CONTINUAL LEARNING TRAJECTORY AS MEMORY FOR REPLAY-FREE CONSOLIDATION IN MALICIOUS CODE ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Malware classifiers must continually adapt as new threats emerge, but sequential updates can overwrite knowledge of earlier malware families, leading to catastrophic forgetting. Continual learning (CL) methods address this problem, but most treat past knowledge as something that must be explicitly stored or reconstructed. In malware analysis, these approaches are undesirable because of storage and computational costs, data-access policies, and security concerns. This leads to a fundamental question: *Can the CL trajectory itself act as a form of memory?*

We study this question through the geometry of CL. Our experiments reveal a clear progression in linear connectivity—independently trained models are separated by high-loss regions; shared initialization introduces partial connectivity; and warm-start CL strengthens this connectivity further by combining common parameter ancestry with a continuous optimization trajectory. Based on this finding, we propose FreeMOCA, a replay-free CL method that recursively consolidates consecutive model states through linear parameter interpolation to reinforce favorable geometry across tasks. Unlike approaches that rely on replay, optimize nonlinear connectivity, or align and repair models before merging, FreeMOCA exploits connectivity already present in the learning trajectory and strengthens it through recursive consolidation. We evaluate FreeMOCA in a class-incremental learning setting on large-scale Windows and Android malware benchmarks against 10 CL baselines. FreeMOCA achieves average accuracies of 65.2% on EMBER-Class and 63.7% on AZ-Class, outperforming the strongest evaluated standard CL baselines by 2.1 and 8.8 percentage points, respectively, while reducing forgetting to 0.006 and 0.000. It requires only 15.98 MB of constant cross-task memory and averages 38.05 s of training time per task. Code is available at `https://anonymous.4open.science/r/FreeMOCA/README.md`.

1 INTRODUCTION

Machine learning (ML) has become a standard tool for malware analysis, but malware classifiers cannot remain static. New families, variants, and behaviors continually emerge while previously observed threats remain active. Updating a detector therefore creates a basic trade-off. Retraining on all past and new samples can preserve earlier knowledge, but adds storage and computational costs (Cho et al., 2026) and may be restricted by data-access policies or sample-sharing constraints (He et al., 2026). Training only on newly observed malware avoids these costs, but can overwrite previously learned behavior and lead to catastrophic forgetting (CF) (French, 1999; Rahman et al., 2022; Park et al., 2025; Rahman et al., 2025). Continual learning (CL) methods address this problem through replay (Park et al., 2025; Rahman et al., 2025; Park et al., 2026), regularization (Kirkpatrick et al., 2017; Zenke et al., 2017), or model expansion (Rusu et al., 2016; Yoon et al., 2017). These approaches largely treat past knowledge as something that must be transferred through an explicit mechanism: stored or reconstructed examples, auxiliary constraints, or additional parameters, each carrying a cost in memory, computation, or adaptation.

This motivates us to ask: *Can the continual learning trajectory itself act as a form of memory?* Recent CL methods explore parameter space for an answer (Kozal et al., 2024; Qiu et al., 2025; Li et al., 2025; Cho et al., 2026; He et al., 2026). WSC retains an exemplar memory (Cho et al.,

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.

2026), while RETROFIT combines low-rank updates, learned masks, and confidence-guided consolidation (He et al., 2026). CLeWI (Kozal et al., 2024) linearly interpolates consecutive CL states after replay-based neuron alignment and BatchNorm re-estimation. A linear interpolation can always be formed between two parameter vectors, but good endpoint solutions do not imply that the intermediate models also have low loss. Earlier mode-connectivity studies further suggest that such a path cannot simply be assumed for consecutive CL solutions.

