Beyond Myopic Alignment: Lookahead Optimization for Online Class-Incremental Learning

Supplementary Material

7. Theoretical Analysis

Notation Throughout this supplementary material, we use the following notation: vectors are denoted by lowercase bold letters (e.g., $\mathbf{g} \in \mathbb{R}^d$), matrices by uppercase bold letters (e.g., $\mathbf{H} \in \mathbb{R}^{d\times d}$), and scalar functions by regular letters. The inner product between two vectors is denoted by $\langle\cdot,\cdot\rangle$. The Frobenius norm of a matrix is denoted by $\|\cdot\|_F$. The gradient of a function $\mathcal{L}$ with respect to parameters θ is denoted by $\nabla_\theta\mathcal{L}$ or $\mathbf{g}$ when the context is clear.

7.1. Proof of Proposition 3.1: Hypergradients Reduce Conflict

Proposition 7.1 (Hypergradients Reduce Conflict)**.** *The hypergradient $\mathbf{g}_j^{HD}(\theta)$ approximates an update that aligns the task-specific gradient $\nabla_\theta\mathcal{L}_j(\theta)$ with the overarching meta-gradient $\nabla_\theta\mathcal{L}_m(\theta)$.*

Proof. Let $\tilde{\theta}_j = \theta - \alpha\nabla_\theta\mathcal{L}_j(\theta)$ be the parameters after one gradient step on task j. Under the first-order (FOMAML) approximation that drops the Jacobian factor $(I-\alpha\nabla^2_\theta\mathcal{L}_j(\theta))^\top$ from the exact hypergradient, the hypergradient is approximated as:

$$\mathbf{g}_j^{\text{HD}}(\theta) \approx \nabla_\theta\mathcal{L}_{\text{m}}(\tilde{\theta}_j) \tag{9}$$

We perform a first-order Taylor expansion of $\nabla_\theta\mathcal{L}_{\text{m}}(\cdot)$ around θ:

$$\begin{aligned}\mathbf{g}_j^{\text{HD}}(\theta) &= \nabla_\theta\mathcal{L}_{\text{m}}(\theta-\alpha\nabla_\theta\mathcal{L}_j(\theta))\\ &= \nabla_\theta\mathcal{L}_{\text{m}}(\theta) + \nabla^2_\theta\mathcal{L}_{\text{m}}(\theta)\cdot(-\alpha\nabla_\theta\mathcal{L}_j(\theta))\\ &\quad + O(\alpha^2)\\ &= \nabla_\theta\mathcal{L}_{\text{m}}(\theta) - \alpha\mathbf{H}_{\text{m}}(\theta)\nabla_\theta\mathcal{L}_j(\theta)\\ &\quad + O(\alpha^2)\end{aligned} \tag{10}$$

where $\mathbf{H}_{\text{m}}(\theta) = \nabla^2_\theta\mathcal{L}_{\text{m}}(\theta)$ is the Hessian matrix of the meta-loss.

The term $\mathbf{H}_{\text{m}}(\theta)\nabla_\theta\mathcal{L}_j(\theta)$ represents a geometric transformation of the task gradient by the curvature of the meta-loss landscape. To understand this, consider the eigendecomposition of the Hessian:

$$\mathbf{H}_{\text{m}}(\theta) = \sum_{i=1}^{d}\lambda_i\mathbf{v}_i\mathbf{v}_i^\top \tag{11}$$

where λ_i are eigenvalues and $\mathbf{v}_i$ are corresponding eigenvectors.

The Hessian-vector product can be written as:

$$\mathbf{H}_{\text{m}}(\theta)\nabla_\theta\mathcal{L}_j(\theta) = \sum_{i=1}^{d}\lambda_i(\mathbf{v}_i^\top\nabla_\theta\mathcal{L}_j(\theta))\mathbf{v}_i \tag{12}$$

This shows that the Hessian scales the components of $\nabla_\theta\mathcal{L}_j(\theta)$ along each eigendirection by the corresponding eigenvalue. Directions with larger eigenvalues (higher curvature in the meta-loss) are amplified.

