from an allocator trained by proximal policy optimization to maximize the likelihood under a penalty on the expected number (Zhao et al., 2025). In each case the router is trained by loss minimization or reward maximization, and the number of experts follows from a threshold, a penalty, or a cost coefficient. RPR also reads experts up to a cumulative mass, but from a distribution whose shape is set by the relative rewards of expert combinations rather than concentrated by loss minimization (Section 3.4).

Generative flow networks. GFlowNets learn a sequential sampler of compositional objects whose terminal distribution is proportional to a reward (Bengio et al., 2021; 2023). Trajectory balance improves their credit assignment (Malkin et al., 2022), and GFlowNet training corresponds to variational inference (Malkin et al., 2023) and to entropy-regularized reinforcement learning (Tiapkin et al., 2024). GFlowNets have served as amortized posteriors over discrete latent structures, alternated with an M-step on the generative model (Hu et al., 2023), over dropout masks (Liu et al., 2023), and over latent reasoning in LLMs (Hu et al., 2024). They extend to multiple objectives (Jain et al., 2023). RPR trains a conditional GFlowNet over variable-size sets of LoRA experts and does not sample at inference: The applied experts are read deterministically from the router's first-step probabilities.

3 REWARD-PROPORTIONAL ROUTING

3.1 PROBLEM SETUP

Tasks $\mathcal{T}_1, \ldots, \mathcal{T}_T$ arrive one at a time, the data of a finished task is not revisited, and inputs arrive without a task identifier at test time. The backbone is frozen. A pool of M LoRA experts is attached to the adapted linear layers: expert $m \in [M]$ holds a low-rank pair $B^{(m)}A^{(m)}$ of rank r at every adapted layer and a routing key $\boldsymbol{e}_m$. For each input x the router selects one combination $z \subseteq [M]$, applied at every adapted layer with weight W_0 and input $\boldsymbol{u}$ as

$$\boldsymbol{h} = W_0\boldsymbol{u} + \frac{s}{|z|}\sum_{m\in z} B^{(m)}A^{(m)}\boldsymbol{u}, \tag{1}$$

with LoRA scaling s. The admissible combinations are $\mathcal{Z} = \{z \subseteq [M] : 1 \le |z| \le K\}$. For an input x with target y, $\ell(z, x)$ denotes the token-averaged cross-entropy of y under the backbone with z applied, and $R(z, x) = \exp(-\ell(z, x))$ is the reward of z. As the weights in equation 1 are uniform and carry no parameters, $\ell(z, x)$ depends on x, z, and the parameters of the experts in z alone (Appendix A.5).

3.2 REWARD-PROPORTIONAL TARGET

RPR trains the router toward the distribution proportional to the tempered reward $R(z, x)^\beta$,

$$q^\star(z \mid x) = \frac{R(z,x)^\beta}{\sum_{z'\in\mathcal{Z}} R(z',x)^\beta} = \frac{\exp(-\beta\,\ell(z,x))}{\sum_{z'\in\mathcal{Z}}\exp(-\beta\,\ell(z',x))}, \tag{2}$$

with temperature $\beta > 0$. Instead of identifying the best combination, the target assigns each combination probability in proportion to how well it solves the input. It is the unique minimizer of $\mathbb{E}_{z\sim p}[\ell(z, x)] - \beta^{-1}H(p)$ over distributions p on $\mathcal{Z}$ (Proposition 3 in Appendix A), whereas routing by loss minimization or reward maximization optimizes an expectation that is linear in p, the expected loss or the expected reward. The two kinds of objectives respond differently to a change of the loss.

Proposition 1 (Loss minimization switches under small loss changes)**.** *Fix x and let $\ell = \ell(\cdot, x)$ have a unique minimizer $z^\star$ and gap $g = \min_{z\neq z^\star}\ell(z) - \ell(z^\star) > 0$. The point mass $\delta_{z^\star}$ is the unique minimizer of $\mathbb{E}_{z\sim p}[\ell(z)]$ over distributions p on $\mathcal{Z}$. If $g < 2\epsilon$, some ℓ' with $\|\ell' - \ell\|_\infty \le \epsilon$ has a unique minimizer $z' \neq z^\star$, so the two minimizing distributions satisfy $\mathrm{TV}(\delta_{z^\star}, \delta_{z'}) = 1$. If $g > 2\epsilon$, every such ℓ' has the unique minimizer $z^\star$.*

Proposition 2 (The reward-proportional target moves continuously)**.** *Let q and q' be the targets equation 2 of losses ℓ and ℓ' with $\|\ell' - \ell\|_\infty \le \epsilon$ for some $\epsilon > 0$. Then $q'(z) \ge e^{-2\beta\epsilon}q(z)$ for every $z \in \mathcal{Z}$, and*

$$\mathrm{TV}(q, q') \;\le\; \tanh(\beta\epsilon/2) \;\le\; \min\left\{\beta\epsilon/2,\; 1 - e^{-2\beta\epsilon}\right\}, \tag{3}$$

independently of $|\mathcal{Z}|$. The first inequality is attained for some ℓ and ℓ' whenever $|\mathcal{Z}| \ge 2$.