Frankle et al. (2020) showed that solutions sharing part of their optimization history can become linearly connected, while Mirzadeh et al. (2021) found that shared initialization causes connectivity between continual and joint solutions. Wen et al. (2023) observed a strong ridge at the first class-incremental transition but more favorable straight-path regions later, suggesting that connectivity may evolve along the continual sequence. CLeWI (Kozal et al., 2024), however, interpolates consecutive CL states without establishing the existence of a low-loss linear path. It also applies replay-based neuron alignment motivated by the permutation ambiguity of independently initialized and trained networks. Consecutive warm-started states differ: the later model is obtained by directly updating the earlier one, so shared parameters keep their identity throughout optimization, even if their functional roles drift (Ainsworth et al., 2023).

Surprisingly, even these parameter-space approaches end up recovering past knowledge through replay samples. This suggests a more specific question: *when can consecutive CL solutions become connected by a low-loss linear path, and does this connectivity tend to strengthen as the trajectory develops?* CL does not produce a collection of independently trained models; it produces a parameter trajectory in which each task begins from a state shaped by those before it.

We separate two properties that are normally entangled in CL: shared parameter ancestry and continuous optimization history. Our experiments reveal that independently initialized solutions have the largest interpolation barriers. Training from a common initialization substantially lowers these barriers even when the models are optimized separately, showing that shared ancestry already provides a basis for connectivity (Mirzadeh et al., 2021). Consecutive warm-started states reduce the barriers further, indicating that continuous optimization can strengthen this basis (Frankle et al., 2020) when successive tasks reuse enough of the learned representation rather than forcing a complete reorganization. The largest barriers occur early; as warm-start training proceeds, consecutive states tend to become more linearly connected even though forgetting remains present. Thus, common initialization explains only part of the observed geometry: the continual trajectory itself develops stronger local connectivity.

FreeMOCA builds on this emerging structure through recursive consolidation. Rather than initializing each new task from the latest task-specific solution alone, it starts from a consolidated state obtained by interpolating the recent trajectory. Because this state already contains earlier consolidations, previous updates are carried forward recursively instead of being replaced by the newest endpoint. Figure 1 illustrates this progression from independent to shared initialization, warm-started CL, and our recursive consolidation. Our results show that FreeMOCA retains its consolidated states within a particularly low-barrier portion of the local trajectory, while maintaining stable activation statistics during interpolation. This matters because low-barrier connectivity is not guaranteed by warm-starting alone: large task shifts can still move optimization into less compatible regions. Rather than repairing models after training or explicitly optimizing a nonlinear path, FreeMOCA exploits the favorable geometry as it emerges and carries it forward through recursive interpolation. In this sense, FreeMOCA uses the continual trajectory itself as a lightweight source of memory.

The main contributions of this work are summarized as follows.

- We conduct systematic comparisons across independently initialized, shared-initialized, and warm-started CL solutions, showing that shared parameter ancestry establishes a basis for linear connectivity, while continuous optimization strengthens it further across consecutive tasks.
- Building on this emerging connectivity, we introduce FreeMOCA with recursive consolidation. Rather than starting each task from the latest solution alone, FreeMOCA carries forward a consolidated interpolation of the recent training trajectory.
- We evaluate FreeMOCA on the EMBER-Class and AZ-Class malware benchmarks against ten CL baselines. It achieves 65.2% and 63.7% Average Accuracy, outperforming the strongest evaluated baselines by 2.1 and 8.8 percentage points, with forgetting of 0.006 and 0.000. FreeMOCA also maintains a constant 15.98 MB cross-task memory footprint and requires 38.05 s per task.

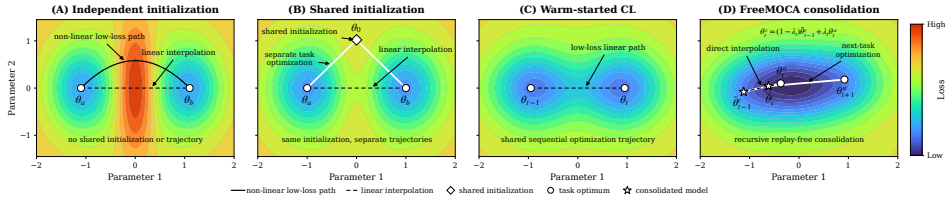


Figure 1: **Overview of FreeMOCA's connectivity motivation and consolidation mechanism.** (A) Independently initialized task solutions may require a non-linear low-loss path, with direct interpolation crossing a high-loss barrier. (B) Shared initialization reduces this mismatch but still yields separate optimization trajectories. (C) Warm-started CL produces consecutive solutions with stronger linear connectivity. (D) FreeMOCA exploits this property by recursively interpolating consecutive warm-started states, enabling replay-free consolidation.