Now we analyze the inner product between hypergradients from the current task n and memory $\mathcal{M}$. Define:

$$\begin{aligned}\mathbf{g}_n^{\text{HD}} &= \mathbf{g}_{\text{m}} - \alpha\mathbf{H}_{\text{m}}\mathbf{g}_n + O(\alpha^2)\\ \mathbf{g}_{\mathcal{M}}^{\text{HD}} &= \mathbf{g}_{\text{m}} - \alpha\mathbf{H}_{\text{m}}\mathbf{g}_{\mathcal{M}} + O(\alpha^2)\end{aligned} \tag{13}$$

where $\mathbf{g}_{\text{m}} = \nabla_\theta\mathcal{L}_{\text{m}}(\theta)$, $\mathbf{g}_n = \nabla_\theta\mathcal{L}_n(\theta)$, and $\mathbf{g}_{\mathcal{M}} = \nabla_\theta\mathcal{L}_{\mathcal{M}}(\theta)$.

The inner product between hypergradients, up to first order in α, is:

$$\begin{aligned}&\langle\mathbf{g}_n^{\text{HD}}, \mathbf{g}_{\mathcal{M}}^{\text{HD}}\rangle\\ &= \langle\mathbf{g}_{\text{m}} - \alpha\mathbf{H}_{\text{m}}\mathbf{g}_n,\ \mathbf{g}_{\text{m}} - \alpha\mathbf{H}_{\text{m}}\mathbf{g}_{\mathcal{M}}\rangle\\ &= \|\mathbf{g}_{\text{m}}\|^2 - \alpha\langle\mathbf{g}_{\text{m}}, \mathbf{H}_{\text{m}}\mathbf{g}_{\mathcal{M}}\rangle - \alpha\langle\mathbf{H}_{\text{m}}\mathbf{g}_n, \mathbf{g}_{\text{m}}\rangle + O(\alpha^2).\end{aligned} \tag{14}$$

Since $\mathbf{H}_{\text{m}}$ is symmetric:

$$\langle\mathbf{g}_{\text{m}}, \mathbf{H}_{\text{m}}\mathbf{g}_{\mathcal{M}}\rangle = \langle\mathbf{H}_{\text{m}}\mathbf{g}_{\text{m}}, \mathbf{g}_{\mathcal{M}}\rangle \tag{15}$$

Under the assumption that the meta-loss $\mathcal{L}_{\text{m}}$ combines losses from all tasks, we have:

$$\mathbf{g}_{\text{m}} = \beta_n\mathbf{g}_n + \beta_{\mathcal{M}}\mathbf{g}_{\mathcal{M}} \tag{16}$$

where $\beta_n + \beta_{\mathcal{M}} = 1$ are the relative weights.

The key insight is that both $\mathbf{g}_n^{\text{HD}}$ and $\mathbf{g}_{\mathcal{M}}^{\text{HD}}$ contain the common term $\mathbf{g}_{\text{m}}$, which acts as an "anchor" that pulls both hypergradients toward a shared direction. This reduces their conflict compared to the original gradients $\mathbf{g}_n$ and $\mathbf{g}_{\mathcal{M}}$.

These observations suggest that, for sufficiently small α and when the meta-gradient mixes information from the current batch and the memory batch, the cosine similarity between $\mathbf{g}_n^{\text{HD}}$ and $\mathbf{g}_{\mathcal{M}}^{\text{HD}}$ tends to be larger than that between $\mathbf{g}_n$ and $\mathbf{g}_{\mathcal{M}}$. We therefore interpret hypergradient updates as reducing gradient conflict under a local first-order approximation. □