As proposition 1 holds because the expected loss is linear in p, its minimizer is a vertex of the simplex, and raising the loss of $z^\star$ by ϵ while lowering that of the runner-up by ϵ reverses their order whenever $g < 2\epsilon$. Proposition 2 holds because the change multiplies each unnormalized probability by a factor in $[e^{-\beta\epsilon}, e^{\beta\epsilon}]$, and the total variation of such a reweighting is largest when the probability splits into two groups moved in opposite directions (proofs in Appendix A). For example, let the candidates for an input be $\{1, 3\}$, $\{1\}$, and $\{3\}$ with losses 0.90, 0.92, and 0.95, and let later training update expert 3, raising the losses of $\{1, 3\}$ and $\{3\}$ to 0.93 and 0.99 ($\epsilon = 0.04$, $g = 0.02 < 2\epsilon$). The minimizer of the expected loss moves from $\{1, 3\}$ to $\{1\}$ and expert 3 is dropped, whereas the target with $\beta = 2$ moves from $(0.349, 0.335, 0.316)$ to $(0.344, 0.351, 0.305)$, a total variation of 0.016 within the bound 0.040 of equation 3 (Figure 1).

3.3 LEARNING THE TARGET WITH A GFLOWNET ROUTER

RPR learns the target with a GFlowNet router because normalizing equation 2 requires the loss of every combination in $\mathcal{Z}$, which has $\sum_{k=1}^{K}\binom{M}{k} = 39{,}202$ elements for $M = 16$ and $K = 8$, whereas a GFlowNet learns a sampler proportional to a reward from the rewards of sampled combinations

alone (Bengio et al., 2021). A trajectory τ starts from the empty set s_0, adds one expert per step, and ends with a terminal action $\perp$ at a combination z, so the size of z is itself decided by the router. At state s_i the forward policy scores each available expert and the terminal action, which has its own key $\boldsymbol{e}_\perp$, by the cosine similarity between a query and their keys,

$$P_F(a \mid s_i, x) \;\propto\; \exp\big(\sigma\,\langle \hat{\boldsymbol{g}}_\phi(\boldsymbol{q}(x), \boldsymbol{c}_i, \boldsymbol{p}_i),\, \hat{\boldsymbol{e}}_a \rangle\big), \qquad a \in ([M] \setminus s_i) \cup \{\perp\}, \tag{4}$$

where $\boldsymbol{q}(x)$ is the backbone representation of x, $\boldsymbol{c}_i$ summarizes the experts in s_i, $\boldsymbol{p}_i$ encodes the step, $\hat{\cdot}$ denotes L_2 normalization, and σ is a learned scale. The terminal action is unavailable at s_0 and forced at $|s_i| = K$. The policy and an input-conditioned normalizer $Z_\phi(x)$ minimize the TB loss (Malkin et al., 2022)

$$\mathcal{L}_{\mathrm{TB}}(\tau, x) = \left(\log \frac{Z_\phi(x)\, P_F(\tau \mid x)}{P_B(\tau \mid z)\, R(z, x)^\beta} \right)^2, \qquad P_B(\tau \mid z) = \frac{1}{|z|!}, \tag{5}$$

where $P_F(\tau \mid x)$ is the product of the step probabilities. A combination z is reached by the $|z|!$ orderings of its elements, and the uniform backward policy P_B assigns them equal probability. Because P_B differs across combinations of different sizes, it cannot be absorbed into Z_ϕ: treating it as a constant shifts the optimum to $P_F(z \mid x) \propto |z|!\, R(z, x)^\beta$, a factor of up to $8! = 40{,}320$ in favor of the largest combinations. When equation 5 is zero on every complete trajectory, the router terminates at z with probability $q^\star(z \mid x)$ (Malkin et al., 2022).

3.4 Reading the number and composition of experts

RPR reads the number and the composition of experts from the router's first-step probabilities instead of sampling from the learned distribution, as a GFlowNet is typically used, so inference is deterministic and evaluates the router once.