2 Related Work

Continual learning (CL) methods address catastrophic forgetting (CF) through replay, regularization, or parameter isolation (Rahman et al., 2025; 2022; Park et al., 2025; Zenke et al., 2017; Kirkpatrick et al., 2017; Van de Ven et al., 2020; Shin et al., 2017; van de Ven & Tolias, 2018; Wang et al., 2023; Gao et al., 2023; Zhang et al., 2023). Replay methods preserve earlier-task information through stored exemplars (Rahman et al., 2025; Rebuffi et al., 2017) or synthetic samples generated by an auxiliary generator (Van de Ven et al., 2020; Park et al., 2025; Shin et al., 2017; van de Ven & Tolias, 2018), at the cost of persistent storage, additional training, or both. Regularization methods constrain how much current-task training can alter previously important parameters or predictions (Zenke et al., 2017; Kirkpatrick et al., 2017; Li & Hoiem, 2017), whereas parameter-isolation methods allocate or protect task-specific capacity. While these approaches do not require explicit replay, they may reduce plasticity or introduce additional parameters, masks, architectural changes, or task identity (Wang et al., 2023; Gao et al., 2023; Zhang et al., 2023). FreeMOCA differs in separating current-task adaptation from knowledge consolidation. It first adapts the model using current-task data and then consolidates the adapted and previous states in parameter space. Mean Teacher (Tarvainen & Valpola, 2017) and SWA (Izmailov et al., 2018) average weights along a single training trajectory, while model soups (Wortsman et al., 2022) combine independently fine-tuned models that share a pretrained initialization. CLS-ER (Arani et al., 2022) maintains slow and fast EMA memories with replay, temporal ensembles use EMA models to improve online stability (Soutif-Cormerais et al., 2023), and momentum-based interpolation similarly maintains fast and slow models (Stojanovski et al., 2022). Unlike within-task averaging methods, FreeMOCA recursively consolidates consecutive task-adapted states, with each consolidated state initializing the next task and thereby becoming part of the continual optimization trajectory.

Continual Learning for Malware Analysis. Malware distributions evolve continuously, and applying general CL methods to malware classification can lead to substantial CF (Rahman et al., 2022). MalCL uses generative replay (Park et al., 2025), TabCL adapts replay to tabular malware features through tabular-aware generation and feature compression (Park et al., 2026), and MADAR preserves historical samples through distribution-aware exemplar management (Rahman et al., 2025). Ghiani et al. (2026) address update-induced security regressions through prediction-consistency regularization, while RETROFIT (He et al., 2026) consolidates previous and adapted models using low-rank updates, sparse masks, and confidence-guided arbitration.

3 FreeMOCA

We study replay-free class-incremental learning (Class-IL), where task t introduces previously unseen classes $\mathcal{Y}_t$ and the model predicts over all classes observed so far, $\mathcal{Y}_{1:t} = \bigcup_{k=1}^{t} \mathcal{Y}_k$, without task identity at inference time. Let $\mathcal{D}_t$ denote the data of task t. When learning task t, only $\mathcal{D}_t$ is available; $\mathcal{D}_1, \ldots, \mathcal{D}_{t-1}$ are not retained.

The central idea of FreeMOCA is to use the sequence of states produced by continual optimization itself as a source of memory. Rather than replacing the previous model after every task, FreeMOCA

first adapts it to the new task and then combines the preceding and adapted states in parameter space. Because each consolidated state is used to initialize the next task, earlier knowledge is carried forward recursively without storing previous data.