7.2. Proof of Theorem 4.1

Theorem 7.2 (Generalization Bound)**.** *Let h_θ be a classifier learned at step t from hypothesis class $\mathcal{H}$. Let $h^*_{\mathcal{M}}$ be the optimal classifier on the complete past-data distribution $P^{full}_{\mathcal{M}}$. The risk of h_θ on test distribution P_{test} is bounded by:*

$$\mathcal{R}(h_\theta, P_{test}) \leq \mathcal{R}(h_\theta, h^*_{\mathcal{M}}; P^{full}_{\mathcal{M}}) + \mathrm{disc}_{\mathcal{H}}(P^{full}_{\mathcal{M}}, P_{test}) + \lambda^* \tag{17}$$

Proof. The bound is derived under the 0-1 loss $\ell_{01}(y, y') = \mathbb{I}[y \neq y']$, which is standard in domain adaptation theory [3]. We state the result in this form for clarity; extending it to surrogate losses such as cross-entropy would require additional assumptions and is not claimed here.

For any classifier $h \in \mathcal{H}$ and distributions P, Q, define the disagreement between two classifiers:

$$d_P(h, h') = \mathbb{E}_{x\sim P}[\mathbb{I}[h(x) \neq h'(x)]] \tag{18}$$

The risk on the test distribution can be decomposed as:

$$\begin{aligned}\mathcal{R}(h_\theta, P_{\text{test}}) &= \mathbb{E}_{(x,y)\sim P_{\text{test}}}[\mathcal{L}(h_\theta(x), y)] \\ &= \mathbb{E}_{x\sim P^X_{\text{test}}}[\mathbb{E}_{y|x}[\mathcal{L}(h_\theta(x), y)]]\end{aligned} \tag{19}$$

Adding and subtracting the optimal past-data classifier $h^*_{\mathcal{M}}$:

$$\begin{aligned}\mathcal{R}(h_\theta, P_{\text{test}}) = \mathbb{E}_{(x,y)\sim P_{\text{test}}}[&\mathcal{L}(h_\theta(x), y) \\ &- \mathcal{L}(h^*_{\mathcal{M}}(x), y) + \mathcal{L}(h^*_{\mathcal{M}}(x), y)]\end{aligned} \tag{20}$$

For 0-1 loss, we have the triangle inequality:

$$\begin{aligned}&|\mathcal{L}(h_\theta(x), y) - \mathcal{L}(h^*_{\mathcal{M}}(x), y)| \\ &\leq \mathbb{I}[h_\theta(x) \neq h^*_{\mathcal{M}}(x)]\end{aligned} \tag{21}$$

This gives us:

$$\mathcal{R}(h_\theta, P_{\text{test}}) \leq d_{P_{\text{test}}}(h_\theta, h^*_{\mathcal{M}}) + \mathcal{R}(h^*_{\mathcal{M}}, P_{\text{test}}) \tag{22}$$

The key step is relating the disagreement on test distribution to that on the past-data distribution. Following [3], for any $h, h' \in \mathcal{H}$:

$$d_{P_{\text{test}}}(h, h') \leq d_{P^{\text{full}}_{\mathcal{M}}}(h, h') + \mathrm{disc}_{\mathcal{H}}(P^{\text{full}}_{\mathcal{M}}, P_{\text{test}}) \tag{23}$$

where the hypothesis-disagreement discrepancy is defined as

$$\mathrm{disc}_{\mathcal{H}}(P, Q) := \sup_{h,h'\in\mathcal{H}} |d_P(h, h') - d_Q(h, h')|, \tag{24}$$

with

$$d_P(h, h') := \mathbb{E}_{x\sim P}\big[\mathbb{I}[h(x) \neq h'(x)]\big]. \tag{25}$$

Combining the above results:

$$\begin{aligned}\mathcal{R}(h_\theta, P_{\text{test}}) \leq\ & d_{P^{\text{full}}_{\mathcal{M}}}(h_\theta, h^*_{\mathcal{M}}) \\ &+ \mathrm{disc}_{\mathcal{H}}(P^{\text{full}}_{\mathcal{M}}, P_{\text{test}}) \\ &+ \mathcal{R}(h^*_{\mathcal{M}}, P_{\text{test}})\end{aligned} \tag{26}$$