Lemma 1 (First-step marginal)**.** *If equation 5 is zero on every complete trajectory for an input x, which a policy class that can represent $q^\star(\cdot \mid x)$ attains at its global minimum, the probability that the router selects expert m at its first step is*

$$q_1(m \mid x) = P_F(m \mid s_0, x) = \sum_{z \ni m} \frac{q^\star(z \mid x)}{|z|}. \tag{6}$$

The marginal divides the probability of each combination evenly among its members, so it spreads over more experts when good combinations are large or numerous. RPR sorts the experts as $m_{(1)}, m_{(2)}, \ldots$ in decreasing q_1 and applies the shortest prefix whose mass reaches ρ:

$$\hat{z}(x) = \{m_{(1)}, \ldots, m_{(n(x))}\}, \qquad n(x) = \min\Big\{ n \le K : \sum_{j=1}^{n} q_1(m_{(j)} \mid x) \ge \rho \Big\}, \tag{7}$$

with $n(x) = K$ if the mass is not reached within K experts, and combines the selected experts by equation 1. With a fixed ρ, $n(x)$ varies with the input because the shape of $q_1(\cdot \mid x)$ does. In the example of Section 3.2, q_1 assigns 0.51 and 0.49 to experts 1 and 3 before the change of the loss and 0.52 and 0.48 after it, so equation 7 with $\rho = 0.6$ selects $\{1, 3\}$ in both cases. The marginal inherits the continuity of the target, $\mathrm{TV}(q_1, q_1') \le \mathrm{TV}(q^\star, q^{\star\prime})$ (Corollary 1), so the selected set changes only for inputs whose marginals lie within this distance of a boundary of equation 7 in the maximum norm.

3.5 Consolidation

When the pool holds at least one expert per task ($M \ge T$), as the compared methods that add one module per task presume, RPR consolidates one expert per task. At the end of each task, it freezes the expert that received the largest share of that task's selections and fixes its key to the mean query of the task's inputs. Frozen experts are removed from the action space while later tasks are trained, so no update is spent on parameters that cannot change, and they are returned to it at inference. Fixing the key anchors a frozen expert to the region of query space that its task occupies. Because the rewards of combinations drawn only from frozen experts no longer change, the relative probabilities that the target assigns to these combinations are unaffected by later training (Proposition 4 in Appendix A.5). A variable size is what makes one frozen expert per task sufficient: a combination consisting of frozen experts exists from the first freeze, whereas with a fixed size k none exists until k experts are frozen. When $M < T$, RPR keeps every expert trainable instead (Section 4.4).

Algorithm 1 RPR: one training step on task t, consolidation at its end, and inference

procedure TRAINSTEP(x)
  $s \leftarrow \emptyset$
  repeat ▷ sample a trajectory; frozen experts F_{t-1} are masked
    $a \sim P_F(\cdot \mid s, x)$ by equation 4; **if** $a \neq \perp$ **then** $s \leftarrow s \cup \{a\}$
  until $a = \perp$
  update the policy, σ, trainable keys, and $\log Z_\phi$ by $\nabla \mathcal{L}_{\mathrm{TB}}$ equation 5 with $z = s$
  if TB residual $<$ threshold **then** ▷ expert update
    $\hat{z} \leftarrow$ readout equation 7 over $[M] \setminus F_{t-1}$; update the LoRA pairs of $\hat{z}$ by $\nabla \ell(\hat{z}, x)$
  end if
end procedure

procedure ENDTASK(t) ▷ consolidation, applied when $M \geq T$
  $m_t \leftarrow$ trainable expert with the largest share of task t's selections; fix $\boldsymbol{e}_{m_t}$ to the task's mean query
  $F_t \leftarrow F_{t-1} \cup \{m_t\}$; freeze $A^{(m_t)}$ and $B^{(m_t)}$
end procedure

procedure INFER(x) ▷ no mask: frozen experts are selectable
  compute $q_1(\cdot \mid x)$ with one evaluation of the router; **return** $\hat{z}(x)$ by equation 7, combined by equation 1
end procedure

3.6 TRAINING

Training alternates two updates on disjoint parameters (Algorithm 1), and the experts frozen after earlier tasks, F_{t-1}, are masked in both while task t is trained. The routing update samples a trajectory from P_F and minimizes equation 5 over the policy network, the scale σ, the keys of trainable experts, and $\log Z_\phi$. The expert update is taken once the TB residual falls below a threshold; it forms the combination with the readout equation 7 restricted to trainable experts and minimizes $\ell(\hat{z}, x)$ over the LoRA matrices of the selected experts. The experts are thus trained with the same readout rule by which they are applied at inference, restricted to trainable experts.