3.1 Warm-Started Adaptation and Recursive Consolidation

We distinguish the task-adapted model θ_t^a from the consolidated model θ_t^c retained after task t. The first task is trained normally, with $\theta_1^c = \theta_1^a$. For each $t \geq 2$, the classifier is expanded for the new classes and optimization is warm-started from the preceding consolidated model,

$$\theta_t^a = \mathrm{Train}\big(\theta_{t-1}^c, \mathcal{D}_t\big)\,. \tag{1}$$

Thus, θ_t^a is a direct optimization descendant of θ_{t-1}^c rather than an independently trained solution.

Newly introduced classifier parameters have no counterpart in θ_{t-1}^c and are therefore copied directly from θ_t^a. We define the dimension-matched reference

$$(\bar{\theta}_{t-1}^c)^{\mathrm{shared}} = (\theta_{t-1}^c)^{\mathrm{shared}}, \qquad (\bar{\theta}_{t-1}^c)^{\mathrm{new}} = (\theta_t^a)^{\mathrm{new}}. \tag{2}$$

where "shared" and "new" denote previously existing and newly introduced parameters, respectively. Thus, newly introduced classifier coordinates are held fixed at their adapted values, while interpolation acts only on parameters shared by the two states. Let

$$d_t = \theta_t^a - \bar{\theta}_{t-1}^c \tag{3}$$

denote the task-induced displacement. The newly introduced classifier parameters have zero displacement by construction. FreeMOCA consolidates the shared parameters through

$$\theta_t^c = \bar{\theta}_{t-1}^c + \lambda d_t = (1-\lambda)\bar{\theta}_{t-1}^c + \lambda\theta_t^a, \qquad 0 \leq \lambda \leq 1. \tag{4}$$

Consequently,

$$\|\theta_t^c - \bar{\theta}_{t-1}^c\|_2 = \lambda\|d_t\|_2. \tag{5}$$

The resulting model θ_t^c becomes the initialization for task $t+1$. Eq. 4 defines the core FreeMOCA mechanism.

3.2 Local Connectivity Along the Continual Trajectory

Warm-starting preserves direct parameter correspondence because the adapted state is obtained by optimizing the preceding consolidated state. This does not by itself guarantee a low-loss linear path. We therefore characterize the interpolation geometry explicitly.

Let

$$\theta_t(\alpha) = \bar{\theta}_{t-1}^c + \alpha d_t, \qquad \alpha \in [0,1], \tag{6}$$

and denote this segment by $\mathcal{S}_t = \{\bar{\theta}_{t-1}^c + \alpha d_t : \alpha \in [0,1]\}$. Let $\mathcal{L}_{\leq t}$ denote the evaluation loss over all classes observed through task t. We define the straight-line interpolation barrier as

$$B_t = \max_{\alpha\in[0,1]} \left[\mathcal{L}_{\leq t}(\theta_t(\alpha)) - (1-\alpha)\mathcal{L}_{\leq t}(\bar{\theta}_{t-1}^c) - \alpha\mathcal{L}_{\leq t}(\theta_t^a)\right]. \tag{7}$$

Unlike endpoint loss, B_t measures the excess loss introduced by the geometry of the interpolation path itself.

Proposition 1 (Local Barrier Bound). Suppose $\mathcal{L}_{\leq t}$ is twice continuously differentiable on $\mathcal{S}_t$ and

$$\sup_{\theta\in\mathcal{S}_t} \|\nabla^2\mathcal{L}_{\leq t}(\theta)\|_2 \leq \beta_t. \tag{8}$$

Then

$$B_t \leq \frac{\beta_t}{8}\|d_t\|_2^2. \tag{9}$$

Hence, local interpolation barriers are governed jointly by task-induced parameter displacement and the curvature encountered between consecutive states. The proposition does not assume that warm-starting guarantees connectivity; Section 5.1 tests this empirically by contrasting independent, shared-initialization, and warm-started solutions. The proof is given in Appendix A.3.

