STOP MARGINALIZING MY DREAMS: MODEL INVERSION VIA LAPLACE KERNEL FOR CONTINUAL LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Data-free class-incremental learning (DFCIL) relies on model inversion to synthesize replay samples from a frozen classifier, using stored feature statistics to mitigate catastrophic forgetting. Existing methods typically model these statistics with a diagonal covariance, discarding dependencies between feature dimensions, not because such correlations are unimportant, but because dense covariance modeling is computationally prohibitive. We show that this restriction can be lifted. We introduce REMIX, a structured covariance framework that parameterizes feature dependencies with a Laplace kernel over channel coordinates. As the Laplace kernel admits a linear-Gaussian state-space representation, REMIX enables exact likelihood and log-determinant computation in linear memory and near-linear time, without approximations or explicitly forming dense covariance matrices. Modeling feature correlations improves the fidelity of model inversion, yielding higher feature log-likelihoods and more coherent synthetic samples. Across CIFAR-100, Tiny-ImageNet, and CUB-200, REMIX consistently improves retention of earlier tasks across ResNet, ViT, and CLIP backbones. These results demonstrate that structured modeling of feature dependencies provides a practical and scalable alternative to the diagonal covariance assumption in data-free continual learning.

1 INTRODUCTION

Continual learning (McCloskey & Cohen, 1989; Wang et al., 2024) aims to enable models to acquire new knowledge without losing previously learned information, a phenomenon known as *catastrophic forgetting*. This challenge is particularly pronounced when privacy regulations or storage constraints prevent retaining data from previous tasks (Kaissis et al., 2020; De Lange et al., 2021; Nayak et al., 2019). Data-free class-incremental learning (DFCIL) addresses this setting by using model inversion to synthesize pseudo-samples that approximate previously observed data, enabling replay without access to the original training set.

The inversion-based approach to DFCIL builds on DeepInversion (Yin et al., 2020), which demonstrated that semantically meaningful images can be recovered by optimizing the statistics stored in the batch-normalization layers of a frozen network. ABD (Smith et al., 2021) adapted this idea to continual learning, replacing a physical replay buffer with synthesized samples of previously learned classes. R-DFCIL (Gao et al., 2022) further improved inversion by preserving relational and geometric structure in the feature space, while PMI (Tong et al., 2025) introduced layer-wise inversion, optimizing pseudo-samples sequentially across layers to accelerate their convergence toward the original feature distributions. Together, these methods established model inversion as an effective mechanism for data-free rehearsal.

Despite these advances, existing inversion methods typically represent feature statistics using a Gaussian model with diagonal covariance. This assumption treats feature dimensions as independent and therefore discards correlations that characterize the joint distribution of learned representations. Such dependencies can arise naturally in deep feature maps, where spatially and semantically related features interact across locations and channels. Ignoring these correlations limits the ability of model inversion to reproduce the joint feature distribution, potentially resulting in less faithful synthetic samples and weaker knowledge retention.

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.

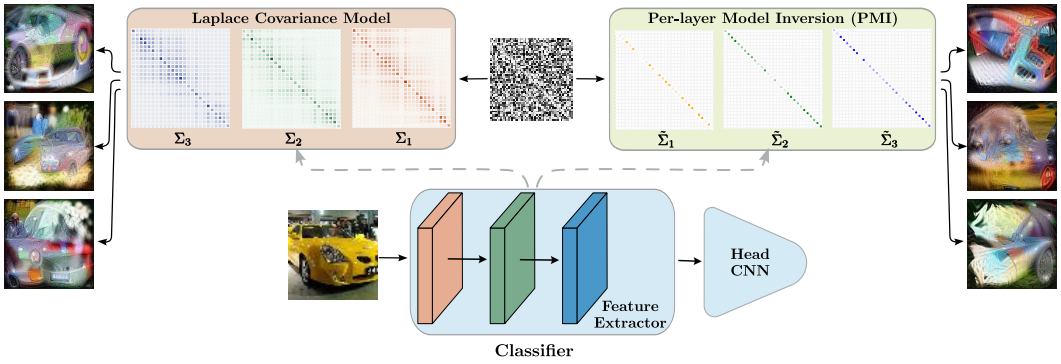

Figure 1: Overview of the shared feature extraction pipeline in DFCIL used by the proposed *Laplace Covariance Model (LCM)* and the *Per-layer Model Inversion (PMI)*. While PMI relies on a simplified diagonal approximation of covariance that assumes independence and ignores cross-feature correlations, our LCM captures full covariance structure, modeling spatial dependencies.

The reason for this simplification is computational rather than conceptual. A dense covariance requires quadratic memory in the feature dimensionality, while likelihood evaluation involves matrix inversion and log-determinant computation with cubic computational cost. These requirements quickly become impractical for the high-dimensional feature representations used by modern neural networks. Consequently, existing inversion methods trade expressive covariance modeling for tractability, restricting model inversion to marginal statistics while leaving dependencies between feature dimensions unmodeled.

In this work, we show that this trade-off can be substantially reduced. We introduce REMIX (**RE**constructing **M**ultivariate **I**nteractions via e**X**ponential kernel), a structured covariance parameterization that captures feature dependencies while remaining computationally tractable. Our key idea is to parameterize the covariance using a Laplace (exponential) kernel over learned channel coordinates. On the real line, the Laplace kernel is the *unique* stationary kernel whose associated Gaussian process is Markov (Doob, 1942; Rasmussen & Williams, 2005), and this Markov property admits an equivalent linear-Gaussian state-space representation (Hartikainen & Särkkä, 2010; Sarkka & Hartikainen, 2012). That structure enables exact computation of likelihoods and log-determinants without explicitly constructing or inverting a dense covariance matrix, resulting in linear memory complexity and near-linear computational cost. REMIX thus makes structured covariance modeling tractable for the high-dimensional representations encountered in model inversion. An overview of our approach is shown in Fig. 1.

We integrate REMIX into the model inversion pipeline and evaluate its impact on feature modeling and continual learning performance. We show that modeling feature dependencies yields higher feature log-likelihoods and more coherent synthetic samples than the diagonal approximation. Across CIFAR-100, Tiny-ImageNet, and CUB-200, REMIX consistently improves retention of previously learned knowledge across ResNet, ViT, and CLIP backbones. We further show that REMIX is complementary to existing continual learning approaches: when applied to parameter-efficient continual learning, it improves performance without modifying the underlying adaptation mechanism. These results demonstrate that modeling feature dependencies provides an effective and scalable alternative to the diagonal covariance assumption in DFCIL.

The main contributions of this paper are summarized as follows:

- We show that the diagonal covariance assumption in existing DFCIL methods is a computational rather than conceptual constraint, and that modeling feature dependencies beyond the diagonal yields higher feature log-likelihoods, more coherent synthetic samples, and improved knowledge retention.
- We introduce a structured covariance model based on the Laplace kernel that captures feature dependencies while avoiding the prohibitive memory and computational costs of dense covariance modeling. Its state-space formulation enables exact likelihood and log-determinant computation with linear memory and near-linear computational complexity.

- We demonstrate the effectiveness of REMIX across standard DFCIL benchmarks and diverse neural architectures, showing more coherent synthetic samples and improved retention of previously learned knowledge. REMIX is also compatible with parameter-efficient continual learning, improving performance without modifying the underlying adaptation mechanism.

2 Related Work

Class-incremental and Data-free Continual Learning. Class-incremental learning (CIL) considers learning a sequence of tasks with disjoint classes while maintaining performance on previously learned classes, with catastrophic forgetting being a central challenge. Conventional approaches typically use rehearsal, storing a subset of past examples in a memory buffer. iCaRL (Rebuffi et al., 2017), for example, combines exemplar replay with knowledge distillation and a nearest-mean classifier, while subsequent methods introduced structural regularizers (Douillard et al., 2020; Castro et al., 2018; Hou et al., 2019; Yu et al., 2020) and improved memory selection and utilization (Bang et al., 2021; Prabhu et al., 2020; Liu et al., 2021).

DFCIL removes the need to retain past examples. Early approaches such as LwF (Li & Hoiem, 2017) relied on knowledge distillation, while subsequent methods introduced generative replay (Shin et al., 2017; Cong et al., 2020; Kemker & Kanan, 2018; Wu et al., 2018; Ye & Bors, 2020). Model inversion instead synthesizes inputs that reproduce the internal statistics of a frozen classifier. DeepInversion (Yin et al., 2020) pioneered this approach by synthesizing images whose intermediate activations match the running means and variances stored in the network's BatchNorm layers. ABD (Smith et al., 2021) adapted model inversion to class-incremental learning, while R-DFCIL (Gao et al., 2022) incorporated relation-guided distillation, and PMI (Tong et al., 2025) extended this approach by matching feature statistics separately at each layer, enabling sequential inversion that accelerates convergence of the synthesized samples toward the original feature distributions. Other methods improve synthesis quality, diversity, and efficiency (Fang et al., 2021; 2022), while recent work has studied inversion of large pretrained models (Kazemi et al., 2024).

Exemplar-free Learning and Pretrained Models. Several exemplar-free methods retain information about past tasks by storing statistics of learned features, typically at the penultimate layer. FeCAM (Goswami et al., 2023) freezes the backbone after the first task and uses a training-free Bayes classifier that assigns samples by Mahalanobis distance under class-specific covariances. FeTrIL (Petit et al., 2023) also freezes the backbone, but synthesizes pseudo-features for past classes by translating current-task features toward stored prototypes and retrains a linear classifier on the resulting mixture. AdaGauss (Rypeść et al., 2024), in contrast, does not freeze the backbone but adapts its stored Gaussian distributions to the evolving feature space.

Another line of work adapts pretrained models to continual learning using parameter-efficient mechanisms. Prompt-based methods include L2P (Wang et al., 2021), DualPrompt (Wang et al., 2022), and CODA-Prompt (Smith et al., 2023), while adapter- and low-rank approaches include InfLoRA (Liang & Li, 2024) and MoE-Adapter (Yu et al., 2024). By restricting adaptation to a small set of parameters, these methods leverage representations acquired during large-scale pretraining while reducing the cost of continual adaptation.

Structured Covariance Modeling. Covariance estimation under computational and memory constraints is a longstanding problem. Common approaches impose structure through low-rank factorizations, such as probabilistic PCA and factor analysis (Tipping & Bishop, 1999; Rubin & Thayer, 1982), shrinkage estimators (Ledoit & Wolf, 2004), sparse precision matrices (Friedman et al., 2007), or algebraic structure such as Toeplitz and Kronecker forms (Gilboa et al., 2015). Low-rank and shrinkage approaches primarily address statistical regularization when the number of samples is small relative to the feature dimension, while algebraic and sparse structures exploit assumptions about the underlying index set. These assumptions are not directly available for neural feature channels, which generally have no predefined ordering or grid structure.

A related line of work exploits the connection between stationary Gaussian processes and state-space models. In particular, the Laplace kernel corresponds to a Markov Gaussian process and admits a linear-Gaussian state-space representation (Hartikainen & Särkkä, 2010; Sarkka & Hartikainen, 2012). This connection is well established when the process index has a given ordering, such as

time or spatial location. For neural feature maps, however, such an ordering is not naturally defined, motivating approaches that learn latent coordinates for the feature dimensions.

3 METHOD

REMIX is a model inversion framework for DFCIL that captures dependencies between feature dimensions using the *Laplace Covariance Model* (LCM), while remaining tractable for high-dimensional features. Existing methods discard these correlations by using diagonal covariance models.

Our method involves three optimization problems. *Inversion* synthesizes samples from fixed stored statistics, following prior DFCIL works. *Fitting* estimates LCM parameters from current-task features, while *merging* combines past and current LCMs without accessing past data. The latter two constitute our contributions.

Background and Problem Setup. We consider the standard DFCIL setting with a sequence of tasks $\{\mathcal{D}_t\}_{t=1}^{T}$, where each $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_t}$ contains a disjoint set of classes. During training on task t, only $\mathcal{D}_t$ is accessible, and past data cannot be stored. To mitigate forgetting, prior methods use *model inversion* to synthesize pseudo-samples from a frozen classifier by matching the distributions of intermediate activations between real and synthetic inputs, conditioned on a target label y. Let $\mathbf{h}_l$ denote the activation at layer l, with $\mathbf{h}_0 = \mathbf{x}$. Since the frozen network deterministically maps inputs to activations, a distribution over synthetic inputs induces a corresponding distribution at every layer. The ideal objective is therefore:

$$\min_{p_s(\mathbf{x})} D_{\mathrm{KL}}\big(p_s(\mathbf{h}_L, \ldots, \mathbf{h}_0 \mid y) \,\|\, p_r(\mathbf{h}_L, \ldots, \mathbf{h}_0 \mid y)\big), \tag{1}$$

where p_s, p_r are the synthetic and real distributions, respectively.

This objective is intractable in practice, so existing methods approximate it in two steps (Tong et al., 2025). First, the joint distribution is decomposed into layer-wise terms, with each layer approximated by a Gaussian $\mathcal{N}(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)$ estimated from real features. Second, the entropy term is omitted, leaving the layer-wise cross-entropy objective

$$\mathcal{L}_{\mathrm{stat}}^{(l)}(\boldsymbol{\psi}) = \mathbb{E}_{\mathbf{h}_l \sim p_s^{(l)}(\cdot\,;\boldsymbol{\psi})}\big[-\log \mathcal{N}(\mathbf{h}_l; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)\big], \tag{2}$$

which is minimized with respect to $\boldsymbol{\psi}$ during *inversion*, while $(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)$ are fixed statistics of the real features and $p_s^{(l)}$ denotes the distribution induced at layer l. Minimizing the cross-entropy term alone is degenerate, as it encourages $p_s^{(l)}$ to collapse to $\boldsymbol{\mu}_l$. Prior methods therefore introduce class-conditional sampling, contrastive feature selection (CFS), and standard DFCIL regularizers to preserve diversity. The complete inversion procedure is described in Appendix E.

The remaining challenge is representing $\boldsymbol{\Sigma}_l$ efficiently. A dense covariance requires storing all pairwise feature interactions, leading to quadratic memory and cubic-time matrix operations. Existing methods therefore typically use diagonal covariance models. We instead aim to capture feature correlations without explicitly storing or manipulating a dense covariance matrix.

Laplace Covariance Model and Fitting. On the real line, the exponential (Laplace) covariance is the *unique* stationary covariance whose Gaussian process is Markov (Rasmussen & Williams, 2005; Doob, 1942). This Markov property enables a linear-Gaussian state-space representation and, consequently, exact likelihood evaluation through linear-time recursions (Hartikainen & Särkkä, 2010; Sarkka & Hartikainen, 2012).

We therefore assign each feature dimension i a learnable coordinate $a_i \in \mathbb{R}$, defining an ordering along which dependencies follow this Markov structure, and use $K_{ij}(\mathbf{a}) = \exp(-|a_i - a_j|)$. Adding a learnable scale $\mathbf{w} \in \mathbb{R}^D$, where D is the feature dimensionality, and a strictly positive diagonal $\mathbf{d} = \mathrm{softplus}(\mathbf{u}) + \varepsilon\mathbf{1}$ for independent residual variance gives, with the layer index omitted,

$$\boldsymbol{\Sigma} = \mathrm{diag}(\mathbf{d}) + \mathrm{diag}(\mathbf{w}) K(\mathbf{a}) \,\mathrm{diag}(\mathbf{w}). \tag{3}$$

Since $K(\mathbf{a})$ is positive semidefinite and $\mathbf{d} \geq \varepsilon\mathbf{1}$, the resulting covariance satisfies $\boldsymbol{\Sigma} \succeq \varepsilon I \succ 0$ for any fitted parameter values. Figure 2 illustrates how the $\mathcal{O}(D)$ parameters define the resulting covariance structure.

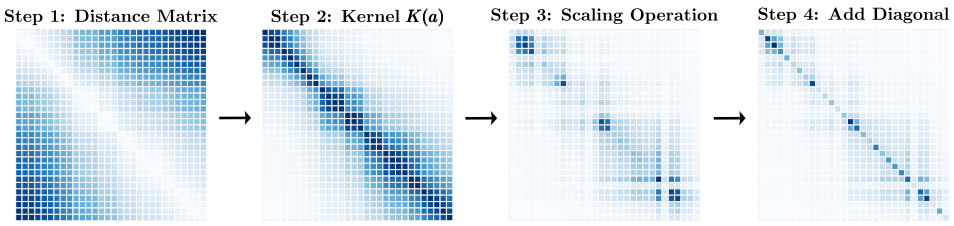

Figure 2: Construction of the REMIX covariance $\mathbf{\Sigma} \in \mathbb{R}^{D\times D}$ from $\mathcal{O}(D)$ parameters. **(1)** Each channel receives a learnable coordinate a_i, defining pairwise distances $|a_i - a_j|$. **(2)** Distances become correlations through the Laplace kernel $K_{ij} = \exp(-|a_i - a_j|)$, whose inverse on sorted coordinates is tridiagonal. **(3)** Rescaling by $\mathrm{diag}(\mathbf{w})$ modulates channel-wise variance. **(4)** Adding $\mathrm{diag}(\mathbf{d})$ models independent residual noise and yields $\mathbf{\Sigma}$.