4 EXPERIMENTS

4.1 SETUP

Benchmarks. Three CL benchmarks are used, none with a task identifier at test time. The Long Sequence benchmark (Razdaibiedina et al., 2023) chains 15 classification datasets, five from Zhang et al. (2015), four from GLUE (Wang et al., 2018), five from SuperGLUE (Wang et al., 2019), and IMDB (Maas et al., 2011), in three published orders (Orders 4–6). The Standard CL benchmark chains four text-classification datasets of Zhang et al. (2015) in the three orders of Wang et al. (2023a) (Orders 1–3), and SuperNI chains 15 instruction-following tasks of Wang et al. (2022) in the two orders of Zhao et al. (2024) (Orders 7–8) and is scored by ROUGE-L. The task orders are listed in Appendix B.1.

Protocol. The backbone is a frozen T5-large (Raffel et al., 2020), and LoRA experts of rank 8 are attached to the query and value projections of the encoder, decoder, and cross-attention blocks. Following the protocol of Wang et al. (2023a), each task is trained on 1,000 examples for one epoch. The pool has $M = 16$ experts, one expert is frozen per task, and the readout uses $\rho = 0.6$ and $K = 8$. Accuracy is the average over all tasks after the last task, and forgetting is the average drop from each task's best accuracy to its final accuracy (Appendix B.2). Accuracy and forgetting are means over three seeds.

Baselines. The compared methods are isolation methods (O-LoRA (Wang et al., 2023a), InfLoRA (Liang & Li, 2024), EPI (Wang et al., 2023b)), routing methods (SAPT (Zhao et al., 2024), AM-LoRA (Liu et al., 2024)), and elastic weight consolidation (EWC) (Kirkpatrick et al., 2017) applied to a

Table 1: Accuracy (%) after the last task and forgetting $\bar{F}$ (%) averaged over task orders, with a frozen T5-large. Ok denotes Order k, SuperNI is scored by ROUGE-L, and Avg and $\bar{F}$ are means over the orders shown. Methods above the rule use no task identifier at test time; rows below the rule are reference points. Bold marks the best value in each column among the methods above the rule. Every entry is a mean over three seeds; per-order forgetting and standard deviations are in Appendix C.1.

	Long Sequence					Standard CL					SuperNI			
Method	O4	O5	O6	Avg	$\bar{F}\downarrow$	O1	O2	O3	Avg	$\bar{F}\downarrow$	O7	O8	Avg	$\bar{F}\downarrow$
RPR	**71.0**	**69.5**	**71.2**	**70.6**	**1.2**	**78.4**	**77.1**	**77.4**	**77.6**	0.1	**44.5**	**42.2**	**43.4**	0.2
InfLoRA	68.1	68.5	69.6	68.7	1.3	76.1	74.8	76.8	75.9	**0.0**	41.0	40.7	40.9	**0.0**
EWC-LoRA	61.6	61.7	64.1	62.5	1.7	75.2	76.4	76.1	75.9	0.1	39.8	38.4	39.1	**0.0**
EPI	57.2	57.9	57.9	57.7	2.4	76.0	76.0	76.7	76.2	0.1	36.8	38.6	37.7	0.1
AM-LoRA	54.2	57.3	61.2	57.6	1.6	74.3	75.1	76.8	75.4	0.1	36.8	38.3	37.6	**0.0**
O-LoRA	56.7	56.0	55.5	56.1	6.1	34.0	36.1	42.2	37.4	0.1	17.1	19.5	18.3	2.4
SAPT	38.6	41.3	48.3	42.7	2.0	69.1	69.4	68.0	68.8	0.1	25.8	24.8	25.3	**0.0**
SeqLoRA	56.0	52.5	55.9	54.8	23.3	65.8	68.2	69.4	67.8	12.2	15.2	25.7	20.5	23.9
ProgPrompt	24.6	22.7	17.5	21.6	46.5	46.3	40.8	44.6	43.9	24.6	12.4	13.1	12.8	13.5
Uniform	18.6	22.0	17.5	19.4	31.1	44.1	51.3	53.8	49.7	8.2	9.3	18.9	14.1	18.2