Define the disagreement risk:

$$\mathcal{R}(h_\theta, h^*_{\mathcal{M}}; P^{\text{full}}_{\mathcal{M}}) = d_{P^{\text{full}}_{\mathcal{M}}}(h_\theta, h^*_{\mathcal{M}}) \tag{27}$$

And let $\lambda^* = \mathcal{R}(h^*_{\mathcal{M}}, P_{\text{test}})$ denote the residual adaptation term. This completes the proof. □

7.3. Proof of Proposition 4.2

Proposition 7.3 (LOR Encourages Lower Disagreement Risk)**.** *The min-max formulation of LOR can be interpreted as biasing optimization toward solutions that are less sensitive to perturbations of the training mixture distribution. Under this interpretation, LOR is expected to reduce the disagreement risk with respect to $h^*_{\mathcal{M}}$ compared with standard ERM.*

Proof. At training step t, standard ERM optimizes:

$$\theta_{\text{ERM}} = \arg\min_\theta \mathcal{L}(\theta; B_t \cup B_{\mathcal{M}}) \tag{28}$$

This corresponds to a fixed mixture distribution:

$$P^{\text{ERM}}_{\text{train}} = \gamma P_{B_t} + (1-\gamma) P_{B_{\mathcal{M}}} \tag{29}$$

where γ is implicitly determined by the batch sizes.

In contrast, LOR considers a set of distributions:

$$\mathcal{P}_{\text{LOR}} = \{P_k : P_k = w_{k,n} P_{B_t} + w_{k,\mathcal{M}} P_{B_{\mathcal{M}}}, \quad k = 1, \ldots, K\} \tag{30}$$

The min-max objective of LOR can be interpreted as distributionally robust optimization (DRO):

$$\theta_{\text{LOR}} = \arg\min_\theta \max_{P\in\mathcal{P}_{\text{LOR}}} \mathbb{E}_{(x,y)\sim P}[\mathcal{L}(h_\theta(x), y)] \tag{31}$$

From [12], the solution to this problem has the property:

$$\sup_{P\in\mathcal{P}_{\text{LOR}}} \mathcal{R}(h_{\theta_{\text{LOR}}}, P) \leq \sup_{P\in\mathcal{P}_{\text{LOR}}} \mathcal{R}(h_{\theta_{\text{ERM}}}, P) \tag{32}$$

The true past-data distribution $P^{\text{full}}_{\mathcal{M}}$ can be viewed as a perturbation of the empirical distribution $P_{B_{\mathcal{M}}}$:

$$P^{\text{full}}_{\mathcal{M}} = P_{B_{\mathcal{M}}} + \Delta \tag{33}$$

where Δ represents the sampling noise and distribution shift.

The set $\mathcal{P}_{\text{LOR}}$ effectively covers a neighborhood around $P_{B_{\mathcal{M}}}$. By optimizing for the worst-case within this set, LOR finds parameters robust to such perturbations.

Let $\mathcal{N}_\epsilon(P)$ denote an ϵ-neighborhood of distribution P. For the disagreement with $h^*_{\mathcal{M}}$:

$$\begin{aligned}&\mathcal{R}(h_\theta, h^*_{\mathcal{M}}; P^{\text{full}}_{\mathcal{M}}) \\ &= \mathbb{E}_{x\sim P^{\text{full}}_{\mathcal{M}}}[\mathbb{I}[h_\theta(x) \neq h^*_{\mathcal{M}}(x)]]\end{aligned} \tag{34}$$

Since $h^*_{\mathcal{M}}$ is optimal for $P^{\text{full}}_{\mathcal{M}}$, and LOR's robustness ensures good performance across $\mathcal{P}_{\text{LOR}}$:

$$\begin{aligned}&\mathbb{E}_\Delta[\mathcal{R}(h_{\theta_{\text{LOR}}}, h^*_{\mathcal{M}}; P^{\text{full}}_{\mathcal{M}})] \\ &\leq \mathbb{E}_\Delta[\sup_{P\in\mathcal{N}_\epsilon(P_{B_{\mathcal{M}}})} \mathcal{R}(h_{\theta_{\text{LOR}}}, h^*_{\mathcal{M}}; P)]\end{aligned} \tag{35}$$