We next fit these parameters to the feature statistics observed on the current task. Given centered activations $\mathbf{z}_i = \mathbf{h}_i - \boldsymbol{\mu}$ and the empirical covariance $\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{z}_i\mathbf{z}_i^\top$, we minimize the Frobenius distance $\mathcal{L}_{\mathrm{F}} = \left\|\mathbf{\Sigma} - \hat{\mathbf{\Sigma}}\right\|_{\mathrm{F}}^2$. Although the objective is defined through $D \times D$ matrices, its expansion reduces to element-wise operations and quadratic forms $\mathbf{x}^\top K(\mathbf{a})\mathbf{x}$, neither of which requires the covariance or kernel matrices to be constructed. Appendix A derives:

$$\mathcal{L}_{\mathrm{F}} = \underbrace{\|\mathbf{d}\|_2^2 + 2\mathbf{d}^\top\mathbf{w}^2 + (\mathbf{w}^2)^\top K(2\mathbf{a})(\mathbf{w}^2)}_{\text{model term}} - \underbrace{\frac{2}{N}\sum_{i=1}^{N}\left(\mathbf{d}^\top\mathbf{z}_i^2 + (\mathbf{w}\odot\mathbf{z}_i)^\top K(\mathbf{a})(\mathbf{w}\odot\mathbf{z}_i)\right)}_{\text{data term}} + \text{const}, \tag{4}$$

where powers are element-wise and $K_{ij}(\mathbf{a})^2 = K_{ij}(2\mathbf{a})$. After sorting the coordinates, evaluation requires $\mathcal{O}(D\log D)$ time and $\mathcal{O}(D)$ memory (Appendix B).

The same parameterization also yields *exact* likelihoods. Eq. 3 is the marginal covariance of a scaled AR(1) latent process ordered by $\mathbf{a}$, observed under independent per-dimension noise. The learned ordering makes the joint latent–observation precision matrix sparse: its only non-zero off-diagonal entries link neighbouring latent states and each latent state to its own observation. This structure allows filtering recursions to evaluate both $\mathbf{x}^\top\mathbf{\Sigma}^{-1}\mathbf{x}$ and $\log\det\mathbf{\Sigma}$ in $\mathcal{O}(D)$ time ($\mathcal{O}(D\log D)$ including the sort) and $\mathcal{O}(D)$ memory, without forming $\mathbf{\Sigma}$ (Appendix D).

Storing and Aggregating Statistics. For model inversion, we retain a compact representation of the feature distributions observed on previous tasks. At each layer, REMIX stores the LCM parameters $(\mathbf{d}, \mathbf{w}, \mathbf{a})$ together with the marginal statistics used by prior methods. These parameters are fitted independently for each task and describe the shared feature distribution rather than separate class-conditional distributions. For architectures with BatchNorm, its running statistics provide the marginal means and variances exactly, so the LCM parameters need only model the correlation structure (Appendix G); for architectures without BatchNorm, such as ViTs (Dosovitskiy et al., 2021), the required layer-wise statistics are recorded directly. The inversion process proceeds sequentially across layers (Tong et al., 2025; Kazemi et al., 2024).

Rather than retaining a separate covariance for every task, we maintain a single LCM per layer. Let $\boldsymbol{\vartheta}_{1:t}$ denote the LCM parameters representing the aggregate covariance of tasks $1,\ldots,t$, and let $\boldsymbol{\vartheta}_{t+1}$ denote the parameters fitted to the current task. We then update the aggregate by weighting the previous and current covariances according to the number of classes they represent:

$$\mathbf{T}_{t+1} = \alpha_t\mathbf{\Sigma}(\boldsymbol{\vartheta}_{1:t}) + \beta_t\mathbf{\Sigma}(\boldsymbol{\vartheta}_{t+1}), \qquad \alpha_t = \frac{|\mathcal{Y}_{1:t}|}{|\mathcal{Y}_{1:t+1}|}, \quad \beta_t = 1 - \alpha_t. \tag{5}$$

Because a convex combination of two LCM covariances is not necessarily itself an LCM covariance, we project the aggregate back onto the LCM family:

$$\boldsymbol{\vartheta}_{1:t+1} = \arg\min_{\boldsymbol{\vartheta}} \|\mathbf{\Sigma}(\boldsymbol{\vartheta}) - \mathbf{T}_{t+1}\|_{\mathrm{F}}^2. \tag{6}$$

Importantly, $\mathbf{T}_{t+1}$ never needs to be explicitly constructed: the projection depends only on the fitted LCM parameters through the closed-form Frobenius inner products derived in Appendix G. The

merge requires $\mathcal{O}(D \log D)$ time and $\mathcal{O}(D)$ memory, without retaining any task data. The merged LCM preserves the marginal variances of $\mathbf{T}_{t+1}$.

The aggregated LCM provides $(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)$ for Eq. 2, so the inversion objective evaluates the full Gaussian likelihood rather than treating feature dimensions independently. The quadratic form and log-determinant are evaluated using the LCM recursions without forming or inverting the covariance matrix. The remaining inversion terms follow standard DFCIL practice (Tong et al., 2025) and are described in Appendix E; within that objective, $\mathcal{L}_{\text{stat}}^{(l)}$ is weighted by a scalar α_{stat}, the only hyperparameter we retune relative to PMI. The resulting pseudo-samples are then used in the standard two-stage DFCIL training protocol, detailed in Appendix F, with pseudocode in Appendix J.

4 Experiments

In this section, we describe the experimental setup used to evaluate REMIX, including the considered benchmarks and datasets. We then present quantitative results demonstrating improvements over SOTA methods, followed by ablation studies.

Experimental Setup. Experiments are conducted on CIFAR-100 (Krizhevsky, 2009), Tiny-ImageNet (Wu et al., 2017), and CUB-200 (Welinder et al., 2010), following the PMI (Tong et al., 2025) and R-DFCIL (Gao et al., 2022) protocols with identical task splits and evaluation settings. CIFAR-100 and Tiny-ImageNet are evaluated with a ResNet-32 backbone, while CIFAR-100 and CUB-200 are additionally considered in a CLIP-based setting with a frozen image encoder (Radford et al., 2021). All experiments are performed in the data-free regime, where no past data are retained and pseudo-samples are generated via model inversion. We compare REMIX against prior DFCIL methods as well as selected approaches that rely on real data buffers. Additional details on task configurations and training setups are provided in Appendix I.

To ensure a fair comparison, we adopt all hyperparameters from PMI, except for α_{stat}, which we retune because the statistical matching term differs: PMI fits marginal feature statistics, whereas REMIX uses the LCM likelihood. See Appendix I for the resulting hyperparameter settings. We evaluate REMIX using two standard continual learning metrics: *average incremental accuracy* and *last task accuracy* (see Appendix H for formal definitions).

Results on ResNet-32 Backbone. Table 1 reports the average incremental accuracy on CIFAR-100 and Tiny-ImageNet using a ResNet-32 backbone under different task splits. REMIX achieves the highest accuracy across all evaluated settings, outperforming the strongest baseline, PMI, by 0.51–1.48 percentage points on CIFAR-100 and 0.74–1.44 percentage points on Tiny-ImageNet.

On CIFAR-100, the largest improvement of **+1.48** p.p. is obtained in the 10-task setting, while on Tiny-ImageNet, REMIX achieves its largest gain of **+1.44** p.p. in the 20-task setting. These results indicate that modeling feature correlations during inversion can improve the quality of generated pseudo-samples and, consequently, the retention of previously learned knowledge over longer task sequences.

Table 1: Average incremental accuracy (%) on CIFAR-100 and Tiny-ImageNet using a ResNet-32 backbone. Results are averaged over five runs. Values in parentheses indicate the improvement of REMIX over the best baseline.

Method	Model Inversion	CIFAR-100			Tiny-ImageNet		
		5 task	10 task	20 task	5 task	10 task	20 task
Upper bound	✗	70.59	70.59	70.59	55.25	55.25	55.25
DeepInversion	✓	20.48	11.26	5.63	-	-	-
ABD	✓	48.84	36.75	24.40	30.83	23.17	14.61
R-DFCIL	✓	49.87	41.80	31.54	35.33	29.05	24.85
PMI w/o CFS	✓	52.05	43.23	32.23	37.65	32.09	25.51
PMI	✓	52.38	43.90	32.60	37.90	32.43	25.67
REMIX	✓	**52.94** (+0.56)	**45.38** (+1.48)	**33.11** (+0.51)	**38.64** (+0.74)	**33.39** (+0.96)	**27.11** (+1.44)

Table 2: Average incremental accuracy (%) on CIFAR-100 and CUB-200 using a ViT backbone. Values in parentheses indicate the improvement of REMIX over the best baseline.

Method	No Real Image Buffer	CIFAR-100		CUB-200	
		Avg.	Last	Avg.	Last
iCaRL	✗	82.20	64.15	82.39	75.11
MEMO	✗	85.31	75.24	77.72	65.95
PROOF	✗	86.95	79.32	85.13	79.69
CLAP4CLIP	✗	86.13	78.21	86.93	81.64
ZS-CLIP	✓	76.71	66.25	66.81	53.52
AttriCLIP	✓	79.31	68.45	65.26	52.12
VPT	✓	83.98	74.34	68.22	55.51
CODA-P	✓	83.64	75.80	73.05	61.83
MoE-Adapter	✓	87.29	79.40	73.34	61.07
PMI + VPT	✓	85.45	76.91	70.14	57.04
PMI + CODA-P	✓	84.78	76.23	74.37	64.63
PMI + MoE-Adapter	✓	88.35	81.06	**78.98**	67.26
REMIX + MoE-Adapter	✓	**88.47** (+0.12)	**81.37** (+0.31)	78.96 (-0.02)	**67.69** (+0.43)

Importantly, our implementation builds directly on the official PMI codebase. We retain the original training protocol and hyperparameters, tuning only α_{stat} to account for the different statistical matching objective. This isolates the contribution of the proposed covariance modeling while keeping the underlying implementation consistent with PMI.

Results on ViT Backbone. Table 2 presents continual learning performance using a ViT-based CLIP backbone on CIFAR-100 and CUB-200. We report both average and last task accuracy to capture overall performance and final retention.

Overall, REMIX combined with MoE-Adapter achieves the best results among data-free approaches and remains competitive with methods that rely on real image buffers. Compared to the strongest baseline (PMI + MoE-Adapter), our method consistently improves performance across both datasets and metrics. On CIFAR-100, REMIX achieves gains of **+0.12** p.p. in average accuracy and **+0.31** p.p. in last-task accuracy. On the more challenging fine-grained CUB-200 benchmark, REMIX remains competitive in average accuracy (-0.02) while achieving a clear improvement of **+0.43** in final accuracy, indicating stronger retention of previously learned classes.

These results demonstrate that modeling structured feature dependencies during inversion also benefits the CLIP setting. The consistent gains across both datasets indicate improved alignment between synthetic and real feature distributions, resulting in better knowledge retention. Importantly, these improvements are achieved without modifying the underlying CLIP adaptation framework, suggesting that they stem directly from the proposed covariance modeling.

Qualitative Analysis of Generated Samples. Fig. 3 illustrates the impact of modeling structured feature dependencies during model inversion across both convolutional (ResNet-34) and transformer-based (ViT-B/16) architectures. Methods relying on diagonal covariance assumptions treat feature dimensions independently, often producing samples with artifacts, blurred structures, and inconsistent semantics. In contrast, REMIX captures correlations within feature representations while remaining computationally efficient, even for high-dimensional feature spaces. As a result, the generated samples exhibit clearer object structure, more coherent textures, and stronger semantic consistency across architectures.

Effect of Covariance Modeling. We further evaluate whether modeling feature dependencies improves the fit to the underlying feature distribution. We compare log-likelihoods under diagonal covariance and the proposed LCM for ResNet-34. We extract intermediate representations from `layer1`–`layer4` for four ImageNet classes (dog, car, koala, greenhouse), flattening each feature map into a high-dimensional vector ($D = C \times H \times W$, up to $\sim 2 \cdot 10^5$).

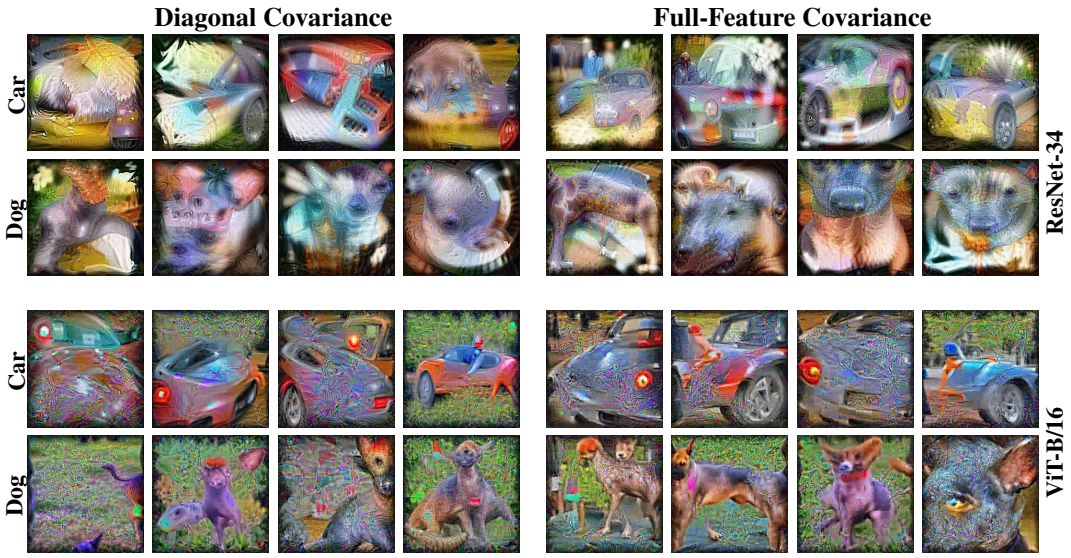

Figure 3: Qualitative comparison of synthetic samples generated using diagonal covariance modeling and the proposed full-feature covariance modeling. Results are shown for both ResNet-34 and ViT-B/16 backbones. Modeling structured feature dependencies produces samples with improved semantic coherence and more consistent object structure across architectures.

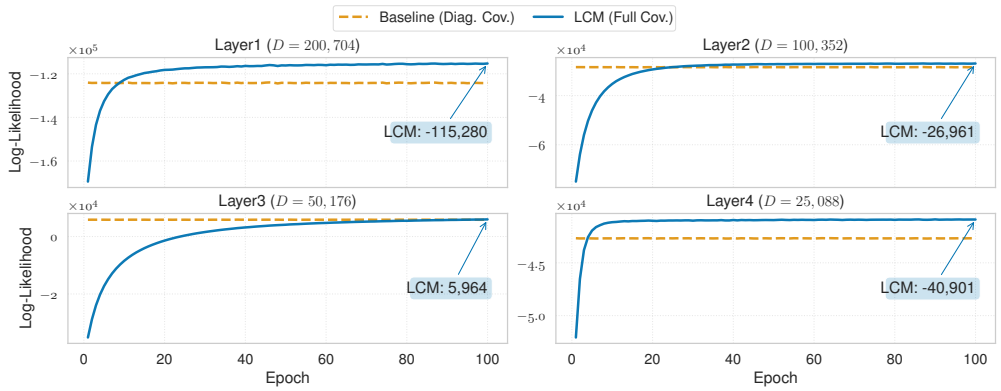

Figure 4: Log-likelihood comparison between diagonal and full-feature covariance models across ResNet-34 layers (`layer1–layer4`) on ImageNet classes (dog, car, koala, greenhouse). The proposed LCM (solid line) consistently outperforms the diagonal baseline (dashed line) across all layers.

As shown in Fig. 4, LCM consistently achieves higher log-likelihood than the diagonal baseline across all layers. The improvement persists in this high-dimensional regime, indicating that modeling dependencies across the feature map provides a closer fit to the observed feature distribution.

Appendix K reports results for ViT architectures, examining when full-covariance modeling is beneficial and when diagonal assumptions remain sufficient across transformer layers, together with a memory analysis. For the largest feature representation, a dense covariance requires 199.30 GB, whereas the LCM requires only 5.74 MB.

Sensitivity to α_{stat}. We analyze the sensitivity of REMIX to the statistical matching weight α_{stat} by sweeping values in the range $[10^{-4}, 5 \cdot 10^{-2}]$. The study is conducted on the CUB-200 dataset using a ViT backbone with the MoE-Adapter framework.

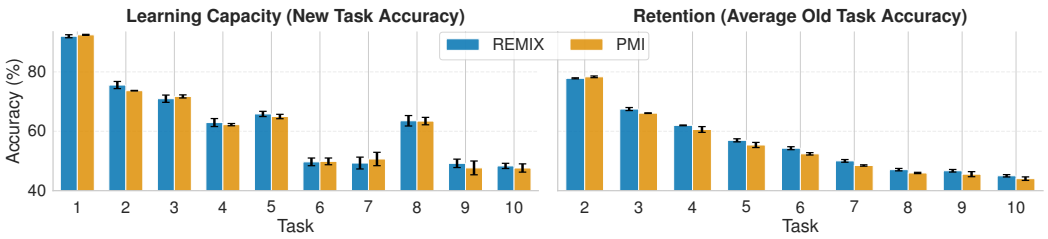

Figure 6: Task-wise performance comparison on CIFAR-100 (ResNet-32). Bars show mean accuracy over five runs, with error bars indicating standard deviation. REMIX consistently outperforms PMI in retaining previously learned knowledge while maintaining similar performance trends across tasks.

The average incremental accuracy (Fig. 5) remains highly stable across all configurations, varying by less than 0.5 p.p., indicating that the overall learning dynamics are robust to the choice of α_{stat}. The final accuracy $\mathcal{A}_{\text{last}}$ is also relatively stable across the tested values, with the highest observed value at $\alpha_{\text{stat}} = 0.01$ (67.69%).

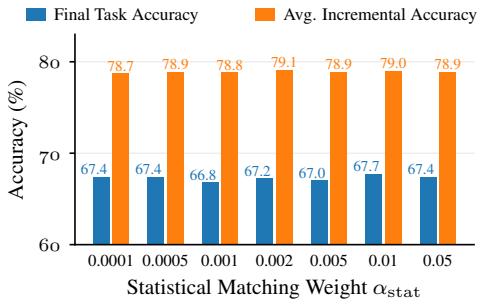

Figure 5: Sensitivity to α_{stat} on CUB-200 (ViT + MoE-Adapter).