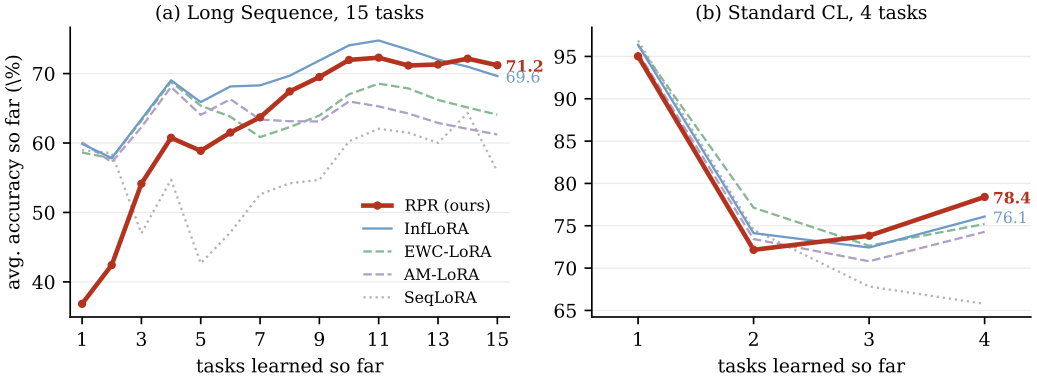


Figure 2: Average accuracy over the tasks seen so far as the stream progressess. (a) Long Sequence, Order 6. (b) Standard CL, Order 1.

single LoRA (EWC-LoRA); sequential training of a single LoRA (SeqLoRA), Progressive Prompts (ProgPrompt) (Razdaibiedina et al., 2023), and uniform random selection from the pool serve as reference points. Methods with multiple LoRA modules share the pool budget of RPR ($M = 16$, $r = 8$, $\alpha = 2$), and EWC-LoRA and SeqLoRA train a single LoRA of rank 8. All methods, including RPR, are run in one code base with the same backbone, data splits, and training schedule (Appendix B.4).

4.2 Main Results

RPR achieves the highest mean accuracy on all eight task orders of the three benchmarks (Table 1). On Long Sequence it averages 70.6% against 68.7% for InfLoRA, the second-best method, and has the lowest average forgetting, 1.2 against 1.3; its lead over the other methods widens as the stream progresses (Figure 2), and the per-task results in Appendix C.1. On Standard CL and SuperNI, RPR leads the second-best method by 1.4 and 2.5 points on average while forgetting 0.1 and 0.2 points. Against the routing methods, the lead over AM-LoRA and SAPT is 13.0 and 27.9 points on Long Sequence, 2.2 and 8.8 on Standard CL, and 5.8 and 18.1 on SuperNI.

Table 2: Component analysis on Long Sequence, three seeds per order. Each row changes one component of RPR. Avg is the mean over the three orders. Bold marks the best value in each column.

Component	Variant	O4	O5	O6	Avg	F ↓	$\lvert\hat{z}\rvert$
–	**RPR** ($\rho = 0.6$)	**71.0**	**69.5**	**71.2**	**70.6**	0.8	2.18
Readout	fixed $k = 1$	69.4	67.4	69.4	68.7	**0.7**	1.00
	fixed $k = 2$	68.9	66.2	67.8	67.6	3.1	2.00
	fixed $k = 3$	69.6	67.9	69.8	69.1	2.4	3.00
	$\rho = 0.50$	69.7	67.8	69.8	69.1	1.4	1.60
	$\rho = 0.75$	69.8	68.1	70.0	69.3	2.2	3.91
Objective	REINFORCE in place of TB	35.2	30.6	25.4	30.4	4.4	4.68
	softmax gate, top-3	50.2	49.9	51.2	50.4	12.0	3.00
	softmax gate, top-1	41.3	45.6	49.9	45.6	7.5	1.00
Consolidation	none	65.6	64.8	67.5	66.0	7.6	1.36
	no key fixing	65.2	63.7	64.7	64.5	2.3	–

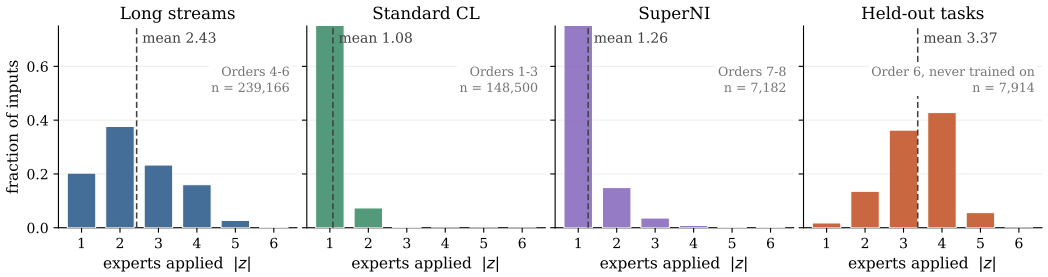


Figure 3: Experts applied per input, by stream family. Dashed line is the mean. Routing widens from 1.08 (standard) to 2.43 (long) and 3.37 on tasks never trained on.