Standard ERM, optimizing for a single fixed mixture, is more susceptible to overfitting to the specific empirical distribution. Using concentration inequalities [5], with probability at least $1-\delta$:

$$\begin{aligned}&\mathcal{R}(h_{\theta_{\text{ERM}}}, h^*_{\mathcal{M}}; P^{\text{full}}_{\mathcal{M}}) \\ &\geq \mathcal{R}(h_{\theta_{\text{ERM}}}, h^*_{\mathcal{M}}; P_{B_{\mathcal{M}}}) + \Omega(\sqrt{\log(1/\delta)/|B_{\mathcal{M}}|})\end{aligned} \tag{36}$$

The robust optimization of LOR reduces this gap:

$$\begin{aligned}&\mathcal{R}(h_{\theta_{\text{LOR}}}, h^*_{\mathcal{M}}; P^{\text{full}}_{\mathcal{M}}) \\ &\leq \mathcal{R}(h_{\theta_{\text{LOR}}}, h^*_{\mathcal{M}}; P_{B_{\mathcal{M}}}) + O(\epsilon)\end{aligned} \tag{37}$$

where ϵ is controlled by the diversity of $\mathcal{P}_{\text{LOR}}$.

Taking expectations over the randomness in data sampling and algorithm execution:

$$\begin{aligned}&\mathbb{E}[\mathcal{R}(h_{\theta_{\text{LOR}}}, h^*_{\mathcal{M}}; P^{\text{full}}_{\mathcal{M}})] \\ &\leq \mathbb{E}[\mathcal{R}(h_{\theta_{\text{ERM}}}, h^*_{\mathcal{M}}; P^{\text{full}}_{\mathcal{M}})]\end{aligned} \tag{38}$$

This provides a heuristic justification for why LOR's robust optimization can reduce disagreement risk in expectation. □

7.4. Additional Remarks and Results

Choice of Preference Sampling. The Dirichlet distribution with symmetric parameters ($\alpha_s = 1.0$) provides uniform coverage over the simplex, ensuring unbiased exploration. Asymmetric choices can be used to incorporate prior knowledge about task importance.

Connection to Sharpness-Aware Minimization. LOR's preference for flatter regions connects to recent work on sharpness-aware minimization (SAM) [13]. However, LOR explicitly considers multiple future trajectories rather than worst-case perturbations in parameter space. While both SAM and LOR seek flatter loss regions, they differ in perturbation strategy. SAM perturbs within a generic ϵ-ball along the gradient direction, agnostic to task structure. LOR explores directions defined by combinations of $\mathbf{g}_n$ (current task) and $\mathbf{g}_{\mathcal{M}}$ (memory buffer). This task-aware exploration addresses the gradient conflict problem rather than generic sharpness.

Table 9. Additional Comparisons. AAA / Acc (%) on Seq-CIFAR10 ($|\mathcal{M}| = 1$k), Seq-CIFAR100 ($|\mathcal{M}| = 1$k), and Seq-TinyImageNet ($|\mathcal{M}| = 2$k) using Reduced ResNet-18.

Method	Seq-CIFAR10	Seq-CIFAR100	Seq-TinyImageNet
ER	54.91 / 42.04	17.86 / 11.89	16.27 / 10.52
Hypergradient-based			
La-MAML	46.08 / 35.89	17.37 / 10.03	15.81 / 8.02
VR-MCL	66.97 / 56.48	27.01 / 19.49	22.48 / 14.15
POCL	66.23 / 58.50	26.68 / 16.54	21.56 / 12.69
Flatness-aware			
SAM	59.00 / 48.79	18.48 / 13.39	17.21 / 11.18
ASAM	64.09 / 53.46	21.52 / 14.96	18.83 / 11.74
C-Flat	67.37 / 54.57	24.61 / 16.25	20.18 / 12.08
LOR	74.22 / 71.47	31.47 / 18.21	25.05 / 12.77

We implemented SAM, ASAM, and C-Flat, which are optimization-based methods, with grid search over $\rho \in \{0.01, 0.05, 0.1, 0.2\}$, reporting the best result for each dataset. Note that these methods are designed for offline settings, so we adapted them to OCIL. Table 9 shows that LOR achieves the best AAA on all three datasets and the best final Acc on Seq-CIFAR10, while remaining competitive with strong hypergradient and flatness-aware baselines on final Acc for Seq-CIFAR100 and Seq-TinyImageNet.