3.3 The Continual Trajectory as Implicit Memory

Recursive consolidation implies that the current state depends on more than the immediately preceding task. Expanding Eq. 4 reveals how earlier adapted states remain implicitly represented.

Proposition 2 (Implicit Trajectory Memory). For a parameter block shared throughout tasks $1, \ldots, t$,

$$\theta_t^c = \sum_{k=1}^{t} \alpha_{k,t} \theta_k^a, \qquad \alpha_{k,t} \geq 0, \qquad \sum_{k=1}^{t} \alpha_{k,t} = 1, \tag{10}$$

where

$$\alpha_{1,t} = (1-\lambda)^{t-1}, \qquad \alpha_{k,t} = \lambda(1-\lambda)^{t-k}, \quad 2 \leq k \leq t. \tag{11}$$

Thus, θ_t^c is a recursive convex aggregation of states encountered along the continual trajectory. Earlier states need not be explicitly retained: their parameter-space impact is integrated into the single consolidated model carried forward. This identity describes implicit parameter memory, not a guarantee that every previous task is perfectly preserved. The proof and the extension to classifier parameters introduced at later tasks are given in Appendix A.4.

3.4 Retention Under Recursive Consolidation

We finally characterize how one consolidation step can perturb an earlier task. Let $\mathcal{L}_j^{(t)}$ denote the loss of task $j < t$ when predictions are made over the class space $\mathcal{Y}_{1:t}$. Expanding the classifier can itself change this loss; we separate this effect as

$$\Delta_{j,t}^{\text{exp}} = \mathcal{L}_j^{(t)}(\bar{\theta}_{t-1}^c) - \mathcal{L}_j^{(t-1)}(\theta_{t-1}^c). \tag{12}$$

Proposition 3 (Controlled Earlier-Task Perturbation). Suppose the earlier-task gradient at $\bar{\theta}_{t-1}^c$ is bounded and the curvature is bounded along the full adaptation segment $\mathcal{S}_t$,

$$\|\nabla \mathcal{L}_j^{(t)}(\bar{\theta}_{t-1}^c)\|_2 \leq \epsilon_{j,t}, \qquad \sup_{\theta \in \mathcal{S}_t} \|\nabla^2 \mathcal{L}_j^{(t)}(\theta)\|_2 \leq \beta_{j,t}. \tag{13}$$

Then

$$\mathcal{L}_j^{(t)}(\theta_t^c) - \mathcal{L}_j^{(t-1)}(\theta_{t-1}^c) \leq \Delta_{j,t}^{\text{exp}} + \lambda \epsilon_{j,t} \|d_t\|_2 + \frac{\beta_{j,t}}{2} \lambda^2 \|d_t\|_2^2. \tag{14}$$

Directly carrying the adapted model forward corresponds to $\lambda = 1$. Therefore, for the same adapted endpoint and local loss geometry, interpolation reduces the first- and second-order terms in this upper bound by factors λ and λ^2, respectively. The result does not guarantee zero forgetting: classifier expansion, task sensitivity, curvature, and the magnitude of the new-task displacement remain relevant. The proof, the exact comparison with $\lambda = 1$, and the accumulated multi-task bound are provided in Appendix A.5.

Together, the three propositions characterize complementary aspects of FreeMOCA: Proposition 1 connects local connectivity to displacement and curvature; Proposition 2 shows that recursive consolidation carries a weighted history of the optimization trajectory; and Proposition 3 characterizes how interpolation bounds perturbations to earlier-task losses. None of these quantities is required during training; previous-task losses, gradients, and curvature appear only in the analysis.

4 Experimental Details

Datasets. Following prior malware Class-IL studies (Rahman et al., 2022; 2025; Park et al., 2025), we evaluate on two large-scale Windows and Android malware benchmarks. EMBER-Class contains 337,035 malicious PE files from the 100 largest malware families, each with over 400 samples (Anderson & Roth, 2018; Rahman et al., 2022). Its static features include PE metadata, imported/exported functions, and statistics such as entropy, with feature hashing used for a fixed-dimensional input. AZ-Class contains 285,582 Android malware samples from 100 families, each with at least 200 samples (Allix et al., 2016; Rahman et al., 2025). It uses DREBIN features (Arp et al., 2014) covering permissions, hardware components, API calls, and network addresses; the original million-plus dimensional space is reduced to 2,439 features using scikit-learn's `VarianceThreshold`.