4.3 The number of experts is read from the distribution

The readout is more accurate than every fixed number of experts and every other mass on each order of Long Sequence (Table 2). With the readout equation 7 replaced by a fixed number of experts and everything else unchanged, one, two, and three experts per input reach 68.7%, 67.6%, and 69.1% on average against 70.6%, and two and three experts raise forgetting on Order 6 to 3.1 and 2.4. Changing the mass to $\rho = 0.50$ or 0.75 shifts the mean number of experts on Order 6 to 1.60 or 3.91 and the average accuracy to 69.1% or 69.3%. At $\rho = 0.6$ the mean is 2.18 experts, and the number varies across inputs: 24% of inputs use one expert, 46% two, 20% three, and 10% four or more (Figure 3).

4.4 What the objective and consolidation contribute

The reward-proportional objective accounts for most of the accuracy of RPR, and consolidation keeps its forgetting low (Table 2). Replacing TB by REINFORCE (Williams, 1992), with the same policy network, pool, consolidation, and optimizer, lowers the average accuracy from 70.6% to 30.4% and raises forgetting from 0.8 to 4.4, and a softmax gate trained by loss minimization on the same pool reaches 50.4% with its top three experts and 45.6% with one. Without consolidation, the average accuracy falls to 66.0% and forgetting on Order 6 rises from 0.8 to 7.6, and with the keys of frozen experts left unfixed the average accuracy is 64.5%. Frozen experts are reused across the stream: tasks of the same kind place their selections on the same experts, and unrelated tasks on different ones (Figure 4).

Consolidation reserves one expert per task and is therefore applied when $M \geq T$, for which the allocation of the compared methods that add one module per task is defined. With fewer experts than tasks, RPR runs without consolidation and still reaches 66.3% on Long Sequence with $M = 5$, whereas InfLoRA and AM-LoRA fall to 31.7% and 29.2% (Appendix C.2).

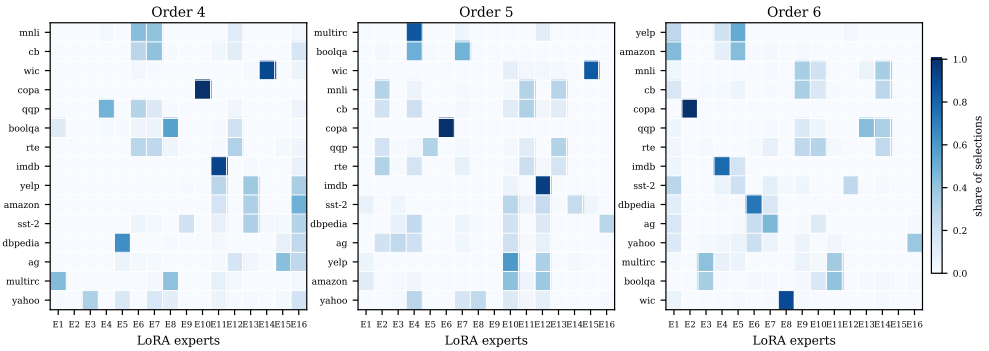


Figure 4: Share of each task's selections received by each expert at the end of each Order. Rows are tasks in stream order and columns the 16 experts. Tasks of the same kind share experts, and unrelated tasks occupy different experts.

Table 3: The last three tasks of Order 6 (MultiRC, BoolQA, WiC), evaluated after training on the first twelve. A prediction is correct if its lowercased form begins with the gold label word and valid if it begins with any label word of the task (VLR: valid-label rate). All three tasks are binary with balanced labels, so chance is 50%. Bold marks the best method trained on the stream.

Method	Acc ↑	VLR ↑	$\lvert\hat{z}\rvert$
Unadapted backbone	62.1	90.1	–
RPR	**58.2**	**87.0**	3.37
InfLoRA (Liang & Li, 2024)	1.0	2.1	–
EWC-LoRA (Kirkpatrick et al., 2017)	0.7	1.7	–
AM-LoRA (Liu et al., 2024)	0.7	1.7	–

4.5 Tasks never trained on

On tasks never trained on, RPR stays close to the unadapted backbone, whereas the compared methods lose the label space of the evaluated task (Table 3). Training on the first twelve tasks of Order 6 and evaluating the remaining three measures what a stream leaves for inputs it was never adapted to. The compared methods produce a label word of the evaluated task for at most 2.1% of inputs; their outputs are labels of previous tasks, such as `entailment` and `Sports` where `True` or `False` is required. RPR retains 94% of the unadapted backbone's accuracy (58.2 against 62.1) and 97% of its valid-label rate (87.0 against 90.1), and it applies 3.37 experts on average, more than the 2.18 it applies on the trained tasks of Order 6 (Figure 3).