Impact of the Number of Lookahead Directions (K) The number of lookahead directions, K, is a key hyperparameter in LOR, controlling the breadth of the landscape exploration. Table 11 shows the performance and computational cost on Seq-CIFAR100 ($|\mathcal{M}| = 1$k) for $K = 2, 5, 10$. While performance improves with K, increasing K from 2 to 5 yields the most significant performance gain for a moderate increase in training time. Further increasing K to 10 results in only marginal improvements at a much higher computational cost. This indicates that $K = 5$ offers an excellent trade-off between performance and efficiency, suggesting that a moderate number of diverse exploration directions is sufficient to realize the benefits of lookahead optimization.

Task-IL and Domain-IL We evaluated LOR on Task-IL and Domain-IL. Table 10 shows LOR outperforms baselines.

Table 10. Task-IL and Domain-IL Acc(%) with $|\mathcal{M}| = 500$

Method	Task-IL		Domain-IL	
	Seq-TinyImageNet	Seq-CIFAR10	P-MNIST	R-MNIST
GEM	-	92.16	76.88	81.15
A-GEM	25.33	89.48	67.56	80.31
ER	48.64	93.61	80.60	88.91
DER++	51.91	93.88	88.21	92.77
LOR	**55.12**	**94.45**	**90.45**	**94.10**

Table 11. Impact of the number of lookahead directions (K) on Seq-CIFAR100 ($|\mathcal{M}| = 1$k).

K	AAA (%)	Acc (%)	Time (s)
2	30.50	17.50	324.5
5	**31.47**	**18.21**	365.1
10	31.60	18.30	482.2

8. Experimental details

8.1. Baseline Method Descriptions

Our proposed LOR method is compared against a variety of established online class-incremental learning techniques. These baselines encompass different strategies for mitigating catastrophic forgetting:

- **Simple SGD**: Models are trained using Stochastic Gradient Descent on the current task's data stream without any explicit mechanism to counteract forgetting. This typically serves as a lower-bound performance reference.
- **Experience Replay (ER)** [33]: A foundational rehearsal method where a small buffer of past examples is maintained. During training on new data, samples from this buffer are also replayed, and the model is trained on a combined loss from current and replayed data.
- **Elastic Weight Consolidation (EWC)** [17, 19]: A regularization-based technique that adds a penalty term to the loss function. This penalty discourages significant changes to model parameters identified as important for previously learned tasks, with importance typically quantified using the Fisher Information Matrix.
- **Gradient Episodic Memory (GEM)** [24]: This method uses episodic memory to store past task samples. It constrains the gradient update for the current task such that the loss on previous tasks (evaluated on memory samples) does not increase.
- **Averaged GEM (A-GEM)** [7]: A more computationally scalable variant of GEM. Instead of per-task constraints, A-GEM uses a single constraint based on the average loss over all samples in the memory buffer, projecting the current task's gradient if it violates this aggregate constraint.
- **Dark Experience Replay (DER & DER++)** [6]: DER and its enhancement DER++ extend ER by also storing and replaying the logits (pre-softmax outputs) of the model for past samples. This is often implemented via a knowledge distillation loss, helping to preserve the previous model's predictive behavior.
- **CLSER** [2]: A rehearsal-based continual learning method inspired by complementary learning systems. It maintains stable and plastic teacher models updated at different time scales and uses consistency regularization during replay to balance rapid adaptation to new data with retention of previously learned knowledge.
- **Continual Bias Adaptor (CBA)** [38]: CBA is designed to address distribution shifts in online CL by introducing an adaptive module. This module helps the classifier manage abrupt changes in data distribution and can be detached during inference, thus not adding computational load at test time.
- **Pareto Optimized Continual Learning (POCL)** [40]: POCL frames online continual learning as a multi-objective optimization problem. It seeks solutions that lie on the Pareto front, effectively balancing the conflicting objectives of plasticity (learning new tasks) and stability (retaining old knowledge).
- **VR-MCL** [39]: This recent meta-learning method formulates the OCIL update as a bi-level optimization problem and employs variance reduction techniques to efficiently approximate hypergradients, aiming for better gradient alignment.