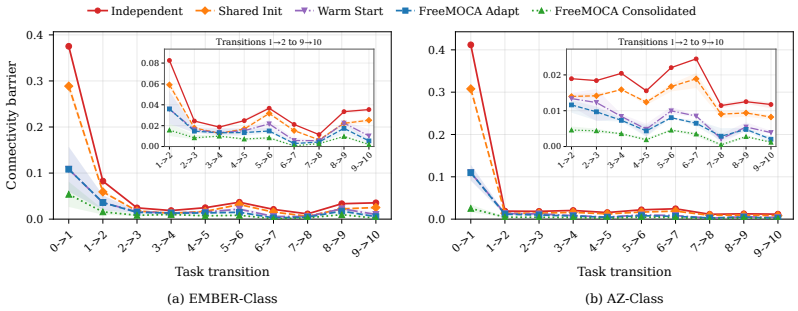


Figure 2: Connectivity barriers across task transitions on (a) EMBER-Class and (b) AZ-Class. Shared initialization lowers the barrier, warm-starting reduces it further, and FreeMOCA consolidation yields the lowest barriers. Shaded regions denote variability over three seeds.

Task Construction. Consistent with prior Class-IL settings (Park et al., 2025; Rahman et al., 2025; 2022), we partition the dataset into 11 sequential tasks: an initial task (Task 0) with 50 randomly selected malware families, followed by 10 tasks (Task 1–10), each introducing five new families and progressively expanding the label space. Section 3 indexes the same sequence from $t = 1$ for notational convenience.

Model and Training. We use a three-layer CNN as the default backbone. The default class order is fixed using seed 20. Each task is trained for 50 epochs with batch size 256 using SGD with a learning rate of 10^{-3}, momentum 0.9, and weight decay 10^{-7}. Unless stated otherwise, methods use the same task order, data splits, backbone, and training schedule. To test dependence on architecture, we additionally evaluate a ViT, a ten-layer CNN, and a Convolutional Kolmogorov–Arnold Network (CKAN) in Section B.7.

Connectivity Evaluation. To separate the roles of shared ancestry and continuous optimization history in linear connectivity (Mirzadeh et al., 2021; Frankle et al., 2020), we evaluate Eq. 7 on the cumulative test set $\mathcal{D}^{\text{test}}_{\leq t-1} = \bigcup_{j=0}^{t-1} \mathcal{D}^{\text{test}}_j$ using $\alpha \in \{0, 0.05, \ldots, 1\}$. We compare independent, shared-initialized, and warm-started training, then examine the corresponding trajectories in FreeMOCA. We also measure activation-variance stability to test whether direct interpolation causes variance collapse without alignment or BN repair.

Baselines. We use *None*, which sequentially fine-tunes without a CL mechanism, as the lower reference and *Joint*, which trains on all data observed so far, as the upper reference. We compare against EWC (Kirkpatrick et al., 2017), SI (Zenke et al., 2017), LwF (Li & Hoiem, 2017), GR (Shin et al., 2017), iCaRL (Rebuffi et al., 2017), MalCL (Park et al., 2025), TAMiL (Bhat et al., 2023), CLeWI (Kozal et al., 2024), WSC (Cho et al., 2026), and MESU (Bonnet et al., 2025). Replay-based methods use a matched replay budget of 500 samples per task and are re-evaluated with our common classifier and training setup, while retaining their method-specific replay and consolidation mechanisms.

Metrics. We report cumulative loss barriers and activation-variance ratios for interpolation analysis, together with Average Accuracy, Forgetting (F), Remembering (REM), Forward Transfer (FWT), and Backward Transfer (BWT/BWT+) for continual-learning performance (Rahman et al., 2022; Díaz-Rodríguez et al., 2018). Results are reported as mean ± standard deviation over three random seeds.