5 Conclusion

This paper has proposed a method of RPR to reduce the forgetting that arises through a router's selection when experts are frozen. Instead of identifying the best combination, RPR trains the router toward a distribution over expert combinations proportional to their reward, reads the number and the composition of experts from the shape of that distribution, and freezes one expert per task. The distribution moves continuously with the loss, and on three CL benchmarks RPR achieves the highest mean accuracy on all eight task orders and the lowest average forgetting on Long Sequence, while remaining close to the unadapted backbone on tasks never trained on. The experiments use an encoder–decoder backbone, and extending RPR to decoder-only LLMs of larger scale is a direct next step, since the router acts only on the LoRA experts and leaves the backbone unchanged. RPR consolidates one expert per task under the allocation $M \geq T$ of the compared methods, and for streams in which tasks continue to arrive beyond the pool size, a criterion based on the saturation of the pool's capacity would allow consolidation to be switched off automatically, which is left for future work.

Ethics statement

All datasets are public academic benchmarks; no new data was collected, and no human subjects were involved. RPR adapts a frozen pre-trained backbone with LoRA experts and introduces no data beyond these benchmarks.

Reproducibility statement

The method is specified in Section 3 and Algorithm 1, and complete proofs of all formal statements are in Appendix A. The benchmark composition, task orders, metrics, backbone configuration, baseline adaptations, and hyperparameters are in Appendix B. Every accuracy and forgetting value is a mean over three seeds. The code is available at `https://anonymous.4open.science/r/RPR_ICLR`.

AI use statement

Large language models were used solely for grammar correction, final proofreading (checking typos), and verifying notation consistency across the manuscript. The authors verified all text, derivations, code, and results and take full responsibility for the content of the paper.

References

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in neural information processing systems*, 34:27381–27394, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. Gflownet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Wuyang Chen, Yanqi Zhou, Nan Du, Yanping Huang, James Laudon, Zhifeng Chen, and Claire Cui. Lifelong language pretraining with distribution-specialized experts. In *International Conference on Machine Learning*, pp. 5383–5395. PMLR, 2023.

Damai Dai, Li Dong, Shuming Ma, Bo Zheng, Zhifang Sui, Baobao Chang, and Furu Wei. Stablemoe: Stable routing strategy for mixture of experts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7085–7095, 2022.

Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Wei Shen, Limao Xiong, Yuhao Zhou, Xiao Wang, Zhiheng Xi, Xiaoran Fan, et al. Loramoe: Alleviating world knowledge forgetting in large language models via moe-style plugin. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1932–1945, 2024.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Chendi Ge, Xin Wang, Zeyang Zhang, Hong Chen, Jiapei Fan, Longtao Huang, Hui Xue, and Wenwu Zhu. Dynamic mixture of curriculum lora experts for continual multimodal instruction tuning. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Yongxin Guo, Zhenglin Cheng, Xiaoying Tang, Zhaopeng Tu, and Tao Lin. Dynamic mixture of experts: An auto-tuning approach for efficient transformer models. In *International Conference on Learning Representations*, volume 2025, pp. 79643–79672, 2025.

Edward Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. Amortizing intractable inference in large language models. In *International Conference on Learning Representations*, volume 2024, pp. 43053–43083, 2024.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Edward J Hu, Nikolay Malkin, Moksh Jain, Katie E Everett, Alexandros Graikos, and Yoshua Bengio. Gflownet-em for learning compositional latent variable models. In *International Conference on Machine Learning*, pp. 13528–13549. PMLR, 2023.

Quzhe Huang, Zhenwei An, Nan Zhuang, Mingxu Tao, Chen Zhang, Yang Jin, Kun Xu, Liwei Chen, Songfang Huang, and Yansong Feng. Harder task needs more experts: Dynamic routing in moe models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12883–12895, 2024.

Moksh Jain, Sharath Chandra Raparthy, Alex Hernández-Garcıa, Jarrid Rector-Brooks, Yoshua Bengio, Santiago Miret, and Emmanuel Bengio. Multi-objective gflownets. In *International conference on machine learning*, pp. 14631–14653. PMLR, 2023.