For some baseline results presented in the main paper, we refer to previously published figures from [40], ensuring a fair comparison under established experimental protocols.

8.2. Dataset Details

The empirical evaluation of LOR is conducted on three widely recognized benchmark datasets for online class-incremental learning. These datasets are configured to simulate a sequential stream of tasks where new classes are introduced over time.

- **Sequential CIFAR-10 (Seq-CIFAR10)**: Derived from the CIFAR-10 dataset [20], which consists of 60,000 32×32 color images in 10 classes (50,000 training, 10,000 testing). For the sequential setting, these 10 classes are divided into a sequence of distinct tasks. As detailed in our main paper, we use a 5-task setup where each task introduces 2 new classes.
- **Sequential CIFAR-100 (Seq-CIFAR100)**: Based on the CIFAR-100 dataset [20], this dataset also contains 60,000 32×32 color images but is distributed over 100 classes. In our experiments, this is structured into 20 tasks, with each task introducing 5 new classes, presenting a more fine-grained incremental learning challenge.
- **Sequential TinyImageNet (Seq-TinyImageNet)**: This dataset is a subset of ImageNet [10], adapted by [6] for continual learning. It comprises 200 classes, with each class having 500 training images. Images are resized to

32×32. For our experiments, it is typically divided into 20 tasks, each introducing 10 new classes. Additionally, to test scalability on longer sequences, we also use a more challenging 40-task split, as detailed in Table 8.

For all datasets, standard data augmentation techniques such as random cropping and horizontal flipping are applied during the training.

8.3. Hyperparameter Settings

The LOR method introduces a few key hyperparameters that govern its behavior. The primary model architecture used in most experiments is a Reduced ResNet-18, trained with SGD. To demonstrate architectural generalizability, we also conduct experiments with a full ResNet-18 and a smaller 3-layer DNN, as shown in our ablation studies.

For LOR specific hyperparameters:

- **Number of Lookahead Directions (K)**: This parameter determines how many potential future states are explored at each training step. In our experiments, unless specified otherwise (e.g., in ablation studies), we set $K = 5$. This value was found to offer a strong balance between performance and computational efficiency, as shown in Table 11.
- **Lookahead Direction Sampling**: The lookahead directions are defined by preference vectors $\mathbf{w}_k = (w_{k,n}, w_{k,\mathcal{M}})$, which are sampled to represent diverse trade-offs. We use a Dirichlet distribution: $\text{Dir}(\alpha_s, \alpha_s)$. For our main results, we used a symmetric concentration parameter $\alpha_s = 1.0$, which encourages a uniform exploration of trade-offs. Our sensitivity analysis in Table 5 shows that LOR is robust to this choice.
- **LSE Smoothing Parameter (μ)**: As shown in our sensitivity analysis (Table 5), LOR's performance is stable across a range of values. We set $\mu = 0.5$ in our experiments.
- **Lookahead Learning Rate (η_L)**: This rate controls the step size of the exploratory lookahead updates (Eq. (4)). For simplicity and stability, we set η_L to be equal to the main learning rate η.

Other general hyperparameters: the learning rate η is set to 0.03, the batch size for new data is 32, and the replay batch from the memory buffer is also 32. The memory buffer is updated using reservoir sampling. All reported results are averaged over 5 independent runs to ensure statistical significance. The experiments were performed on a server equipped with 8 NVIDIA A100 GPUs.