Computational Resources. Experiments are conducted on an Ubuntu 22 server equipped with NVIDIA RTX A6000 GPUs with 48 GB of VRAM.

5 Results

5.1 Emergence of Linear Connectivity Along the Continual Trajectory

Figure 2 evaluates straight-line connectivity between consecutive task solutions on the cumulative test set using Eq. 7, as described in Section 4. Independent initialization has the largest mean barrier,

Table 1: Variance-collapse diagnostics at `block2`. Shaded ratios indicate significant collapse.

EMBER-Class										
Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
V_{prev}	0.1450	0.0595	0.1045	0.1750	0.1280	0.1946	0.1726	0.1768	0.1994	0.2078
V_{curr}	0.3241	0.1981	0.2008	0.2654	0.1922	0.2173	0.1836	0.1704	0.1638	0.1921
Ratio	1.0376	**0.7239**	0.8109	0.8594	0.9292	0.9842	0.9708	0.9856	1.0188	0.9560
Mean ratio: 0.928			**Collapses:** 1/10							
AZ-Class										
Task	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
V_{prev}	0.3107	0.1350	0.1847	0.2339	0.3553	0.3724	0.3401	0.3745	0.4357	0.4866
V_{curr}	0.7059	0.5838	0.4786	0.5726	0.5664	0.6145	0.4588	0.5105	0.5216	0.5250
Ratio	1.1509	**0.5913**	**0.6786**	**0.7188**	**0.7709**	0.8857	0.8917	0.9151	0.9591	0.9481
Mean ratio: 0.852			**Collapses:** 4/10							

shared initialization reduces it, and warm-starting reduces it further. On EMBER-Class, the mean barrier decreases from 0.0664 ± 0.0008 to 0.0496 ± 0.0013 and 0.0255 ± 0.0054; on AZ-Class, from 0.0567 ± 0.0002 to 0.0427 ± 0.0006 and 0.0179 ± 0.0015. Shared Init. is below Independent in all 30 seed–transition comparisons on both datasets, while Warm Start is below Shared Init. in $24/30$ on EMBER-Class and $28/30$ on AZ-Class. The largest barriers occur at the first transition and are substantially smaller thereafter.

FreeMOCA with $\lambda = 0.6$ in its adaptation path $\bar{\theta}_{t-1} \to \theta_t$ mostly remains below Warm Start, particularly after first transitions, while the retained path $\bar{\theta}_{t-1} \to \bar{\theta}_t$ has the lowest mean barriers (0.0119 on EMBER-Class and 0.0052 on AZ-Class). Since this path is already part of the adaptation line, we use it to characterize where FreeMOCA retains its consolidated states, not as a separate connectivity control. Overall, shared ancestry explains part of the observed connectivity, while sequential warm-starting strengthens it further, supporting the use of the continual optimization trajectory for replay-free consolidation.

5.2 Do Consecutive CL States Require Alignment and Variance Repair?

The loss-barrier analysis shows that consecutive CL states can become linearly connected, but a low-loss path does not guarantee that their internal representations remain aligned during interpolation. This matters because CLeWI uses replay-based neuron alignment and normalization repair before and after interpolation (Kozal et al., 2024). These steps are designed to address mismatches between models obtained from independent initializations or separate training trajectories.

FreeMOCA operates in a different manner. Each adapted model is obtained directly from the preceding consolidated state, so shared parameters keep their identity throughout training. The remaining question is whether their activations also stay compatible enough to avoid variance collapse.

Using the same probe batch for the previous, adapted, and interpolated models, we measure activation variance at `block2` and define

$$R_t = \frac{\mathrm{Var}(\Phi(\theta_{\text{interp}}))}{(1-\lambda)\,\mathrm{Var}(\Phi(\theta_{\text{prev}})) + \lambda\,\mathrm{Var}(\Phi(\theta_{\text{adapt}}))}, \tag{15}$$

where $\Phi(\theta)$ denotes the `block2` activations and ratio values near one indicate that interpolation preserves the variance expected from its endpoints, while smaller values indicate contraction.