Learning Capacity and Retention. We analyze the trade-off between plasticity and stability by comparing REMIX and PMI on CIFAR-100 using a ResNet-32 backbone. Fig. 6 reports task-wise performance, where the left plot shows accuracy on the current task (learning capacity), while the right plot shows the average accuracy on previously learned tasks (retention), both averaged over five runs. We observe that REMIX maintains performance comparable to PMI on new tasks, indicating that modeling feature dependencies does not impair learning capacity. At the same time, REMIX consistently achieves higher accuracy on previously learned tasks, demonstrating improved retention. These results indicate that incorporating structured covariance effectively mitigates forgetting while preserving the plasticity required for continual learning. Analogous results on Tiny-ImageNet are provided in Appendix K.

5 Conclusions

We introduce REMIX, a model inversion framework for DFCIL that captures dependencies between feature dimensions through a compact Laplace-based parameterization. REMIX replaces the diagonal statistical model used in prior inversion methods with full-covariance modeling that remains tractable in high-dimensional feature spaces. This improves the reconstruction of intermediate representations, producing pseudo-samples that better preserve previously learned knowledge and improve retention in the DFCIL setting.

Limitations. The proposed parameterization may be less expressive than fully flexible generative models, as it imposes a specific form on feature dependencies. In addition, REMIX inherits limitations of the PMI inversion framework, including imperfect alignment between synthesized and real feature distributions. Exploring more expressive generative models and improved inversion objectives is an important direction for future work.

AI use statement

In this work, we used generative AI tools for language editing, improving the clarity and organization of the manuscript, and assisting with the discussion and refinement of research ideas. We have not used generative AI tools to generate experimental results, fabricate data, or produce the final scientific claims of this work. Other required disclosure categories are not applicable to this work.

Additionally, we used generative AI tools for assistance with code development and debugging. All AI-assisted text, ideas, and code were reviewed by the authors; AI-assisted code was tested and verified before use, and all scientific claims and experimental results were independently checked by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility statement

All information required to reproduce our results is provided in the paper and its appendices. The derivation of the Frobenius objective in Eq. 4 is given in Appendix A, and its quasi-linear-time evaluation in Appendix B. The complete inversion objective is specified in Appendix E, while Appendix D details the state-space representation and filtering recursions used for exact likelihood and log-determinant computation. The continual merging procedure, including its assumptions and proofs, is detailed in Appendix G, while the complete two-stage training protocol and pseudocode are provided in Appendices F and J, respectively.

Appendix I provides the benchmarks and task splits, which follow the published PMI and R-DFCIL protocols without modification, together with optimizer settings, the α_{stat} search grid and selected values, library versions, and hardware details. Evaluation metrics are defined in Appendix H, and results over five random seeds, including standard deviations, are reported in Appendix K. Our full implementation, configuration files, and scripts for reproducing all tables and figures are provided as anonymized supplementary material.

References

Jihwan Bang, Heesu Kim, YoungJoon Yoo, Jung-Woo Ha, and Jonghyun Choi. Rainbow memory: Continual learning with a memory of diverse samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8218–8227, 2021.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 233–248, 2018.

Yulai Cong, Miaoyun Zhao, Jianqiao Li, Sijia Wang, and Lawrence Carin. Gan memory with no forgetting. *Advances in neural information processing systems*, 33:16481–16494, 2020.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

Joseph L. Doob. The brownian movement and stochastic equations. *Annals of Mathematics*, 43: 351–369, 1942. URL https://api.semanticscholar.org/CorpusID:120984585.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European conference on computer vision*, pp. 86–102. Springer, 2020.

Gongfan Fang, Jie Song, Xinchao Wang, Chengchao Shen, Xingen Wang, and Mingli Song. Contrastive model inversion for data-free knowledge distillation. *arXiv preprint arXiv:2105.08584*, 2021.

Gongfan Fang, Kanya Mo, Xinchao Wang, Jie Song, Shitao Bei, Haofei Zhang, and Mingli Song. Up to 100x faster data-free knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6597–6604, 2022.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the lasso, 2007. URL https://arxiv.org/abs/0708.3517.

Qiankun Gao, Chen Zhao, Bernard Ghanem, and Jian Zhang. R-dfcil: Relation-guided representation learning for data-free class incremental learning. In *European Conference on Computer Vision*, pp. 423–439. Springer, 2022.

Elad Gilboa, Yunus Saatçi, and John P. Cunningham. Scaling multidimensional inference for structured gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):424–436, 2015. doi: 10.1109/TPAMI.2013.192.

Dipam Goswami, Yuyang Liu, Bartłomiej Twardowski, and Joost Van De Weijer. Fecam: Exploiting the heterogeneity of class distributions in exemplar-free continual learning. *Advances in Neural Information Processing Systems*, 36:6582–6595, 2023.

Jouni Hartikainen and Simo Särkkä. Kalman filtering and smoothing solutions to temporal gaussian process regression models. In *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 379–384, 2010. doi: 10.1109/MLSP.2010.5589113.

Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 831–839, 2019.

Saurav Jha, Dong Gong, and Lina Yao. CLAP4CLIP: Continual learning with probabilistic finetuning for vision-language models. In *Thirty-eighth Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/pdf/2403.19137.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*, 2022.

Georgios A Kaissis, Marcus R Makowski, Daniel Rückert, and Rickmer F Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6): 305–311, 2020.

Hamid Kazemi, Atoosa Chegini, Jonas Geiping, Soheil Feizi, and Tom Goldstein. What do we learn from inverting clip models?, 2024. URL https://arxiv.org/abs/2403.02580.

Ronald Kemker and Christopher Kanan. Fearnet: Brain-inspired model for incremental learning. In *International Conference on Learning Representations*, 2018.

Alex Krizhevsky. Learning multiple layers of features from tiny images. pp. 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Olivier Ledoit and Michael Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, 2004. ISSN 0047-259X. doi: https://doi.org/10.1016/S0047-259X(03)00096-4. URL https://www.sciencedirect.com/science/article/pii/S0047259X03000964.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

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

Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pp. 2544–2553, 2021.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, Venkatesh Babu Radhakrishnan, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. In *International conference on machine learning*, pp. 4743–4751. PMLR, 2019.

Grégoire Petit, Adrian Popescu, Hugo Schindler, David Picard, and Bertrand Delezoide. Fetril: Feature translation for exemplar-free class-incremental learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 3911–3920, 2023.

Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *European conference on computer vision*, pp. 524–540. Springer, 2020.

Zihuan Qiu, Yi Xu, Fanman Meng, Hongliang Li, Linfeng Xu, and Qingbo Wu. Dual-consistency model inversion for non-exemplar class incremental learning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 24025–24035, 2024. doi: 10.1109/CVPR52733.2024.02268.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005. ISBN 9780262256834. doi: 10.7551/mitpress/3206.001.0001. URL https://doi.org/10.7551/mitpress/3206.001.0001.

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.

Donald B. Rubin and Dorothy T. Thayer. Em algorithms for ml factor analysis. *Psychometrika*, 47 (1):69–76, 1982. doi: 10.1007/BF02293851.

Grzegorz Rypeść, Sebastian Cygert, Tomasz Trzciński, and Bartłomiej Twardowski. Task-recency bias strikes back: Adapting covariances in exemplar-free class incremental learning. *Advances in Neural Information Processing Systems*, 37:63268–63289, 2024.

Simo Sarkka and Jouni Hartikainen. Infinite-dimensional kalman filtering approach to spatio-temporal gaussian process regression. In Neil D. Lawrence and Mark Girolami (eds.), *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pp. 993–1001, La Palma, Canary Islands, 21–23 Apr 2012. PMLR. URL https://proceedings.mlr.press/v22/sarkka12.html.

Wuxuan Shi and Mang Ye. Prototype reminiscence and augmented asymmetric knowledge aggregation for non-exemplar class-incremental learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1772–1781, 2023.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. Always be dreaming: A new approach for data-free class-incremental learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9374–9384, 2021.

James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11909–11919, June 2023.

Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999. doi: https://doi.org/10.1111/1467-9868.00196. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00196.

Ruilin Tong, Haodong Lu, Yuhang Liu, and Dong Gong. Model inversion with layer-specific modeling and alignment for data-free continual learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024. doi: 10.1109/TPAMI.2024.3367329.

Runqi Wang, Xiaoyue Duan, Guoliang Kang, Jianzhuang Liu, Shaohui Lin, Songcen Xu, Jinhu Lv, and Baochang Zhang. Attriclip: A non-incremental learner for incremental knowledge learning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3654–3663, 2023. doi: 10.1109/CVPR52729.2023.00356.

Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer G. Dy, and Tomas Pfister. Learning to prompt for continual learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 139–149, 2021. URL https://api.semanticscholar.org/CorpusID:245218925.

Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *European Conference on Computer Vision*, 2022.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 09 2010.

Chenshen Wu, Luis Herranz, Xialei Liu, Joost Van De Weijer, Bogdan Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. *Advances in neural information processing systems*, 31, 2018.

Jiayu Wu, Qixiang Zhang, and Guoxia Xu. Tiny imagenet challenge. 2017. URL https://api.semanticscholar.org/CorpusID:212697711.

Fei Ye and Adrian G Bors. Learning latent representations across multiple data domains using lifelong vaegan. In *European Conference on Computer Vision*, pp. 777–795. Springer, 2020.

Hongxu Yin, Pavlo Molchanov, Jose M Alvarez, Zhizhong Li, Arun Mallya, Derek Hoiem, Niraj K Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8715–8724, 2020.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23219–23230, 2024. doi: 10.1109/CVPR52733.2024.02191.

Lu Yu, Bartlomiej Twardowski, Xialei Liu, Luis Herranz, Kai Wang, Yongmei Cheng, Shangling Jui, and Joost van de Weijer. Semantic drift compensation for class-incremental learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6982–6991, 2020.

Da-Wei Zhou, Qi-Wei Wang, Han-Jia Ye, and De-Chuan Zhan. A model or 603 exemplars: Towards memory-efficient class-incremental learning. In *ICLR*, 2023.

Da-Wei Zhou, Yuanhan Zhang, Yan Wang, Jingyi Ning, Han-Jia Ye, De-Chuan Zhan, and Ziwei Liu. Learning without forgetting for vision-language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(6):4489–4504, 2025.

Kai Zhu, Wei Zhai, Yang Cao, Jiebo Luo, and Zhengjun Zha. Self-sustaining representation expansion for non-exemplar class-incremental learning. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9286–9295, 2022. URL https://api.semanticscholar.org/CorpusID:247447482.

APPENDIX

A DERIVATION OF THE FROBENIUS-NORM COVARIANCE OBJECTIVE

Given N centered feature vectors $\{\mathbf{v}_i\}_{i=1}^N$, the empirical covariance is $\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{i=1}^N \mathbf{v}_i\mathbf{v}_i^\top$. We parameterize the model covariance $\mathbf{M}$ using the Laplace kernel $K(\mathbf{a})_{ij} = e^{-|a_i - a_j|}$:

$$\mathbf{M} = \mathrm{diag}(\mathbf{d}) + \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w}).$$

We minimize the squared Frobenius distance $\mathcal{L}_\mathrm{F} = \|\mathbf{M} - \hat{\mathbf{\Sigma}}\|_\mathrm{F}^2$. Expanding the norm yields:

$$\|\mathbf{M} - \hat{\mathbf{\Sigma}}\|_\mathrm{F}^2 = \langle \mathbf{M}, \mathbf{M}\rangle_\mathrm{F} - \frac{2}{N}\sum_{i=1}^N \langle \mathbf{M}, \mathbf{v}_i\mathbf{v}_i^\top\rangle_\mathrm{F} + \langle \hat{\mathbf{\Sigma}}, \hat{\mathbf{\Sigma}}\rangle_\mathrm{F}.$$

The term $\langle \hat{\mathbf{\Sigma}}, \hat{\mathbf{\Sigma}}\rangle_\mathrm{F}$ is constant with respect to the model parameters and is omitted from the optimization objective.

We expand the model self-inner product $\langle \mathbf{M}, \mathbf{M}\rangle_\mathrm{F}$:

$$\begin{aligned}\langle \mathbf{M}, \mathbf{M}\rangle_\mathrm{F} =& \langle \mathrm{diag}(\mathbf{d}) + \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w}), \mathrm{diag}(\mathbf{d}) + \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w})\rangle_\mathrm{F} \\ =& \langle \mathrm{diag}(\mathbf{d}), \mathrm{diag}(\mathbf{d})\rangle_\mathrm{F} + 2\langle \mathrm{diag}(\mathbf{d}), \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w})\rangle_\mathrm{F} \\ &+ \langle \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w}), \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w})\rangle_\mathrm{F}.\end{aligned}$$

Evaluating each component individually:

$$\begin{aligned}\langle \mathrm{diag}(\mathbf{d}), \mathrm{diag}(\mathbf{d})\rangle_\mathrm{F} &= \|\mathbf{d}\|_2^2, \\ 2\langle \mathrm{diag}(\mathbf{d}), \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w})\rangle_\mathrm{F} &= 2\sum_j d_j w_j^2 (K(\mathbf{a}))_{jj} = 2\,\mathbf{d}^\top\mathbf{w}^2, \\ \langle \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w}), \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w})\rangle_\mathrm{F} &= \sum_{i,j} w_i^2 w_j^2 (K(\mathbf{a})_{ij})^2 = (\mathbf{w}^2)^\top K(2\mathbf{a})(\mathbf{w}^2).\end{aligned}$$

The kernel self-inner product relies on $(e^{-|a_i - a_j|})^2 = e^{-2|a_i - a_j|} = K(2\mathbf{a})_{ij}$. Thus, the complete model term is:

$$\langle \mathbf{M}, \mathbf{M}\rangle_\mathrm{F} = \|\mathbf{d}\|_2^2 + 2\,\mathbf{d}^\top\mathbf{w}^2 + (\mathbf{w}^2)^\top K(2\mathbf{a})(\mathbf{w}^2).$$

Next, we evaluate the data-alignment term using the identity $\langle \mathbf{M}, \mathbf{v}_i\mathbf{v}_i^\top\rangle_\mathrm{F} = \mathbf{v}_i^\top\mathbf{M}\mathbf{v}_i$:

$$\begin{aligned}\mathbf{v}_i^\top\mathbf{M}\mathbf{v}_i =& \mathbf{v}_i^\top \left(\mathrm{diag}(\mathbf{d}) + \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w})\right)\mathbf{v}_i \\ =& \mathbf{v}_i^\top \mathrm{diag}(\mathbf{d})\mathbf{v}_i + \mathbf{v}_i^\top \mathrm{diag}(\mathbf{w})K(\mathbf{a})\mathrm{diag}(\mathbf{w})\mathbf{v}_i \\ =& \mathbf{d}^\top\mathbf{v}_i^2 + (\mathbf{w}\odot\mathbf{v}_i)^\top K(\mathbf{a})(\mathbf{w}\odot\mathbf{v}_i).\end{aligned}$$

Summing over the batch yields the final tractable objective function:

$$\mathcal{L}_\mathrm{F} = \|\mathbf{d}\|_2^2 + 2\,\mathbf{d}^\top\mathbf{w}^2 + (\mathbf{w}^2)^\top K(2\mathbf{a})(\mathbf{w}^2) - \frac{2}{N}\sum_{i=1}^N \left(\mathbf{d}^\top\mathbf{v}_i^2 + (\mathbf{w}\odot\mathbf{v}_i)^\top K(\mathbf{a})(\mathbf{w}\odot\mathbf{v}_i)\right).$$

B QUASI-LINEAR TIME EVALUATION OF THE LAPLACE KERNEL VIA PREFIX AND SUFFIX SUMS

In the main text, we claim that the quadratic form $\mathbf{x}^\top K(\mathbf{a})\mathbf{x}$ can be evaluated in $\mathcal{O}(D\log D)$ time and linear $\mathcal{O}(D)$ memory without ever constructing the dense $D\times D$ kernel matrix $K(\mathbf{a})$. This efficiency is achieved by exploiting the multiplicative separability of the 1D Laplace kernel to compute the matrix-vector product $\mathbf{y} = K(\mathbf{a})\mathbf{x}$ using recursive prefix and suffix sums.

Formulating the Matrix-Vector Product. As established in Appendix C, we first apply a sorting permutation π such that the latent coordinates are monotonically increasing: $a_{\pi(1)} \leq a_{\pi(2)} \leq \cdots \leq a_{\pi(D)}$. Let $\tilde{\mathbf{a}}$ and $\tilde{\mathbf{x}}$ denote the permuted coordinate and input vectors, respectively. The i-th element of the matrix-vector product $\tilde{\mathbf{y}} = K(\tilde{\mathbf{a}})\tilde{\mathbf{x}}$ is given by the sum:

$$\tilde{y}_i = \sum_{j=1}^{D} e^{-|\tilde{a}_i - \tilde{a}_j|} \tilde{x}_j.$$

By explicitly splitting this sum into the elements occurring before i (the prefix), the element exactly at i, and the elements occurring after i (the suffix), we can drop the absolute value:

$$\tilde{y}_i = \underbrace{\sum_{j=1}^{i-1} e^{-(\tilde{a}_i - \tilde{a}_j)} \tilde{x}_j}_{\text{Prefix: } L_i} + \tilde{x}_i + \underbrace{\sum_{j=i+1}^{D} e^{-(\tilde{a}_j - \tilde{a}_i)} \tilde{x}_j}_{\text{Suffix: } R_i}.$$

Recursive Prefix and Suffix Evaluation. Because the exponential function is multiplicatively separable (i.e., $e^{-(\tilde{a}_i - \tilde{a}_j)} = e^{-(\tilde{a}_i - \tilde{a}_{i-1})} e^{-(\tilde{a}_{i-1} - \tilde{a}_j)}$), we can compute the prefix sum L_i and suffix sum R_i recursively, entirely avoiding nested loops.