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. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Hongbo Li, Sen Lin, Lingjie Duan, Yingbin Liang, and Ness Shroff. Theory on mixture-of-experts in continual learning. In *International Conference on Learning Representations*, volume 2025, pp. 8169–8206, 2025.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23638–23647. IEEE, 2024.

Yan-Shuo Liang, Jia-Rui Chen, and Wu-Jun Li. Gated integration of low-rank adaptation for continual learning of large language models. *Advances in Neural Information Processing Systems*, 38:68659–68689, 2026.

Dianbo Liu, Moksh Jain, Bonaventure FP Dossou, Qianli Shen, Salem Lahlou, Anirudh Goyal, Nikolay Malkin, Chris Chinenye Emezue, Dinghuai Zhang, Nadhir Hassen, et al. Gflowout: Dropout with generative flow networks. In *International Conference on Machine Learning*, pp. 21715–21729. PMLR, 2023.

Jialin Liu, Jianhua Wu, Jie Liu, and Yutai Duan. Learning attentional mixture of loras for language model continual learning. *arXiv preprint arXiv:2409.19611*, 2024.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pp. 142–150, 2011.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in gflownets. *Advances in Neural Information Processing Systems*, 35: 5955–5967, 2022.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward J Hu, Katie E Everett, Dinghuai Zhang, and Yoshua Bengio. GFlownets and variational inference. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=uKiE0VIluA-.

S Mangrulkar, S Gugger, L Debut, Yo Belkada, S Paul, and B Bossan. Peft: state-of-the-art parameter-efficient fine-tuning methods (2022). *URL https://github. com/huggingface/peft*, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=UJTgQBc91_.

Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1ckMDqlg.

Daniil Tiapkin, Nikita Morozov, Alexey Naumov, and Dmitry P Vetrov. Generative flow networks as entropy-regularized rl. In *International Conference on Artificial Intelligence and Statistics*, pp. 4213–4221. PMLR, 2024.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP workshop BlackboxNLP: Analyzing and interpreting neural networks for NLP*, pp. 353–355, 2018.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Mingyang Wang, Heike Adel, Lukas Lange, Jannik Strötgen, and Hinrich Schütze. Rehearsal-free modular and compositional continual learning for language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pp. 469–480, 2024.

Renzhi Wang and Piji Li. Lemoe: Advanced mixture of experts adaptor for lifelong model editing of large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 2551–2575, 2024.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuan-Jing Huang. Orthogonal subspace learning for language model continual learning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, 2023a.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 conference on empirical methods in natural language processing*, pp. 5085–5109, 2022.

Zhicheng Wang, Yufang Liu, Tao Ji, Xiaoling Wang, Yuanbin Wu, Congcong Jiang, Ye Chao, Zhencong Han, Ling Wang, Xu Shao, et al. Rehearsal-free continual language learning via efficient parameter isolation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 10933–10946, 2023b.

Zitao Wang, Xinyi Wang, and Wei Hu. Mixture of lora experts for continual information extraction with llms. In *EMNLP (Findings)*, pp. 13324–13339, 2025a.

Ziteng Wang, Jun Zhu, and Jianfei Chen. Remoe: Fully differentiable mixture-of-experts with relu routing. In *International Conference on Learning Representations*, volume 2025, pp. 59486–59507, 2025b.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pp. 38–45, 2020.

Yuanhang Yang, Shiyi Qi, Wenchao Gu, Chaozheng Wang, Cuiyun Gao, and Zenglin Xu. Xmoe: Sparse models with fine-grained and adaptive expert selection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 11664–11674, 2024.

Zihao Zeng, Yibo Miao, Hongcheng Gao, Hao Zhang, and Zhijie Deng. Adamoe: Token-adaptive routing with null experts for mixture-of-experts language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 6223–6235, 2024.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Heng Zhao, Zilei Shao, Guy Van den Broeck, and Zhe Zeng. Probmoe: Differentiable probabilistic routing for mixture-of-experts. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=7zOtnPt85B.

Weixiang Zhao, Shilong Wang, Yulin Hu, Yanyan Zhao, Bing Qin, Xuanyu Zhang, Qing Yang, Dongliang Xu, and Wanxiang Che. Sapt: A shared attention framework for parameter-efficient continual learning of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11641–11661, 2024.

Zijia Zhao, Longteng Guo, Jie Cheng, Xuange Gao, Hua Huang, and Jing Liu. Ada-k routing: Boosting the efficiency of moe-based llms. In *International Conference on Learning Representations*, volume 2025, pp. 89619–89635, 2025.