We define the spatial decay factor between adjacent sorted coordinates as $\rho_i = e^{-(\tilde{a}_{i+1} - \tilde{a}_i)}$. The left-to-right prefix sum $\mathbf{L}$ is computed via a single forward pass:

$$L_1 = 0, \quad L_i = \rho_{i-1}(L_{i-1} + \tilde{x}_{i-1}) \quad \text{for } i = 2, \ldots, D.$$

Similarly, the right-to-left suffix sum $\mathbf{R}$ is computed via a single backward pass:

$$R_D = 0, \quad R_i = \rho_i(R_{i+1} + \tilde{x}_{i+1}) \quad \text{for } i = D-1, \ldots, 1.$$

Final Assembly and Complexity. Once the recursive passes are complete, the output vector $\tilde{\mathbf{y}}$ is simply assembled via an element-wise addition:

$$\tilde{\mathbf{y}} = \mathbf{L} + \tilde{\mathbf{x}} + \mathbf{R}.$$

We then apply the inverse permutation π^{-1} to restore the original channel ordering, yielding $\mathbf{y} = K(\mathbf{a})\mathbf{x}$. Finally, the quadratic form is computed via a standard dot product:

$$\mathbf{x}^\top K(\mathbf{a})\mathbf{x} = \mathbf{x}^\top \mathbf{y}.$$

By replacing the dense matrix–vector multiplication with a forward pass, a backward pass, and an element-wise sum, the computational complexity is reduced from $\mathcal{O}(D^2)$ to $\mathcal{O}(D \log D)$, where the $\log D$ term arises from sorting. Furthermore, because the memory footprint only requires storing the 1D vectors $\mathbf{L}$, $\mathbf{R}$, and $\boldsymbol{\rho}$, the memory complexity is bounded to $\mathcal{O}(D)$.

C Gaussian Markov Random Field and the Tridiagonal Precision Matrix

In this section, we show that the Laplace kernel on a one-dimensional coordinate system induces a Gaussian Markov Random Field (GMRF): the corresponding correlation matrix $K(\mathbf{a})$ has a strictly tridiagonal inverse. This Markov structure underlies the state-space representation introduced in Section 3 and enables the linear-time filtering recursions derived in Appendix D.

1D Ordering and Markov Structure. The kernel component of our model is defined as $(K(\mathbf{a}))_{ij} = e^{-|a_i - a_j|}$, where $\mathbf{a}$ denotes learned 1D coordinates. Without loss of generality, we assume the coordinates are sorted such that $a_1 \leq a_2 \leq \cdots \leq a_D$.

Define $\rho_i = e^{-(a_{i+1} - a_i)}$ as the correlation between adjacent channels. For any $i < j$, the distance decomposes as

$$|a_i - a_j| = \sum_{k=i}^{j-1} (a_{k+1} - a_k),$$

which implies a multiplicative factorization of the kernel:

$$\mathbf{K}_{ij} = e^{-\sum_{k=i}^{j-1}(a_{k+1}-a_k)} = \prod_{k=i}^{j-1} \rho_k, \quad \text{for } i \le j.$$

By symmetry, $\mathbf{K}_{ji} = \mathbf{K}_{ij}$. This decomposition reveals a Markov structure: correlations between distant channels are mediated through intermediate ones.

Precision Matrix via Autoregressive Representation. Let $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$, where $\mathbf{K}$ is a correlation matrix induced by the Laplace kernel. By construction, its diagonal entries satisfy $K_{i,i} = 1$, implying unit marginal variance for all variables, i.e., $\mathrm{Var}(x_i) = 1$.

Due to the Markov structure of the 1D exponential kernel, $\mathbf{x}$ admits an equivalent autoregressive (AR(1)) representation:

$$\begin{aligned} x_1 &= \epsilon_1, & \epsilon_1 &\sim \mathcal{N}(0,1), \\ x_i &= \rho_{i-1} x_{i-1} + \epsilon_i, & \epsilon_i &\sim \mathcal{N}(0, 1-\rho_{i-1}^2), \quad i \ge 2, \end{aligned}$$

where ρ_{i-1} encodes the correlation between consecutive variables.

The variance of the innovations ϵ_i follows directly from the unit marginal constraint. Applying the variance operator to $x_i = \rho_{i-1}x_{i-1} + \epsilon_i$ and using independence yields

$$\mathrm{Var}(x_i) = \rho_{i-1}^2 \mathrm{Var}(x_{i-1}) + \mathrm{Var}(\epsilon_i).$$

Substituting $\mathrm{Var}(x_i) = \mathrm{Var}(x_{i-1}) = 1$ gives $\mathrm{Var}(\epsilon_i) = 1 - \rho_{i-1}^2$.

This system can be written compactly as $\mathbf{L}\mathbf{x} = \boldsymbol{\epsilon}$, where $\mathbf{L}$ is a lower bidiagonal matrix:

$$\mathbf{L} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -\rho_1 & 1 & 0 & \dots & 0 \\ 0 & -\rho_2 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -\rho_{D-1} & 1 \end{bmatrix},$$

and $\boldsymbol{\epsilon}$ has diagonal covariance

$$\boldsymbol{\Lambda} = \mathrm{diag}(1,\, 1-\rho_1^2,\, \dots,\, 1-\rho_{D-1}^2).$$

It follows that the covariance and precision matrices admit the factorization

$$\mathbf{K} = \mathbf{L}^{-1}\boldsymbol{\Lambda}\mathbf{L}^{-\top}, \quad \mathbf{Q} = \mathbf{K}^{-1} = \mathbf{L}^{\top}\boldsymbol{\Lambda}^{-1}\mathbf{L}.$$

Explicit Form of the Precision Matrix. The inverse covariance $\mathbf{Q}$ is given by $\mathbf{Q} = \mathbf{L}^{\top}\boldsymbol{\Lambda}^{-1}\mathbf{L}$, where

$$\boldsymbol{\Lambda}_{1,1}^{-1} = 1, \quad \boldsymbol{\Lambda}_{k,k}^{-1} = \frac{1}{1-\rho_{k-1}^2}, \quad 1 < k \le D.$$

For diagonal entries:

- $i = 1$:
$$\mathbf{Q}_{1,1} = \frac{1}{1-\rho_1^2}.$$
- $1 < i < D$:
$$\mathbf{Q}_{i,i} = \frac{1}{1-\rho_{i-1}^2} + \frac{\rho_i^2}{1-\rho_i^2}.$$
- $i = D$:
$$\mathbf{Q}_{D,D} = \frac{1}{1-\rho_{D-1}^2}.$$

For off-diagonal entries, by symmetry it suffices to consider $i < j$:

$$\mathbf{Q}_{i,j} = \sum_{k=1}^{D} \mathbf{L}_{k,i} \mathbf{\Lambda}_{k,k}^{-1} \mathbf{L}_{k,j}.$$

Due to the bidiagonal structure:

- If $j = i + 1$:
$$\mathbf{Q}_{i,i+1} = -\frac{\rho_i}{1 - \rho_i^2}.$$
- If $|i - j| > 1$:
$$\mathbf{Q}_{i,j} = 0.$$

This derivation shows that the precision matrix $\mathbf{Q}$ is strictly tridiagonal. Consequently, the Laplace kernel over 1D coordinates induces a GMRF with local dependencies, enabling efficient computation while preserving structured correlations.

Remark. The sparsity discussed above applies to the kernel $K(\mathbf{a})$ and is *not* inherited by the LCM covariance $\mathbf{\Sigma} = \mathrm{diag}(\mathbf{d}) + \mathrm{diag}(\mathbf{w})K(\mathbf{a})\,\mathrm{diag}(\mathbf{w})$. In particular, $\mathbf{\Sigma}^{-1}$ is generally dense due to the independent diagonal noise $\mathbf{d}$. The banded precision structure exploited in Appendix D instead belongs to the joint latent–observed state-space representation, which enables the corresponding linear-time likelihood computation.

D Exact Negative Log-Likelihood Calculation

We regularize intermediate activations to match the feature statistics observed on real data by modeling their distribution with the proposed LCM and minimizing the exact Gaussian negative log-likelihood (NLL).

Given a batch of generated features $\{\hat{\mathbf{h}}_{l,i}\}_{i=1}^{N} \subset \mathbb{R}^D$, the NLL under the multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}^{(l)}, \mathbf{\Sigma}^{(l)})$ is

$$\mathcal{L}_{\text{stat}}^{(l)} = \frac{1}{2N} \sum_{i=1}^{N} \left(\log\det \mathbf{\Sigma}^{(l)} + (\hat{\mathbf{h}}_{l,i} - \boldsymbol{\mu}^{(l)})^\top (\mathbf{\Sigma}^{(l)})^{-1} (\hat{\mathbf{h}}_{l,i} - \boldsymbol{\mu}^{(l)}) \right). \tag{7}$$

For a dense covariance, evaluating the quadratic form and log-determinant directly requires $\mathcal{O}(D^3)$ time. This is prohibitive for high-dimensional feature representations. The LCM structure instead allows both quantities to be evaluated exactly using the state-space formulation described below, without forming or inverting $\mathbf{\Sigma}^{(l)}$.

Generative Matrix Factorization. A random vector distributed as $\mathcal{N}(\boldsymbol{\mu}^{(l)}, \mathbf{\Sigma}^{(l)})$ can be constructively generated by drawing a latent vector $\mathbf{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$ and an independent noise vector $\boldsymbol{\eta}_i \sim \mathcal{N}(\mathbf{0}, \mathrm{diag}(\mathbf{d}))$, where $\mathbf{W} = \mathrm{diag}(\mathbf{w})$, such that:

$$\hat{\mathbf{h}}_{l,i} - \boldsymbol{\mu}^{(l)} = \mathbf{W}\mathbf{z}_i + \boldsymbol{\eta}_i. \tag{8}$$

Eq. 8 yields the exact covariance $\mathbb{E}[(\mathbf{W}\mathbf{z} + \boldsymbol{\eta})(\mathbf{W}\mathbf{z} + \boldsymbol{\eta})^\top] = \mathbf{W}\mathbf{K}\mathbf{W}^\top + \mathrm{diag}(\mathbf{d}) = \mathbf{\Sigma}^{(l)}$. This factorization reveals that the final generated features are simply a standard correlated latent process ($\mathbf{z}$), scaled element-wise by the diagonal weights ($\mathbf{W}$), with added noise ($\boldsymbol{\eta}$).

Mapping to a 1D State-Space Model. Recall from Eq. 8 that the generated features are driven by a latent random vector $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$. Conceptually, $\mathbf{z}$ represents an unobserved, normalized hidden signal that correlates all the feature channels together.

The dependence of $K_{ij} = \exp(-|a_i - a_j|)$ only on the distance $|a_i - a_j|$ makes the kernel stationary, while its exponential form gives the corresponding Gaussian process the Markov property along the ordered coordinates. We therefore apply a permutation π that sorts the channels by their learned coordinates, such that $a_{\pi(1)} \leq \cdots \leq a_{\pi(D)}$.

Under this sorted ordering, we can break down the dense vector equation (Eq. 8) into a step-by-step scalar sequence. Let $z_{i,k}$ denote the k-th scalar element of the sorted hidden signal of sample i. Similarly, let $x_{i,k} = \hat{h}_{l,i,\pi(k)}$ be the actual observed feature for the k-th sorted channel, with corresponding historical parameters $\mu_k = \mu^{(l)}_{\pi(k)}$, $w_k = w^{(l)}_{\pi(k)}$, and $d_k = d^{(l)}_{\pi(k)}$, where $\mathbf{d} = \mathrm{softplus}(\mathbf{u}) + \varepsilon\mathbf{1}$ is the diagonal variance of Eq. 3 and $\mathbf{u}$ its unconstrained parameterization.

The spatial distance between adjacent channels is $\Delta a_k = a_{\pi(k)} - a_{\pi(k-1)}$ for $k \geq 2$. This distance dictates how strongly the hidden signal decays from one channel to the next, defining the autoregressive transition $\phi_k = \exp(-\Delta a_k)$ and the injected process noise $q_k = 1 - \phi_k^2$. At the first channel we set $\phi_1 = 0$ and $q_1 = 1$, which reproduces the marginal $z_{i,1} \sim \mathcal{N}(0, 1)$. The dense multivariate model is thus mathematically equivalent to a 1D linear state-space model (SSM):

$$z_{i,k} = \phi_k z_{i,k-1} + \epsilon_{i,k}, \quad \epsilon_{i,k} \sim \mathcal{N}(0, q_k), \tag{9}$$

$$x_{i,k} = \mu_k + w_k z_{i,k} + \eta_{i,k}, \quad \eta_{i,k} \sim \mathcal{N}(0, d_k). \tag{10}$$

Eq. 9 describes how the unobserved hidden signal $z_{i,k}$ evolves smoothly across the sorted channels. Eq. 10 demonstrates how the actual observed feature $x_{i,k}$ is generated: by scaling the hidden signal by the weight w_k, shifting it by the historical mean μ_k, and adding independent channel noise $\eta_{i,k}$. Marginalizing $\mathbf{z}$ out of Eqs. 9–10 returns exactly $\mathrm{Cov}(\mathbf{x}) = \mathrm{diag}(\mathbf{d}) + \mathrm{diag}(\mathbf{w})\mathbf{K}(\mathbf{a})\mathrm{diag}(\mathbf{w})$, so the SSM is a reparameterization of the LCM rather than an approximation of it.

Exact Likelihood via Kalman Filtering. To compute the exact likelihood of the entire sequence, we use a Kalman filter. The filter moves step-by-step through the sorted channels, maintaining a running estimate of the true hidden signal. At the beginning of step k, our knowledge of the previous hidden state $z_{i,k-1}$ is represented by a Gaussian distribution with mean $m_{i,k-1}$ and variance P_{k-1}, initialized as $m_{i,0} = 0$ and $P_0 = 1$. The filtering process at each channel follows three distinct stages.

Before observing the k-th channel, we use the autoregressive transition (Eq. 9) to predict the current hidden state. The expected mean and variance of this prior prediction are:

$$m_{i,k|k-1} = \phi_k m_{i,k-1},$$
$$P_{k|k-1} = \phi_k^2 P_{k-1} + q_k.$$

With $\phi_1 = 0$ and $q_1 = 1$ this gives $m_{i,1|0} = 0$ and $P_{1|0} = 1$, as required.

Next, we predict what the actual observed feature $x_{i,k}$ should be, based on our hidden state prediction. The difference between the true observation and our expectation is called the *innovation* (or prediction error) $v_{i,k}$:

$$v_{i,k} = (x_{i,k} - \mu_k) - w_k \underbrace{m_{i,k|k-1}}_{\text{Predicted State}}.$$

Because both the hidden state and the channel observation noise are uncertain, the variance of this expected observation, denoted as S_k, is the sum of both uncertainties:

$$S_k = w_k^2 \underbrace{P_{k|k-1}}_{\text{Prior State Variance}} + \underbrace{d_k}_{\text{Channel Noise}}.$$

Finally, to prepare for the next step $k + 1$, the filter corrects its internal estimate of the hidden state using the newly observed prediction error $v_{i,k}$. The degree of this correction is controlled by the Kalman gain $G_k = P_{k|k-1} w_k / S_k$, which acts as a dynamic "trust ratio." It balances how much the filter trusts its own prior prediction ($P_{k|k-1}$) versus the noisy new observation (S_k).

Using this gain, we compute the updated posterior mean and variance:

$$m_{i,k} = m_{i,k|k-1} + G_k v_{i,k}, \tag{11}$$

$$P_k = (1 - G_k w_k) P_{k|k-1}. \tag{12}$$

Eq. 11 intuitively shows that the new mean is simply the old prediction shifted by a fraction of the error. Eq. 12 demonstrates that observing new data always reduces our uncertainty: the prior variance $P_{k|k-1}$ is strictly shrunk by the factor $(1 - G_k w_k)$. Note that P_k, S_k and G_k depend only on the

LCM parameters and not on the sample index i; the entire variance recursion is therefore run once per layer, and only the scalar means $m_{i,k}$ are propagated per sample.

The primary advantage of this sequential formulation is how elegantly it simplifies the likelihood calculation. Direct evaluation of the full-covariance multivariate Gaussian requires calculating a massive, tangled joint probability. However, the Kalman filter mathematically untangles this vector into a sequence of purely independent "surprises"—the innovations $v_{i,k}$.

Because each innovation $v_{i,k}$ captures only the orthogonal, completely new information arriving at channel k, these errors are statistically independent from one another and follow a simple 1D Gaussian distribution: $v_{i,k} \sim \mathcal{N}(0, S_k)$. This is the classical prediction-error decomposition, which for the model of Eqs. 9–10 yields the two exact identities

$$\log\det \mathbf{\Sigma}^{(l)} = \sum_{k=1}^{D} \log S_k, \qquad (\hat{\mathbf{h}}_{l,i} - \boldsymbol{\mu}^{(l)})^\top (\mathbf{\Sigma}^{(l)})^{-1} (\hat{\mathbf{h}}_{l,i} - \boldsymbol{\mu}^{(l)}) = \sum_{k=1}^{D} \frac{v_{i,k}^2}{S_k},$$

neither of which requires $\mathbf{\Sigma}^{(l)}$ to be formed. Substituting them into Eq. 7 replaces its $\mathcal{O}(D^3)$ dense evaluation by a sum of standard 1D NLLs, computable in $\mathcal{O}(D \log D)$ time (dominated by the sort) and $\mathcal{O}(D)$ memory:

$$\mathcal{L}_{\text{stat}}^{(l)} = \frac{1}{2N} \sum_{i=1}^{N} \sum_{k=1}^{D} \left(\log(2\pi S_k) + \frac{v_{i,k}^2}{S_k} \right),$$

which equals Eq. 7 up to the additive constant $\frac{D}{2} \log 2\pi$ and therefore has identical gradients.

E Generator Training Details

In DFCIL, the generator acts as a synthesizer that produces pseudo-samples $\mathbf{X}_{\text{old}}$ for previously learned tasks. Following the PMI framework, we do not train an auxiliary parametric generative model (e.g., a GAN), but instead instantiate the generator through a *per-layer model inversion* procedure. In this formulation, intermediate feature tensors are treated as optimization variables, enabling the reconstruction of high-fidelity inputs by propagating constraints backward through the frozen network in a layer-wise manner.

Building on this paradigm, we extend PMI with REMIX, enabling explicit modeling of structured correlations within feature representations during inversion and overcoming the diagonal covariance assumptions used in prior work.

Contrastive Feature Selection (CFS). A key challenge in model inversion is defining informative and diverse semantic targets. In standard classification models, mapping a high-dimensional feature vector to a scalar label discards a substantial amount of information. As a result, relying solely on class centroids as inversion targets often leads to mode collapse and synthetic samples that lack diversity.

To preserve the geometry of the original feature distribution, we follow PMI and adopt CFS. For each previously learned class c, we approximate the feature distribution at the penultimate layer using a Gaussian model $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2)$.

We then train a lightweight contrastive MLP f_{cont} on real features using a negative contrastive objective:

$$\mathcal{L}_{\text{cont}}(\mathbf{h}_{L,i}, \mathcal{S}_{\text{neg}}; f_{\text{cont}}) = \log \mathbb{E}_{\mathbf{h}_{L,j} \in \mathcal{S}_{\text{neg}}} \left[e^{\cos(f_{\text{cont}}(\mathbf{h}_{L,i}), f_{\text{cont}}(\mathbf{h}_{L,j}))} \right],$$

where $\mathcal{S}_{\text{neg}}$ denotes a negative set and $\cos(\cdot,\cdot)$ is cosine similarity. Optimizing this objective encourages features to be uniformly distributed on a hypersphere, ensuring that no semantic subspace is overly compressed.

During generation, we sample a large candidate pool from $\mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\sigma}_c^2)$ and apply a greedy selection procedure (see Algorithm 2 in Tong et al. (2025)). At each step, we select features that minimize the contrastive loss with respect to already chosen samples. This produces a subset $\mathcal{S}_{\text{feat}}$ that maximizes diversity and distributional coverage, providing robust semantic anchors for inversion.

Linking Generation to Per-layer Inversion. We decompose the feature extractor f_{feat} into a sequence of L blocks (e.g., residual blocks for CNNs or transformer blocks for ViTs). Directly optimizing pixels to match high-level semantic representations is highly non-convex. To address this, we decompose the problem into a sequence of tractable layer-wise sub-problems.

Let $\mathbf{h}_l$ denote the input to block l, and $\mathbf{h}_{l+1}$ its output. The inversion proceeds backward from the deepest layer L to the input:

1. Obtain target features $\mathcal{S}_{\text{feat}}$ at layer L using CFS.
2. Initialize $\hat{\mathbf{h}}_l$ as Gaussian noise scaled by stored input statistics.
3. Optimize $\hat{\mathbf{h}}_l$ via gradient descent so that its forward pass through the frozen block matches the target.
4. Use the optimized $\hat{\mathbf{h}}_l$ as the target for the preceding layer $l-1$.

This sequential procedure decomposes a highly non-convex global objective into a series of well-conditioned local problems.

Layer-wise Optimization Objective. At each block $l > 0$, we optimize $\hat{\mathbf{h}}_l$ using

$$\mathcal{L}_{\text{layer}}^{(l)} = \alpha_{\text{match}}\mathcal{L}_{\text{match}}^{(l)} + \alpha_{\text{stat}}\mathcal{L}_{\text{stat}}^{(l)} + \alpha_{\text{in}}\mathcal{L}_{\text{in}}^{(l)}.$$

1. Feature Matching Loss ($\mathcal{L}_{\text{match}}^{(l)}$). The formulation of the deterministic matching loss depends strictly on the depth of the layer being optimized.

At the topmost layer ($l = L$), the optimization aims to align the generated features directly with the target class label y. Therefore, the matching objective utilizes a standard CE loss: $\mathcal{L}_{\text{match}}^{(L)} = \mathcal{L}_{\text{ce}}(\hat{\mathbf{h}}_L, y)$.

However, for all intermediate layers ($l < L$), the optimization relies on the cascading targets from the layer above. Here, the objective transitions to a Mean Squared Error (MSE) loss, which enforces consistency with the localized target features propagated backward from the previously optimized layer:

$$\mathcal{L}_{\text{match}}^{(l)} = \left\|\text{norm}(\text{Block}_l(\hat{\mathbf{h}}_l)) - \mathcal{S}_{\text{feat}}^{(l+1)}\right\|_2^2.$$

Specifically, $\hat{\mathbf{h}}_l$ is the intermediate feature representation being optimized, which serves as the input to Block_l. Block_l refers to a primary structural unit of the frozen network, $\text{norm}(\cdot)$ denotes the normalization applied to the block's output, and $\mathcal{S}_{\text{feat}}^{(l+1)}$ represents the fixed target features obtained from the previously optimized layer $l+1$.

2. Distribution Statistic Loss ($\mathcal{L}_{\text{stat}}^{(l)}$): This term aligns the activations of synthesized samples with the feature distribution observed on real data. Let $\hat{\mathbf{h}}_{l,i}$ denote the activation of the i-th synthesized sample at layer l, and let $(\boldsymbol{\mu}^{(l)}, \boldsymbol{\Sigma}^{(l)})$ denote the aggregated LCM for that layer from Eq. 3. We use the exact Gaussian negative log-likelihood

$$\mathcal{L}_{\text{stat}}^{(l)} = \frac{1}{2N}\sum_{i=1}^{N}\big(\hat{\mathbf{h}}_{l,i} - \boldsymbol{\mu}^{(l)}\big)^\top\big(\boldsymbol{\Sigma}^{(l)}\big)^{-1}\big(\hat{\mathbf{h}}_{l,i} - \boldsymbol{\mu}^{(l)}\big) + \tfrac{1}{2}\log\det\boldsymbol{\Sigma}^{(l)}, \tag{13}$$

up to an additive constant. Unlike the moment-matching losses used in prior data-free inversion methods, this objective accounts for the full covariance structure, rather than only the marginal variances. The quadratic form and log-determinant are evaluated by the filtering recursions in Appendix D in $\mathcal{O}(D\log D)$ time and $\mathcal{O}(D)$ memory, without forming or inverting $\boldsymbol{\Sigma}^{(l)}$.

3. Input Statistic Prior ($\mathcal{L}_{\text{in}}^{(l)}$): Due to the strong non-linearity of deep networks, directly optimizing the input tensor $\hat{\mathbf{h}}_l$ to match downstream targets can lead to adversarial or out-of-distribution activations. Although such features may satisfy the feature-matching objective, they often lack meaningful structure and can destabilize subsequent inversion steps.

To mitigate this issue, we constrain $\hat{\mathbf{h}}_l$ using a Gaussian prior defined over the input activations of block l, of which there are C_l channels. During training on tasks $1{:}t$, we record the empirical channel-wise mean and variance of these activations, denoted as $\boldsymbol{\mu}_{1:t}^{(l,\text{in})}$ and $(\boldsymbol{\sigma}_{1:t}^{(l,\text{in})})^2$.

During inversion, we enforce the empirical batch statistics of the optimized tensor ($\hat{\boldsymbol{\mu}}^{(l)}$ and $(\hat{\boldsymbol{\sigma}}^{(l)})^2$) to match these stored values. This is achieved by minimizing the KL divergence between the current batch distribution and the historical Gaussian prior, applied independently across channels:

$$\mathcal{L}_{\text{in}}^{(l)} = \frac{1}{2C_l} \sum_{c=1}^{C_l} \left(\log \frac{(\sigma_{1:t,c}^{(l,\text{in})})^2}{(\hat{\sigma}_c^{(l)})^2} + \frac{(\hat{\sigma}_c^{(l)})^2 + (\hat{\mu}_c^{(l)} - \mu_{1:t,c}^{(l,\text{in})})^2}{(\sigma_{1:t,c}^{(l,\text{in})})^2} - 1 \right).$$

This channel-wise prior anchors the optimization to the distribution of features naturally observed at the input of block l, preventing drift toward unrealistic activations and stabilizing the layer-wise inversion process.

Pixel-Level Optimization. The final output of the generator must be replay images $\hat{\mathbf{x}} \in \mathbb{R}^{3\times H\times W}$; therefore, the last stage of layer-wise inversion ($l = 0$) operates directly in pixel space. However, optimizing pixels solely to match high-level semantic features often yields high-frequency, adversarial artifacts that are unsuitable for continual learning. To mitigate this, we augment the objective with an image prior $\mathcal{L}_{\text{pr}}$ regularization, which enforces spatial smoothness by penalizing local differences:

$$\mathcal{L}_{\text{pr}}(\hat{\mathbf{x}}) = \sum_{c=1}^{3} \sum_{h=1}^{H-1} \sum_{w=1}^{W-1} \left((\hat{x}_{c,h+1,w} - \hat{x}_{c,h,w})^2 + (\hat{x}_{c,h,w+1} - \hat{x}_{c,h,w})^2 \right).$$

Full-Model Refinement and Pseudo-Sample Generation. Although layer-wise inversion decomposes a highly non-convex optimization problem into tractable sub-problems, small approximation errors inevitably accumulate across layers. To correct these errors, the intermediate reconstructions $\hat{\mathbf{x}}$ are not treated as final outputs, but instead serve as strong initializations for a subsequent end-to-end refinement stage.

During this phase, the reconstructed inputs $\hat{\mathbf{x}}$ are optimized directly through the entire frozen feature extractor f_{feat}. Unlike the sequential inversion stage, intermediate layer-specific targets ($\mathcal{S}_{\text{feat}}^{(l+1)}$) are no longer available, and therefore the layer-wise reconstruction losses $\mathcal{L}_{\text{mse}}^{(l)}$ are discarded.

The refinement objective instead combines the final-layer CE loss $\mathcal{L}_{\text{ce}}$, which preserves target class semantics, with the proposed full-covariance Distribution Statistic Loss $\mathcal{L}_{\text{stat}}^{(l)}$ evaluated across intermediate representations. In this setting, the statistical prior replaces explicit layer-wise supervision and constrains the generated samples to remain consistent with the learned feature distribution. This final global optimization stage compensates for accumulated approximation errors and produces pseudo-samples $\mathbf{X}_{\text{old}}$ with improved structural and semantic coherence for replay.

F Feature Extractor and Classifier Training Details

In this section, we provide a detailed description of the feature extractor and classifier training protocol used in our method. The procedure follows the two-stage optimization strategy introduced in R-DFCIL and adopted in our REMIX framework. We emphasize both the formal definitions and the underlying intuition behind each component.

Problem Formulation and Notation. Let $f_{\text{feat}}(\cdot;\boldsymbol{\theta})$ denote the feature extractor and $f_{\text{head}}(\cdot;\boldsymbol{\phi})$ denote the linear classification head. Assume the model has already been trained on tasks 1 through t. We retain a frozen copy of this historical model, parameterized by $(\boldsymbol{\theta}_{1:t}, \boldsymbol{\phi}_{1:t})$, which serves as a teacher during subsequent training.

When learning a new task $t+1$, the model updates its active parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$. To accommodate the new label space $\mathcal{Y}_{t+1}$, the classifier is expanded with additional parameters, denoted by $\boldsymbol{\phi}_{\text{new}}$, while the parameters corresponding to previously learned classes are denoted by $\boldsymbol{\phi}_{\text{old}}$.

A key challenge in this setting is catastrophic forgetting caused by the absence of past data. To mitigate this, the training process interleaves two types of data: (i) real samples from the current task, $(\mathbf{X}_{\text{new}}, Y_{\text{new}}) \subset \mathcal{D}_{t+1}$, and (ii) synthesized pseudo-samples from previous tasks, $(\mathbf{X}_{\text{old}}, Y_{\text{old}})$.

Stage 1: Relation-Guided Representation Learning (RRL). In the first stage, we jointly optimize the feature extractor and classifier parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$ using the relation-guided representation learning objective:

$$\mathcal{L}_{\text{rrl}} = \lambda_{\text{lce}}\mathcal{L}_{\text{lce}} + \lambda_{\text{hkd}}\mathcal{L}_{\text{hkd}} + \lambda_{\text{rkd}}\mathcal{L}_{\text{rkd}}.$$

These three components are designed to balance the competing objectives of learning new information (plasticity) and preserving previously acquired knowledge (stability).

1. Local Classification Loss ($\mathcal{L}_{\text{lce}}$). The local classification loss is a standard CE objective applied exclusively to real samples from the current task. By restricting supervision to $\mathbf{X}_{\text{new}}$, the model learns new classes without being biased by imperfections in synthetic data:

$$\mathcal{L}_{\text{lce}} = \frac{1}{|\mathbf{X}_{\text{new}}|} \sum_{(\mathbf{x},y)\in(\mathbf{X}_{\text{new}},Y_{\text{new}})} \mathcal{L}_{\text{ce}}\big(\text{softmax}(f_{\text{head}}(f_{\text{feat}}(\mathbf{x};\boldsymbol{\theta});\boldsymbol{\phi}_{\text{new}})), y\big).$$

2. Hard Knowledge Distillation ($\mathcal{L}_{\text{hkd}}$). To explicitly preserve knowledge from previous tasks, we apply hard knowledge distillation on synthetic samples. This term enforces consistency between the outputs of the current model and the frozen teacher by penalizing deviations in logits:

$$\mathcal{L}_{\text{hkd}} = \frac{1}{|\mathbf{X}_{\text{old}}|\,|\mathcal{Y}_{1:t}|} \sum_{\mathbf{x}\in\mathbf{X}_{\text{old}}} \left\| f_{\text{head}}(f_{\text{feat}}(\mathbf{x};\boldsymbol{\theta}_{1:t});\boldsymbol{\phi}_{1:t}) - f_{\text{head}}(f_{\text{feat}}(\mathbf{x};\boldsymbol{\theta});\boldsymbol{\phi}_{\text{old}}) \right\|_1 .$$

3. Relational Knowledge Distillation ($\mathcal{L}_{\text{rkd}}$). While $\mathcal{L}_{\text{hkd}}$ constrains absolute predictions, it can overly restrict the feature space. To counterbalance this effect, we introduce relational knowledge distillation, which preserves the geometric structure of the feature space by matching angular relationships between features.

Let u and v be learnable projection functions, and define $\mathbf{t}_k = u(f_{\text{feat}}(\mathbf{x}_k;\boldsymbol{\theta}_{1:t}))$ and $\mathbf{s}_k = v(f_{\text{feat}}(\mathbf{x}_k;\boldsymbol{\theta}))$. For a triplet $(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$, the loss is:

$$\mathcal{L}_{\text{rkd}} = \frac{1}{|\mathbf{X}_{\text{new}}|^3} \sum_{\mathbf{x}_a,\mathbf{x}_b,\mathbf{x}_c\in\mathbf{X}_{\text{new}}} \left\| \cos\angle(\mathbf{t}_a,\mathbf{t}_b,\mathbf{t}_c) - \cos\angle(\mathbf{s}_a,\mathbf{s}_b,\mathbf{s}_c) \right\|_1 .$$

Adaptive Scaling Factors. The contributions of individual loss terms are dynamically adjusted based on the ratio of new to previously learned classes:

$$\lambda_{\text{lce}} = \frac{1+1/\alpha}{\beta}\lambda_{\text{lce}}^{\text{base}}, \quad \lambda_{\text{hkd}} = \alpha\beta\lambda_{\text{hkd}}^{\text{base}}, \quad \lambda_{\text{rkd}} = \alpha\beta\lambda_{\text{rkd}}^{\text{base}},$$

where

$$\alpha = \log_2\left(\frac{|\mathcal{Y}_{t+1}|}{2} + 1\right), \quad \beta = \sqrt{\frac{|\mathcal{Y}_{1:t}|}{|\mathcal{Y}_{t+1}|}}.$$

Stage 2: Classification Head Refinement. In the second stage, the feature extractor is frozen ($\boldsymbol{\theta}^*$), and only the classifier $\boldsymbol{\phi}$ is updated. The goal of this stage is to correct biases introduced during the first stage and to ensure balanced decision boundaries across all classes.

We optimize a global, class-balanced CE loss over the combined dataset $\mathbf{X}_{\text{all}} = \mathbf{X}_{\text{new}} \cup \mathbf{X}_{\text{old}}$:

$$\mathcal{L}_{\text{gce}} = \frac{1}{|\mathbf{X}_{\text{all}}|} \sum_{(\mathbf{x},y)\in(\mathbf{X}_{\text{all}},Y_{\text{all}})} \frac{w_y}{\sum_j w_j} \mathcal{L}_{\text{ce}}\big(\text{softmax}(f_{\text{head}}(f_{\text{feat}}(\mathbf{x};\boldsymbol{\theta}^*);\boldsymbol{\phi})), y\big),$$

where w_y denotes the inverse frequency of class y in the current training data.

G Approximating a Linear Combination of LCMs

The continual aggregation step of Eq. 5 forms a convex combination of fitted LCM covariances and then projects it back onto the LCM family (Eq. 6). This appendix derives that projection. Three results follow: the objective depends on the inputs only through Frobenius inner products between family members (Sec. G.3); those inner products have a closed form (Prop. 2); and the diagonal parameters can be eliminated analytically, so the merged model reproduces the marginal variances *exactly* and the entire approximation error lies in the off-diagonal (Prop. 3). At no point is a $D \times D$ matrix formed.

G.1 Setup

Write $\boldsymbol{\vartheta} = (\mathbf{d}, \mathbf{w}, \mathbf{a})$ for the LCM parameters of one layer and

$$\boldsymbol{\Sigma}(\boldsymbol{\vartheta}) = \operatorname{diag}(\mathbf{d}) + \operatorname{diag}(\mathbf{w})\, K(\mathbf{a})\, \operatorname{diag}(\mathbf{w}), \qquad K_{ij}(\mathbf{a}) = e^{-|a_i - a_j|}, \tag{14}$$

so that $\boldsymbol{\Sigma}(\boldsymbol{\vartheta})_{ij} = d_i \delta_{ij} + w_i w_j K_{ij}(\mathbf{a})$. We use $K_{ii} = 1$ and $K_{ij}(\mathbf{a})^2 = K_{ij}(2\mathbf{a})$ throughout; $\langle \mathbf{A}, \mathbf{B} \rangle_{\mathrm{F}} = \sum_{ij} A_{ij} B_{ij}$, and $\odot$ is the element-wise product.

Given K fitted models $\boldsymbol{\vartheta}_1, \ldots, \boldsymbol{\vartheta}_K$ and weights $\pi_k \geq 0$ with $\sum_k \pi_k = 1$, the target is

$$\mathbf{T} = \sum_{k=1}^{K} \pi_k\, \boldsymbol{\Sigma}(\boldsymbol{\vartheta}_k), \tag{15}$$

and the merged parameters are

$$\boldsymbol{\vartheta}^\star = \arg\min_{\boldsymbol{\vartheta}} \mathcal{L}(\boldsymbol{\vartheta}), \qquad \mathcal{L}(\boldsymbol{\vartheta}) = \left\| \boldsymbol{\Sigma}(\boldsymbol{\vartheta}) - \mathbf{T} \right\|_{\mathrm{F}}^2,$$

minimised over all of $(\mathbf{d}, \mathbf{w}, \mathbf{a})$. The aggregation of Eq. 5 is the case $K = 2$ with $(\pi_1, \pi_2) = (\alpha_t, \beta_t)$.

G.2 Correlation parameterization

The covariance family in Eq. 14 can be normalized to unit diagonal, allowing the marginal scale to be separated from the correlation structure. Let $\boldsymbol{\vartheta} = (\mathbf{d}, \mathbf{w}, \mathbf{a})$ with $\mathbf{w} \geq 0$ and define

$$\sigma_i^2 = \boldsymbol{\Sigma}(\boldsymbol{\vartheta})_{ii} = d_i + w_i^2, \qquad \rho_i = \frac{w_i^2}{d_i + w_i^2} \in [0, 1). \tag{16}$$

Here, ρ_i is the fraction of the variance of coordinate i explained by the latent process. Thus, $\rho_i \to 0$ corresponds to an independent coordinate, while $\rho_i \to 1$ corresponds to a fully latent-driven coordinate. The condition $d_i \geq \varepsilon > 0$ ensures $\rho_i < 1$.

Proposition 1. *The covariance can be written as*

$$\boldsymbol{\Sigma}(\boldsymbol{\vartheta}) = \operatorname{diag}(\boldsymbol{\sigma})\, \mathbf{R}(\boldsymbol{\rho}, \mathbf{a})\, \operatorname{diag}(\boldsymbol{\sigma}),$$

where

$$\mathbf{R}(\boldsymbol{\rho}, \mathbf{a})_{ij} = \sqrt{\rho_i \rho_j}\, e^{-|a_i - a_j|} \quad (i \neq j), \qquad \mathbf{R}(\boldsymbol{\rho}, \mathbf{a})_{ii} = 1. \tag{17}$$

Moreover,

$$\mathbf{R}(\boldsymbol{\rho}, \mathbf{a}) = \boldsymbol{\Sigma}(\mathbf{1} - \boldsymbol{\rho}, \sqrt{\boldsymbol{\rho}}, \mathbf{a}),$$

so $\mathbf{R}$ *belongs to the same covariance family and is positive definite whenever* $\rho_i < 1$ *for all* i.

Proof. For $i \neq j$,

$$\sigma_i \sigma_j \sqrt{\rho_i \rho_j}\, K_{ij} = (\sigma_i \sqrt{\rho_i})(\sigma_j \sqrt{\rho_j}) K_{ij} = w_i w_j K_{ij} = \boldsymbol{\Sigma}_{ij},$$

where $\sigma_i \sqrt{\rho_i} = w_i$ follows from Eq. 16. On the diagonal, $\sigma_i^2 = d_i + w_i^2 = \boldsymbol{\Sigma}_{ii}$. Thus the stated factorization holds. Substituting $\mathbf{d} = \mathbf{1} - \boldsymbol{\rho}$ and $\mathbf{w} = \sqrt{\boldsymbol{\rho}}$ into Eq. 14 gives $\sqrt{\rho_i \rho_j} K_{ij}$ off the diagonal and $(1 - \rho_i) + \rho_i = 1$ on the diagonal. Hence $\mathbf{R}$ belongs to the family. Positive definiteness follows from $\mathbf{1} - \boldsymbol{\rho} > 0$ and the positive semidefiniteness of $K(\mathbf{a})$. □

Remark 1. *For architectures with BatchNorm, the running variances provide* $\boldsymbol{\sigma}$ *directly, so only* $(\boldsymbol{\rho}, \mathbf{a})$ *need to be fitted. The marginal scale is then reproduced by construction, leaving the LCM to model the correlation structure alone. This is the parameterization used in Section 3.*

G.3 Reduction to inner products

$\mathcal{L}_{\mathrm{F}}$ is quadratic in $\boldsymbol{\Sigma}(\boldsymbol{\vartheta})$ and linear in $\mathbf{T}$, so Eq. 15 passes straight through:

$$\mathcal{L}(\boldsymbol{\vartheta}) = \big\|\boldsymbol{\Sigma}(\boldsymbol{\vartheta})\big\|_{\mathrm{F}}^2 - 2\sum_{k=1}^{K} \pi_k \big\langle \boldsymbol{\Sigma}(\boldsymbol{\vartheta}), \boldsymbol{\Sigma}(\boldsymbol{\vartheta}_k)\big\rangle_{\mathrm{F}} + \underbrace{\|\mathbf{T}\|_{\mathrm{F}}^2}_{\text{const}}.$$

The merge therefore depends on the inputs only through K inner products between members of the family; no task data and no stored matrix are required.

G.4 Inner product of two LCMs

Proposition 2. *Let $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\mathbf{d}, \mathbf{w}, \mathbf{a})$ and $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}(\mathbf{d}', \mathbf{w}', \mathbf{a})$ share the same coordinate vector $\mathbf{a}$. Then*

$$\big\langle \boldsymbol{\Sigma}, \boldsymbol{\Sigma}' \big\rangle_{\mathrm{F}} = \mathbf{d}^\top \mathbf{d}' + \mathbf{d}^\top (\mathbf{w}')^2 + (\mathbf{d}')^\top \mathbf{w}^2 + (\mathbf{w} \odot \mathbf{w}')^\top K(2\mathbf{a})\, (\mathbf{w} \odot \mathbf{w}'). \tag{18}$$

Proof. Substituting the entrywise form of Eq. 14,

$$\langle \boldsymbol{\Sigma}, \boldsymbol{\Sigma}' \rangle_{\mathrm{F}} = \sum_{i,j} \big(d_i \delta_{ij} + w_i w_j K_{ij}\big)\big(d'_i \delta_{ij} + w'_i w'_j K_{ij}\big),$$

which expands into four sums. Using $\delta_{ij}\delta_{ij} = \delta_{ij}$ and $K_{ii} = 1$,

$$\sum_{i,j} d_i d'_i \delta_{ij} = \mathbf{d}^\top \mathbf{d}', \qquad \sum_{i,j} d_i \delta_{ij}\, w'_i w'_j K_{ij} = \mathbf{d}^\top (\mathbf{w}')^2,$$

$$\sum_{i,j} w_i w_j K_{ij}\, d'_i \delta_{ij} = (\mathbf{d}')^\top \mathbf{w}^2, \qquad \sum_{i,j} (w_i w'_i)(w_j w'_j) K_{ij}^2 = (\mathbf{w} \odot \mathbf{w}')^\top K(2\mathbf{a})\, (\mathbf{w} \odot \mathbf{w}'),$$

the last using $K_{ij}(\mathbf{a})^2 = K_{ij}(2\mathbf{a})$. □

Remark 2. *Setting $(\mathbf{d}', \mathbf{w}') = (\mathbf{d}, \mathbf{w})$ collapses Eq. 18 to $\|\mathbf{d}\|_2^2 + 2\,\mathbf{d}^\top \mathbf{w}^2 + (\mathbf{w}^2)^\top K(2\mathbf{a})(\mathbf{w}^2)$, the model term of Eq. 4, which is therefore the special case $\boldsymbol{\Sigma}' = \boldsymbol{\Sigma}$ of Prop. 2.*

Corollary 1 (Correlation form)**.** *In the parameterization of Prop. 1, i.e. with $\mathbf{d} = \mathbf{1} - \boldsymbol{\rho}$ and $\mathbf{w} = \sqrt{\boldsymbol{\rho}}$ (likewise for $\boldsymbol{\rho}'$),*

$$\big\langle \mathbf{R}(\boldsymbol{\rho}, \mathbf{a}),\, \mathbf{R}(\boldsymbol{\rho}', \mathbf{a}) \big\rangle_{\mathrm{F}} = D - \boldsymbol{\rho}^\top \boldsymbol{\rho}' + \mathbf{p}^\top K(2\mathbf{a})\, \mathbf{p}, \qquad \mathbf{p} = \sqrt{\boldsymbol{\rho} \odot \boldsymbol{\rho}'}.$$

Proof. The first three terms of Eq. 18 become $\sum_i [(1-\rho_i)(1-\rho'_i) + (1-\rho_i)\rho'_i + (1-\rho'_i)\rho_i] = \sum_i (1 - \rho_i \rho'_i) = D - \boldsymbol{\rho}^\top \boldsymbol{\rho}'$, and $\mathbf{w} \odot \mathbf{w}' = \sqrt{\boldsymbol{\rho} \odot \boldsymbol{\rho}'}$. □

G.5 The merge is exact on the marginals

The diagonal parameters admit a closed-form solution, which reduces the merge to a problem in $(\mathbf{w}, \mathbf{a})$ alone.

Proposition 3. *For fixed $(\mathbf{w}, \mathbf{a})$, the unconstrained minimiser of $\mathcal{L}_F$ over $\mathbf{d}$ is*

$$\mathbf{d}^\star = \mathrm{diag}(\mathbf{T}) - \mathbf{w}^2, \qquad \mathrm{diag}(\mathbf{T})_i = \sum_{k=1}^{K} \pi_k \big(d_{k,i} + w_{k,i}^2\big), \tag{19}$$

which is available in $\mathcal{O}(KD)$ from the component parameters alone. Under the constraint $\mathbf{d} \geq \varepsilon$ the minimiser is $\max(\mathrm{diag}(\mathbf{T}) - \mathbf{w}^2,\, \varepsilon)$ componentwise.

Proof. Only the i-th diagonal entry of $\boldsymbol{\Sigma}(\boldsymbol{\vartheta})$ depends on d_i, with $\partial \boldsymbol{\Sigma}_{ii} / \partial d_i = 1$, so $\partial \mathcal{L} / \partial d_i = 2(\boldsymbol{\Sigma}_{ii} - T_{ii}) = 2(d_i + w_i^2 K_{ii} - T_{ii}) = 2(d_i + w_i^2 - T_{ii})$. Setting this to zero gives Eq. 19; the objective is convex in $\mathbf{d}$, so the stationary point is the minimum, and the constrained case follows by projection onto a box. The expression for $\mathrm{diag}(\mathbf{T})$ follows from $K_{ii} = 1$ applied to each component of Eq. 15. □

Corollary 2. *At* $\mathbf{d} = \mathbf{d}^\star$,

$$\mathcal{L}_F(\mathbf{d}^\star, \mathbf{w}, \mathbf{a}) = \sum_{i \neq j} \big(w_i w_j K_{ij}(\mathbf{a}) - T_{ij}\big)^2. \tag{20}$$

Proof. By Prop. 3 the diagonal residual vanishes identically, so only off-diagonal terms remain, and off the diagonal $\mathbf{\Sigma}_{ij} = w_i w_j K_{ij}$. □

Remark 3. *Aggregation is performed in covariance space. Since* $\mathrm{diag}(\mathbf{T}) = \sum_k \pi_k \boldsymbol{\sigma}_k^2$ *while the off-diagonal entries of* $\mathbf{T}$ *mix the component scales, the correlation matrix of* $\mathbf{T}$ *is in general* not *the convex combination of the component correlations; the two coincide only when all* $\boldsymbol{\sigma}_k$ *agree. Prop. 3 accounts for this by reading* $\mathbf{d}^\star$ *off* $\mathrm{diag}(\mathbf{T})$*, so Eq. 17 reparameterizes the merged model rather than defining a separate merge.*

Two consequences follow. First, the merged model preserves the marginal variances of the aggregate *exactly*; any approximation introduced by the projection therefore affects only the correlation structure. Second, this provides a formal justification for the modelling choice in Section 3: the marginal scale is supplied by the network, while LCM models the correlation structure (Prop. 1). Indeed, Eq. 20 shows that the Frobenius merge naturally induces the same decomposition.

G.6 Distinct coordinates

Prop. 2 assumes shared coordinates. In the general case, each LCM has its own coordinates $\mathbf{a}$ and $\mathbf{a}_k$. For $\mathbf{p} = \mathbf{w} \odot \mathbf{w}_k$, the kernel–kernel term in the LCM inner product becomes

$$\sum_{i,j} p_i p_j \, e^{-|a_i - a_j|} \, e^{-|a_{k,i} - a_{k,j}|} = \sum_{i,j} p_i p_j \exp\big(-\|\mathbf{b}_i - \mathbf{b}_j\|_1\big), \qquad \mathbf{b}_i = (a_i, a_{k,i}) \in \mathbb{R}^2, \tag{21}$$

which corresponds to a Laplace kernel under the ℓ_1 metric on scattered points in the plane. The one-dimensional prefix-sum recursion therefore does not apply directly. Nevertheless, the kernel–kernel term, as well as the corresponding matrix–vector product, can still be evaluated in $\mathcal{O}(D \log D)$ time.

Assume that the a_i are distinct. Sorting the points by a_i and splitting the sum according to the sign of $a_i - a_j$ gives

$$\sum_{i,j} p_i p_j e^{-\|\mathbf{b}_i - \mathbf{b}_j\|_1} = \sum_i p_i^2 + 2 \sum_i p_i e^{-a_i} \sum_{j:\, a_j < a_i} p_j e^{a_j} e^{-|a_{k,i} - a_{k,j}|}. \tag{22}$$

The remaining absolute value can then be removed by splitting according to the sign of $a_{k,i} - a_{k,j}$:

$$\sum_{j:\, a_j < a_i} p_j e^{a_j} e^{-|a_{k,i} - a_{k,j}|} = e^{-a_{k,i}} \sum_{\substack{a_j < a_i \\ a_{k,j} \leq a_{k,i}}} p_j e^{a_j + a_{k,j}} + e^{a_{k,i}} \sum_{\substack{a_j < a_i \\ a_{k,j} > a_{k,i}}} p_j e^{a_j - a_{k,j}}. \tag{23}$$

These sums can be maintained while processing the indices in increasing order of a_i using two Fenwick trees indexed by the rank of $a_{k,i}$. One tree accumulates $p_j e^{a_j + a_{k,j}}$, while the other accumulates $p_j e^{a_j - a_{k,j}}$. Each query and update costs $\mathcal{O}(\log D)$, giving $\mathcal{O}(D \log D)$ time and $\mathcal{O}(D)$ memory after sorting. Processing the indices in both directions and accumulating the contributions for each index yields the full matrix–vector product $\big(K(\mathbf{a}) \odot K(\mathbf{a}_k)\big)\mathbf{x}$.

H Metrics Definition

We evaluate the performance of REMIX using two standard metrics in continual learning: *average accuracy* and *last task accuracy*. Let $a_{t,i}$ denote the classification accuracy on the test set of task i after the model has been trained up to task t (where $i \leq t$).

Average Accuracy. Average accuracy measures the overall performance of the model across all tasks observed up to a given incremental step t. It is defined as

$$\mathcal{A}_t = \frac{1}{t} \sum_{i=1}^{t} a_{t,i}.$$

This metric captures the balance between retaining previously learned knowledge and adapting to new tasks. When reporting overall performance, it is common to compute the *average incremental accuracy*, defined as the mean of $\mathcal{A}_t$ over all T incremental steps.

Last Task Accuracy. Last task accuracy evaluates the model after completing the full sequence of T tasks. It reflects the final performance across all learned classes and is defined as

$$\mathcal{A}_{\text{last}} = \mathcal{A}_T = \frac{1}{T}\sum_{i=1}^{T} a_{T,i}.$$

I Implementation Details

Benchmarks and Task Splits. To ensure a fair and direct comparison, we build REMIX on the official PMI implementation and retain its experimental protocol, including task divisions, class orderings, and evaluation settings. Each dataset is partitioned into a sequence of disjoint incremental tasks, where new classes are introduced over time without access to data from previous tasks.

For CIFAR-100, we use splits into 5, 10, and 20 tasks in the ResNet-32 setting, and a 10-task split (10 classes per task) in the CLIP-based setting. Tiny-ImageNet is partitioned analogously into 5, 10, and 20 tasks. For CUB-200, we adopt a 10-task split with 20 classes per task, following PMI.

All experiments are conducted in the DFCIL setting, where no real samples from previous tasks are retained. Instead, pseudo-samples are generated via model inversion using stored feature statistics. Thus, all experimental results use the same underlying implementation and task configurations as PMI, with the proposed covariance modeling as the primary change.

Baselines. We compare REMIX with three groups of approaches, distinguished by how information from previous tasks is retained.

Rehearsal-based methods retain real exemplars from previous tasks. iCaRL (Rebuffi et al., 2017) combines exemplar replay with knowledge distillation and a nearest-mean classifier; MEMO (Zhou et al., 2023) decouples generalized and specialized network blocks to reduce the memory cost of stored exemplars; PROOF (Zhou et al., 2025) adapts vision–language models through projection fusion over frozen CLIP embeddings; and CLAP4CLIP (Jha et al., 2024) fine-tunes CLIP with a probabilistic task-conditioned adapter. These methods do not satisfy the data-free constraint and are included to indicate the performance attainable when real images from previous tasks are available. The *upper bound* reports joint training on all classes simultaneously.

Exemplar-free methods without model inversion retain compact statistics of previously learned classes. SSRE (Zhu et al., 2022) reorganizes the network into expanded and preserved branches and replays stored prototypes, while PRAKA (Shi & Ye, 2023) augments stored prototypes and asymmetrically aggregates old and new knowledge. In the CLIP setting, we additionally report zero-shot CLIP (ZS-CLIP) (Radford et al., 2021) as a no-adaptation reference, together with parameter-efficient adaptation methods VPT (Jia et al., 2022), AttriCLIP (Wang et al., 2023), CODA-Prompt (Smith et al., 2023), and MoE-Adapter (Yu et al., 2024), which adapt a frozen pretrained backbone through prompts, attributes, or expert adapters.

Data-free methods based on model inversion constitute the direct comparison for REMIX. DeepInversion (Yin et al., 2020) synthesizes images by matching BatchNorm statistics; ABD (Smith et al., 2021) replaces the replay buffer with inverted samples; DCMI (Qiu et al., 2024) enforces consistency between inverted samples and the frozen model; R-DFCIL (Gao et al., 2022) adds relation-guided distillation; and PMI (Tong et al., 2025) performs sequential inversion using layer-specific feature statistics. Our implementation builds on PMI, making it the primary data-free baseline for comparison. We also report PMI w/o CFS, which disables contrastive feature selection. All inversion-based baselines use diagonal covariance to model their feature statistics; REMIX replaces this assumption with the proposed covariance model.

Grid Search and Optimal Hyperparameters. For experiments with the ResNet-32 backbone, we perform a grid search over the regularization weight α_{stat}, which controls the contribution of the statistical matching loss $\mathcal{L}_{\text{stat}}$ (Eq. 2), using the range $\{0.01, 0.05, 0.1, 0.25, 0.5, 1.0\}$. For CLIP-based experiments, we adopt a finer grid: $\{0.0001, 0.0005, 0.001, 0.002, 0.005, 0.01, 0.05\}$.

The LCM parameters are optimized using Adam with a learning rate of 0.01 for 200 epochs. Despite this number of epochs, optimization remains computationally efficient, as discussed in Appendix K, where we report training time in comparison to PMI.

We adopt the same task splits as defined in the PMI and R-DFCIL protocols. Table 3 reports the selected optimal values of α_{stat} for each dataset and number of tasks.

Table 3: Selected α_{stat} values for all experiments.

Architecture	Dataset	5 tasks	10 tasks	20 tasks
ResNet-32	CIFAR-100	0.05	0.5	0.5
ResNet-32	Tiny-ImageNet	0.1	0.25	0.05
CLIP	CIFAR-100	–	0.0001	–
CLIP	CUB-200	–	0.01	–

Code and Reproducibility. To ensure a fair comparison, we adopt the same software environment as the PMI codebase. The experiments are implemented in Python using `PyTorch` (v1.13.1) with `torchvision` (v0.14.1) and `timm` (v0.9.7) for model architectures. We rely on `NumPy` (v1.26.4), `SciPy` (v1.13.1), and `scikit-learn` (v1.6.1) for numerical computations, and `matplotlib` (v3.8.2) and `tensorboard` (v2.12.1) for visualization and logging. For CLIP-based experiments, we use `open-clip-torch` (v2.22.0).

All experiments were conducted on NVIDIA A100 (40GB) and RTX 4090 (24GB) GPUs. Full training pipelines, configuration files, and scripts for reproducing all reported results will be made publicly available in our GitHub repository.

J ALGORITHMS

Algorithm 1 evaluates products with the Laplace kernel without explicitly forming the kernel matrix. We use this operation to compute the quadratic form $\mathrm{QuadForm}(\mathbf{x}, \mathbf{a}) = \mathbf{x}^\top \mathrm{KernelMatVec}(\mathbf{a}, \mathbf{x})$. Algorithms 2 and 3 then use this operation to fit the LCM to task data and evaluate its exact likelihood during inversion, respectively. Algorithm 4 performs the continual merge, while Algorithm 5 specifies the complete procedure. Throughout, $\boldsymbol{\vartheta} = (\mathbf{d}, \mathbf{w}, \mathbf{a})$ denotes the LCM parameters for a single layer; the layer index is omitted, and all algorithms are applied independently to each layer.

Algorithm 1: KERNELMATVEC: $\mathbf{y} = K(\mathbf{a})\mathbf{x}$ in $\mathcal{O}(D \log D)$ time, $\mathcal{O}(D)$ memory

Input : vector $\mathbf{a} \in \mathbb{R}^D$, vector $\mathbf{x} \in \mathbb{R}^D$
Output : $\mathbf{y} = K(\mathbf{a})\mathbf{x}$, without forming $K(\mathbf{a})$

$\pi \leftarrow \mathrm{argsort}(\mathbf{a})$
$\tilde{\mathbf{a}} \leftarrow \mathbf{a}[\pi], \quad \tilde{\mathbf{x}} \leftarrow \mathbf{x}[\pi]$
$\phi_i \leftarrow \exp(\tilde{a}_i - \tilde{a}_{i+1}) \quad$ for $i = 1, \ldots, D-1$ `// adjacent decay`
$L_1 \leftarrow 0$
for $i \leftarrow 2$ **to** D **do**
  $L_i \leftarrow \phi_{i-1}(L_{i-1} + \tilde{x}_{i-1})$ `// prefix sums`
$R_D \leftarrow 0$
for $i \leftarrow D-1$ **to** 1 **do**
  $R_i \leftarrow \phi_i(R_{i+1} + \tilde{x}_{i+1})$ `// suffix sums`
return $(\mathbf{L} + \tilde{\mathbf{x}} + \mathbf{R})[\pi^{-1}]$

Algorithm 2: FITLCM: fit one task by the Frobenius objective

Input : centred activations $\{\mathbf{z}_i\}_{i=1}^N$, steps T, rate η, ε
Output : $\boldsymbol{\vartheta} = (\mathbf{d}, \mathbf{w}, \mathbf{a})$

$\mathbf{u} \leftarrow \mathbf{0}, \quad \mathbf{w} \leftarrow \frac{1}{2}\mathbf{1}, \quad \mathbf{a} \leftarrow \mathrm{linspace}(-1, 1, D)$
for $s \leftarrow 1$ **to** T **do**
  $\mathbf{d} \leftarrow \mathrm{softplus}(\mathbf{u}) + \varepsilon\mathbf{1}$ `// keeps` $\boldsymbol{\Sigma} \succ 0$
  $\mathcal{M} \leftarrow \|\mathbf{d}\|^2 + 2\,\mathbf{d}^\top \mathbf{w}^2 + \mathrm{QuadForm}(\mathbf{w}^2, 2\mathbf{a})$
  $\mathcal{E} \leftarrow \frac{1}{N}\sum_i \left(\mathbf{d}^\top \mathbf{z}_i^2 + \mathrm{QuadForm}(\mathbf{w} \odot \mathbf{z}_i, \mathbf{a})\right)$
  $(\mathbf{u}, \mathbf{w}, \mathbf{a}) \leftarrow (\mathbf{u}, \mathbf{w}, \mathbf{a}) - \eta\,\nabla(\mathcal{M} - 2\mathcal{E})$ `// Adam in practice`
return $(\mathrm{softplus}(\mathbf{u}) + \varepsilon\mathbf{1}, \mathbf{w}, \mathbf{a})$

Algorithm 3: EXACTNLL: exact Gaussian NLL under the LCM

Input : $\mathbf{x}$, mean $\boldsymbol{\mu}$, parameters $(\mathbf{d}, \mathbf{w}, \mathbf{a})$
Output : $-\log \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}(\boldsymbol{\vartheta}))$, exactly

sort $\mathbf{x}, \boldsymbol{\mu}, \mathbf{d}, \mathbf{w}, \mathbf{a}$ by $\mathbf{a}$
$\phi_1 \leftarrow 0$, $q_1 \leftarrow 1$ // gives $z_1 \sim \mathcal{N}(0,1)$
$\phi_k \leftarrow \exp(a_{k-1} - a_k)$, $q_k \leftarrow 1 - \phi_k^2$ for $k > 1$
$m \leftarrow 0$, $P \leftarrow 1$, $\ell \leftarrow 0$
for $k \leftarrow 1$ **to** D **do**
  $m^- \leftarrow \phi_k m, \quad P^- \leftarrow \phi_k^2 P + q_k$ // predict
  $v \leftarrow (x_k - \mu_k) - w_k m^-, \quad S \leftarrow w_k^2 P^- + d_k$
  $\ell \leftarrow \ell + \log(2\pi S) + v^2/S$
  $G \leftarrow P^- w_k / S$
  $m \leftarrow m^- + Gv, \quad P \leftarrow (1 - G w_k) P^-$ // update
return $\ell/2$

Algorithm 4: MERGELCM: project $\alpha\boldsymbol{\Sigma}(\boldsymbol{\vartheta}_1) + \beta\boldsymbol{\Sigma}(\boldsymbol{\vartheta}_2)$ onto the LCM family

Input : $\boldsymbol{\vartheta}_1, \boldsymbol{\vartheta}_2$, weights $\alpha, \beta = 1 - \alpha$, steps T, rate η, ε
Output : $\boldsymbol{\vartheta}^\star$, with $\mathbf{T}$ never formed

Function $\mathrm{Inner}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}')$**:**
  $c \leftarrow \mathbf{d}^\top \mathbf{d}' + \mathbf{d}^\top (\mathbf{w}')^2 + (\mathbf{d}')^\top \mathbf{w}^2$
  if $\mathbf{a} = \mathbf{a}'$ **then**
    return $c + \mathrm{QuadForm}(\mathbf{w} \odot \mathbf{w}', 2\mathbf{a})$ // Prop. 1
  else
    $\mathbf{p} \leftarrow \mathbf{w} \odot \mathbf{w}', \quad \mathbf{b}_i \leftarrow (a_i, a_i') \in \mathbb{R}^2$
    return $c + \sum_{i,j} p_i p_j \, e^{-\|\mathbf{b}_i - \mathbf{b}_j\|_1}$ // $\mathcal{O}(D \log D)$, App. G
$\mathbf{t} \leftarrow \alpha(\mathbf{d}_1 + \mathbf{w}_1^2) + \beta(\mathbf{d}_2 + \mathbf{w}_2^2)$ // $= \mathrm{diag}(\mathbf{T})$
$(\mathbf{w}, \mathbf{a}) \leftarrow (\mathbf{w}_1, \mathbf{a}_1)$
for $s \leftarrow 1$ **to** T **do**
  $\mathbf{d} \leftarrow \max(\mathbf{t} - \mathbf{w}^2, \varepsilon\mathbf{1})$ // optimal $\mathbf{d}$, Prop. 2
  $\mathcal{L} \leftarrow \mathrm{Inner}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}) - 2\alpha \,\mathrm{Inner}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}_1) - 2\beta \,\mathrm{Inner}(\boldsymbol{\vartheta}, \boldsymbol{\vartheta}_2)$
  $(\mathbf{w}, \mathbf{a}) \leftarrow (\mathbf{w}, \mathbf{a}) - \eta \nabla_{(\mathbf{w},\mathbf{a})} \mathcal{L}$
return $(\max(\mathbf{t} - \mathbf{w}^2, \varepsilon\mathbf{1}), \mathbf{w}, \mathbf{a})$

Algorithm 5: REMIX: one incremental step, task $t \to t+1$

Input : model $(\boldsymbol{\theta}_{1:t}, \boldsymbol{\phi}_{1:t})$, stored $\{\boldsymbol{\mu}^{(l)}_{1:t}, \boldsymbol{\vartheta}^{(l)}_{1:t}\}_l$, data $\mathcal{D}_{t+1}$
Output : updated model and per-layer statistics

Stage 1 — fit the current task
foreach *modelled layer* l **do**
  $\boldsymbol{\vartheta}^{(l)}_{t+1} \leftarrow \mathrm{FitLCM}(\text{activations of } \mathcal{D}_{t+1} \text{ at layer } l)$ `// Alg. 2`

Stage 2 — merge into the running estimate
$\alpha \leftarrow |\mathcal{Y}_{1:t}|/|\mathcal{Y}_{1:t+1}|, \quad \beta \leftarrow 1-\alpha$
foreach *modelled layer* l **do**
  $\boldsymbol{\vartheta}^{(l)}_{1:t+1} \leftarrow \mathrm{MergeLCM}(\boldsymbol{\vartheta}^{(l)}_{1:t}, \boldsymbol{\vartheta}^{(l)}_{t+1}, \alpha, \beta)$ `// Alg. 4`
  $\boldsymbol{\mu}^{(l)}_{1:t+1} \leftarrow \alpha\boldsymbol{\mu}^{(l)}_{1:t} + \beta\boldsymbol{\mu}^{(l)}_{t+1}$

Stage 3 — synthesise replay samples
$\mathcal{X}_{\text{old}} \leftarrow \emptyset$
foreach *target class* $y \in \mathcal{Y}_{1:t}$ **do**
  foreach *block* l*, deepest first* **do**
    minimise $\alpha_{\text{match}}\mathcal{L}_{\text{match}} + \alpha_{\text{stat}}\,\mathrm{ExactNLL}(\hat{\mathbf{h}}_l) + \alpha_{\text{in}}\mathcal{L}_{\text{in}}$ over $\hat{\mathbf{h}}_l$ `// Alg. 3`
  $\mathcal{X}_{\text{old}} \leftarrow \mathcal{X}_{\text{old}} \cup \mathrm{CFS}(\hat{\mathbf{h}}_0)$

Stage 4 — train on real and synthetic data
$(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow \arg\min \mathcal{L}_{\text{rrl}}$ on $\mathcal{D}_{t+1} \cup \mathcal{X}_{\text{old}}$, distilling from $(\boldsymbol{\theta}_{1:t}, \boldsymbol{\phi}_{1:t})$
$\boldsymbol{\phi} \leftarrow \arg\min \mathcal{L}_{\text{gce}}$ on a class-balanced mixture, $\boldsymbol{\theta}$ frozen
return $(\boldsymbol{\theta}, \boldsymbol{\phi})$, $\{\boldsymbol{\mu}^{(l)}_{1:t+1}, \boldsymbol{\vartheta}^{(l)}_{1:t+1}\}_l$

In Algorithm 3, the quantities P, S, and G depend only on $(\mathbf{d}, \mathbf{w}, \mathbf{a})$, not on $\mathbf{x}$. Thus, for a batch, the variance recursion is computed once, while only the scalar mean m is propagated for each sample, giving $\mathcal{O}(ND)$ cost for N observations after a single $\mathcal{O}(D \log D)$ sort. In Algorithm 4, $\mathbf{d}$ is eliminated analytically at each step, ensuring that the merged model exactly matches $\mathrm{diag}(\mathbf{T})$ and leaving only the off-diagonal residual to optimize. Stages 1 and 2 of Algorithm 5 constitute our contribution, while Stages 3 and 4 follow PMI (Tong et al., 2025), with the diagonal statistical term replaced by the exact LCM likelihood. After task $t+1$, the retained state is $\{\boldsymbol{\mu}^{(l)}, \boldsymbol{\vartheta}^{(l)}\}_l$, requiring only $4D$ floats per layer, independent of the number of tasks.

K Additional Experimental Results

K.1 ResNet-32 Backbone

Table 1 with Standard Deviations. Table 4 reports average incremental accuracy together with standard deviations (over five runs), corresponding to Table 1 in the main text.

Table 4: Average incremental accuracy (%) with standard deviations on CIFAR-100 and Tiny-ImageNet using a ResNet-32 backbone. Results are averaged over five runs.

Method	Model inversion	CIFAR-100			Tiny-ImageNet		
		5 task	10 task	20 task	5 task	10 task	20 task
Upper bound	✗	70.59±0.14	70.59±0.14	70.59±0.14	55.25±0.41	55.25±0.41	55.25±0.41
SSRE	✗	30.39±0.04	17.77±0.14	10.97±0.48	26.22±0.32	18.58±0.43	10.09±0.34
PRAKA	✗	37.74±0.48	26.72±0.38	16.32±0.66	31.69±0.20	22.37±0.14	13.62±0.42
DeepInversion	✓	20.48±1.11	11.26±0.46	5.63±0.10	-	-	-
ABD	✓	48.84±0.33	36.75±0.45	24.40±0.60	30.83±0.46	23.17±0.45	14.61±0.47
DCMI	✓	41.05±0.67	27.70±1.07	18.09±0.85	35.78±0.37	25.89±0.17	17.03±0.08
R-DFCIL	✓	49.87±0.45	41.80±0.24	31.54±0.54	35.33±0.02	29.05±0.28	24.85±0.16
PMI w/o CFS	✓	52.05±0.02	43.23±0.28	32.23±0.42	37.65±0.24	32.09±0.20	25.51±0.56
PMI	✓	**52.38±0.53**	**43.90±0.35**	**32.60±0.29**	**37.90±0.10**	**32.43±0.09**	**25.67±0.71**
REMIX	✓	**52.94 ± 0.07**	**45.38 ± 0.13**	**33.11 ± 0.28**	**38.64 ± 0.35**	**33.39 ± 0.11**	**27.11 ± 0.31**

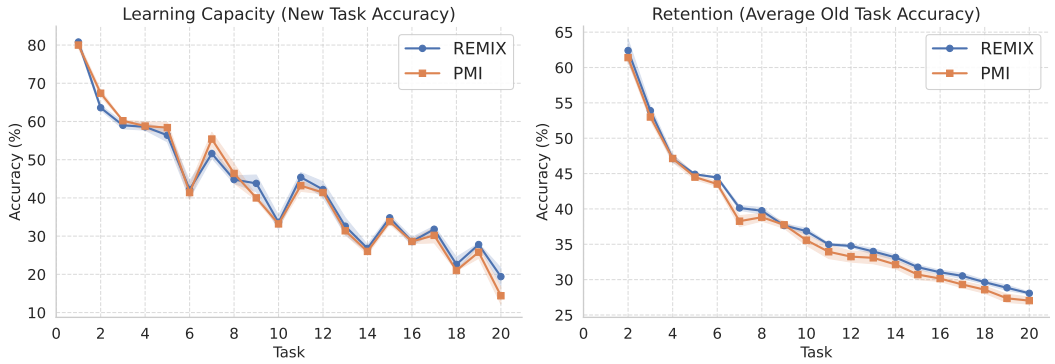

Figure 7: Task-wise performance on Tiny-ImageNet with a ResNet-32 backbone. Left: accuracy on the current task (learning capacity). Right: average accuracy on previously learned tasks (retention). Solid lines denote mean over five runs, with shaded regions indicating standard deviation. REMIX improves retention while maintaining comparable learning capacity.

Tiny-ImageNet. We further evaluate REMIX on Tiny-ImageNet using a ResNet-32 backbone. Fig. 7 presents task-wise performance, where the left plot shows accuracy on the current task (learning capacity) and the right plot shows the average accuracy on previously learned tasks (retention), both averaged over five runs.

Consistent with the observations on CIFAR-100, REMIX maintains comparable performance to PMI on newly introduced tasks, indicating that the ability to acquire new knowledge is preserved. At the same time, REMIX achieves consistently higher accuracy on previously learned tasks, demonstrating improved retention. This confirms that modeling structured feature dependencies effectively reduces forgetting without compromising learning capacity.

K.2 Qualitative Analysis of Generated Samples

Implementation Details and Architectural Adaptations. We generate synthetic samples via model inversion applied to pretrained backbones (ResNet-34 and ViT-B/16). Starting from random noise, input pixels are optimized using Adam with a cosine annealing schedule. We adopt a multi-resolution pyramid (e.g., $112\times112 \rightarrow 224\times224$) to progressively refine global structure. During the early stages, strong augmentations (random flips, affine transformations, spatial jitter) are applied to suppress high-frequency artifacts.

For this experiment, we adopt the DeepInversion optimization pipeline as the base inversion procedure. While PMI introduces a per-layer inversion scheme, in practice, it produces samples through a similar objective and optimization process, differing primarily in convergence speed. We therefore use DeepInversion to ensure a consistent and stable optimization setup, while isolating the effect of our contribution to feature distribution modeling. Importantly, all methods share identical optimization procedures, augmentations, and hyperparameters, and differ only in how feature statistics are modeled.

We directly optimize pixels rather than training a parametric generator; full loss definitions are provided in Appendix E.

Architectural Adaptation of LCM. The LCM is adapted to match the structure of each backbone.

ResNet-34. We extract intermediate feature maps from residual stages (`layer1`–`layer4`), obtaining multi-scale representations of shape (B, C, H, W). These are flattened into high-dimensional vectors ($D = C\times H\times W$, up to $\sim 10^6$), and LCM is fitted directly to model dependencies across both spatial and channel dimensions. During inversion, we evaluate the exact $\mathcal{O}(D\log D)$ NLL of generated features at each layer and use it as a density-matching prior, together with BatchNorm statistics alignment and classification loss.

ViT-B/16. Features are represented as sequences of patch tokens from transformer blocks. We discard the `[CLS]` token and retain only spatial tokens, which are reshaped into vectors for distribution

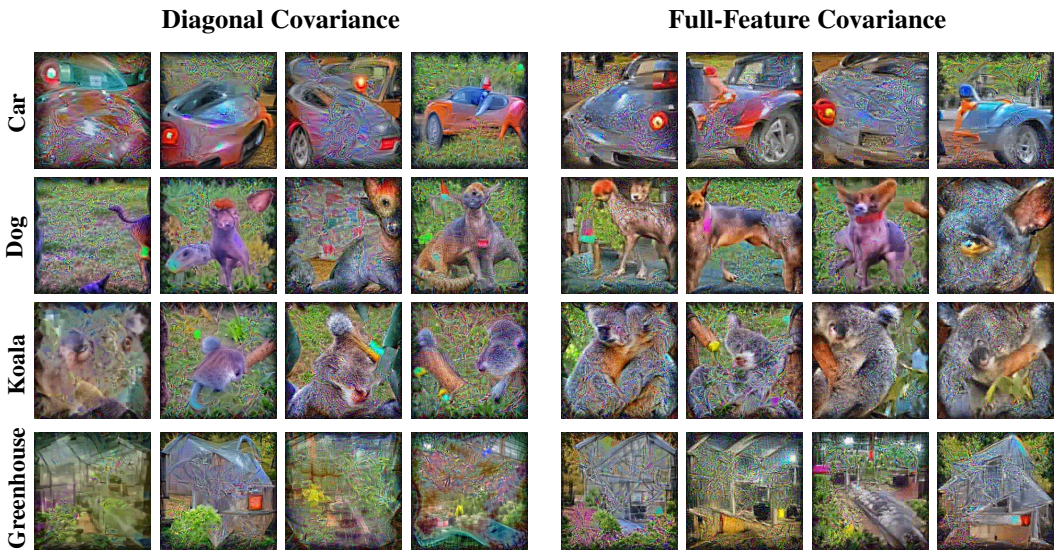

Figure 8: Qualitative comparison of generated samples using the ViT backbone. Left: prior approaches (e.g., PMI, DeepInversion) relying on diagonal covariance, which neglect dependencies across feature dimensions. Right: REMIX, which efficiently models correlations over the full feature map (up to millions of elements) while remaining computationally tractable. This leads to samples with improved structural coherence, fewer artifacts, and stronger semantic consistency.

modeling. LCM is applied to these representations to capture inter-patch dependencies during inversion. Since ViTs use Layer Normalization instead of BatchNorm, we match empirical means and variances computed over both batch and sequence dimensions at each block. This stabilizes global statistics while LCM enforces structured feature correlations.

The complete generation pipeline and implementation details are provided in the accompanying codebase.

Qualitative Analysis of Generated Samples. Figures 8 and 9 present qualitative comparisons of synthetic samples generated using ViT-B/16 and ResNet-34 backbones, respectively. Across both architectures, methods based on diagonal covariance assumptions produce samples with weaker structural consistency, visible artifacts, and less coherent semantics. In contrast, REMIX generates samples with clearer object boundaries, more consistent textures, and stronger semantic alignment across all categories. These improvements are particularly noticeable for fine-grained categories such as *koala*, where local textures and object structure are better preserved, as well as for scene-level classes such as *greenhouse*, where REMIX produces more globally coherent layouts. Overall, the results suggest that modeling structured feature dependencies leads to substantially more coherent synthetic representations across both convolutional and transformer-based architectures.

K.3 Effect of Covariance Modeling

We analyze the representational capacity of the proposed LCM on the ViT-B/16 backbone by evaluating the log-likelihood of feature distributions across transformer blocks. Specifically, during LCM parameter fitting, we compare the total log-likelihood obtained under the proposed covariance parameterization with a diagonal Gaussian baseline trained via maximum likelihood estimation (MLE).

To focus on spatial representations, we discard the `[CLS]` token and aggregate patch embeddings through average pooling, resulting in a $D = 768$ feature vector for each image at every transformer block. Empirically, this aggregation produces more stable and accurate density estimates than directly modeling token-level representations, unlike in the ResNet-34 setting where full spatial feature maps are modeled explicitly. The reasons for this behavior are discussed below.

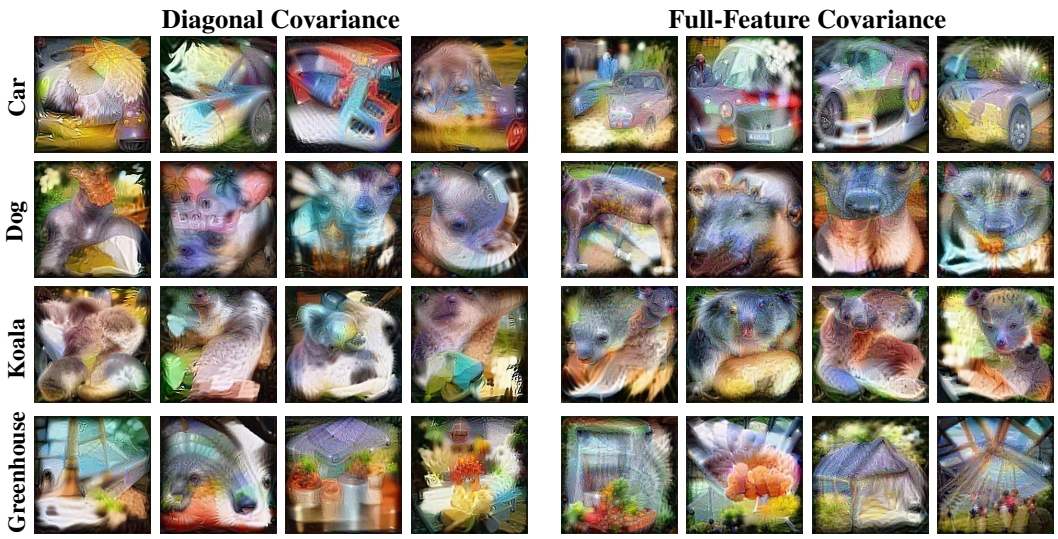

Figure 9: Qualitative comparison of generated samples using the ResNet-34 backbone. Left: prior methods based on diagonal covariance estimation, which ignore dependencies across feature dimensions. Right: REMIX, which models correlations over the full feature map (up to millions of elements) while remaining computationally tractable.

Fig. 10 shows a clear dependence on representation depth. In early blocks (0–4), the diagonal model achieves higher likelihood than LCM. This aligns with the structure of ViTs, where initial layers process non-overlapping patches, producing features that are largely local and weakly correlated. In this regime, modeling full covariance offers limited benefit and can introduce unnecessary complexity.

From intermediate layers onward (around Block 5), LCM begins to outperform the diagonal baseline, with the gap increasing in deeper layers (8–11). At these stages, self-attention aggregates information across patches, yielding more structured and correlated representations. The improved likelihood indicates that LCM effectively captures these dependencies, providing a more accurate model of the feature distribution.

K.4 Computational Efficiency

We evaluate the computational efficiency of REMIX in comparison to PMI. As shown in Tables 6 and 7, REMIX consistently improves performance while maintaining comparable training time and GPU memory usage across both ResNet-32 and ViT backbones.

The efficiency of our method stems from the structured LCM parameterization, which enables explicit covariance modeling while remaining computationally tractable. Naively evaluating the Gaussian NLL with a dense covariance matrix requires $\mathcal{O}(D^2)$ memory and $\mathcal{O}(D^3)$ computation due to inversion and log-determinant operations, making full-covariance modeling impractical for modern deep representations. In contrast, our formulation imposes a 1D conditional structure that enables exact likelihood evaluation in $\mathcal{O}(D \log D)$ time.

Moderate-Dimensional Regime (Channel/Token Level). When modeling correlations over channel-wise features (ResNet) or token embeddings (ViT), the dimensionality remains moderate. In this setting, we fit the LCM to explicitly model the covariance structure of features, i.e., the underlying density of the representation space.

Although we store only the compact LCM parameters, the corresponding covariance matrix can be reconstructed when needed. As its size is manageable, it can be explicitly materialized, enabling direct optimization of covariance matching objectives. Specifically, with $\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{v}_i \mathbf{v}_i^\top$, we can form $\mathbf{\Sigma}$ and directly minimize $\|\mathbf{\Sigma} - \hat{\mathbf{\Sigma}}\|_{\mathrm{F}}^2$, instead of relying on the decomposed Frobenius formulation (Eq. 4).

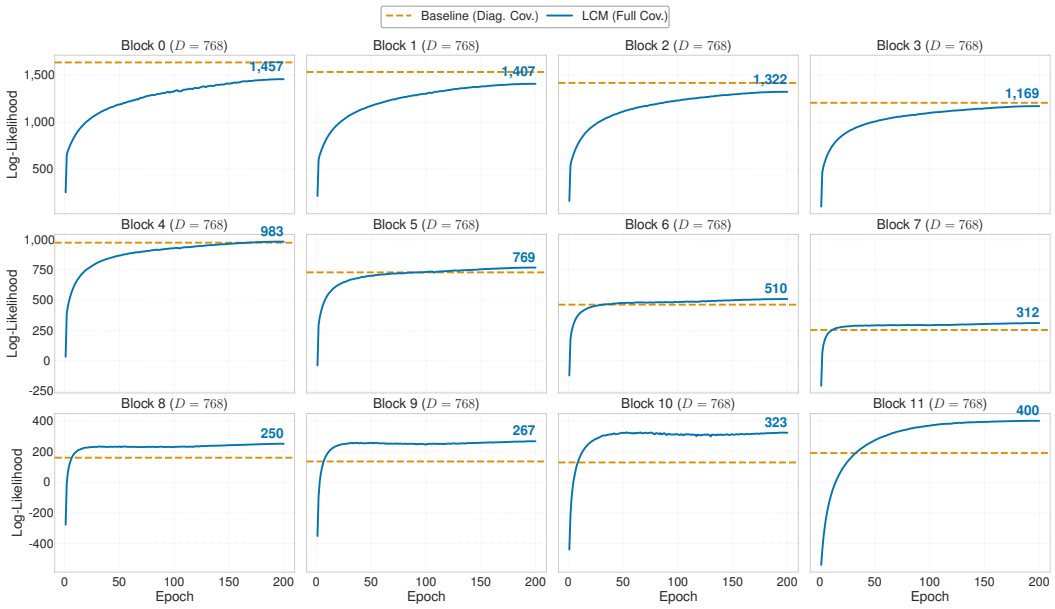

Figure 10: Log-likelihood comparison between diagonal and full-feature covariance models across ViT-B/16 blocks. While diagonal covariance performs better in early layers with weakly correlated patch features, LCM achieves higher likelihood in deeper layers where self-attention induces structured dependencies.

Crucially, once the feature distribution is modeled, we can directly optimize the exact Gaussian NLL during model inversion. This allows synthetic samples to be generated by maximizing likelihood under the learned full-covariance model.

High-Dimensional Regime (Full Feature Maps). When extending covariance modeling to full spatial feature maps, the dimensionality grows to $D = C \times H \times W$, making naive covariance estimation completely intractable. As shown in Table 5, a dense covariance would require up to 199.30 GB of memory for a ResNet-34 backbone.

In contrast, the LCM parameterization reduces this requirement to only 5.74 MB, achieving a $35{,}541\times$ reduction. Importantly, while dense methods cannot even represent the covariance in this regime, our structured formulation allows us to both represent and use it.

In this setting, the covariance matrix is never explicitly materialized due to its size. Instead, we leverage the LCM structure to compute the exact Gaussian NLL directly in $\mathcal{O}(D \log D)$ time (as presented in Appendix D). When covariance matching is required, we employ a Frobenius objective that is evaluated implicitly through the LCM using Eq. 4, without constructing dense matrices.

Summary. Overall, REMIX makes full-covariance modeling of deep feature distributions practical. It supports explicit optimization in moderate dimensions and remains scalable in high-dimensional regimes by avoiding dense matrix representations while preserving exact likelihood computation. This enables model inversion with full-feature dependencies, which is otherwise computationally infeasible.

Table 5: Memory and computational cost comparison for full spatial feature map covariance estimation on ResNet-34. By avoiding $\mathcal{O}(D^2)$ dense matrix allocations, our LCM formulation reduces the total memory footprint from 199.30 GB to 5.74 MB, achieving a $35{,}541\times$ efficiency gain.

Layer	Dimension (D)	LCM Mem.	Dense Mem.	Savings
Layer 1	200,704	**3.06 MB**	150.06 GB	50,176×
Layer 2	100,352	**1.53** MB	37.52 GB	25,088×
Layer 3	50,176	**0.77 MB**	9.38 GB	12,544×
Layer 4	25,088	**0.38 MB**	2.34 GB	6,272×
Total	376,320	**5.74 MB**	199.30 GB	35,541×

Table 6: Comparison with PMI on CIFAR-100 (5 tasks, 20 classes per task) using a ResNet-32 backbone. REMIX achieves higher average incremental accuracy with comparable time and memory usage.

Method	Avg.	Time	Total GPU Mem. (GB)
PMI	52.38±0.53	**1h 27m**	**7.28**
REMIX	**52.94±0.07**	1h 37m	**7.28**

Table 7: Comparison with PMI on CIFAR-100 (10 tasks, 10 classes per task) using a ViT backbone with MoE-Adapter. REMIX improves performance with negligible computational overhead.

Method	Avg.	Time	Total GPU Mem. (GB)
PMI	88.35	**3h 33m**	**7.39**
REMIX	**88.47**	3h 42m	**7.39**