Table 1 shows that variance collapse is mostly confined to the early transitions: it appears in only $1/10$ transitions on EMBER-Class and $4/10$ on AZ-Class using $R_t < 0.8$ as variance collapse threshold, with no further collapse after the fifth transition. After the final collapse, the mean ratio rises to 0.939 and 0.920, respectively. This mirrors the loss-barrier result: as the continual trajectory develops, consecutive states become both easier to connect and more stable to interpolate. In FreeMOCA, parameter correspondence is inherited from sequential optimization, and persistent variance repair is largely confined to early transitions.

Table 2: FreeMOCA in Class-IL: Average accuracy (%) on EMBER-Class and AZ-Class compared with CL baselines. Results are mean ± standard deviation, grouped by replay requirement.

Dataset	None	Replay-free Methods				FreeMOCA (ours)
		EWC	SI	LwF	MESU	$\lambda = 0.6$
EMBER-Class	15.4±2.1	26.4±0.3	26.9±0.3	31.1±0.3	18.0±0.8	**65.2±0.05**
AZ-Class	14.5±0.5	27.3±6.2	28.0±0.3	33.9±1.4	15.5±1.8	**63.7±0.03**

Dataset	Joint	Replay-based Methods and Joint Training					
		GR	MalCL	TAMiL	iCaRL	CLeWI	WSC
EMBER-Class	89.1±0.6	27.0±0.3	50.3±4.1	63.1±2.7	37.7±0.8	60.3±2.3	62.1±1.0
AZ-Class	87.2±1.1	27.9±0.1	51.6±6.3	52.1±1.8	40.4±0.5	54.9±1.4	51.7±0.2

Table 3: CL transfer and forgetting metrics. All values are reported as fractions.

Method	EMBER-Class					AZ-Class				
	FWT ↑	BWT ↑	BWT$^+$ ↑	F ↓	REM ↑	FWT ↑	BWT ↑	BWT$^+$ ↑	F ↓	REM ↑
FreeMOCA	**0.0002**	**-0.0060**	0.0000	**0.0060**	**0.9940**	0.0000	**0.1480**	**0.1480**	**0.0000**	**1.0000**
EWC	0.0000	-0.4059	0.0000	0.4482	0.5941	0.0000	-0.6549	0.0000	0.5680	0.3450
SI	0.0000	-0.8534	0.0000	0.9447	0.1465	0.0000	-0.8463	0.0000	0.8498	0.1536
GR	0.0000	-0.6040	0.0000	0.6469	0.3960	0.0000	-0.5264	0.0000	0.6138	0.4735
MalCL	0.0000	-0.1477	0.0000	0.1507	0.8523	0.0000	-0.1753	0.0000	0.2563	0.8246
LwF	0.0000	-0.1691	0.0000	0.0958	0.8309	0.0000	-0.1625	0.0000	0.0900	0.8374
TAMiL	0.0000	-0.2569	0.0000	0.5334	0.7431	0.0000	-0.2948	0.0000	0.3793	0.7051
iCaRL	0.0000	-0.0986	0.0000	0.1470	0.9013	0.0000	-0.1270	0.0000	0.1806	0.8729
CLeWI	0.0000	-0.1444	0.0000	0.0625	0.8555	0.0000	0.0144	0.0144	0.0589	**1.0000**
WSC	0.0000	-0.2144	0.0000	0.2052	0.7856	0.0000	-0.3397	0.0000	0.3770	0.6603

5.3 Overall Class-IL Performance

Table 2 shows that FreeMOCA outperforms all replay-free baselines, reaching $65.2 \pm 0.05\%$ on EMBER-Class and $63.7 \pm 0.03\%$ on AZ-Class. It also remains competitive with, and overall stronger than, the replay-based methods despite using no replay. The closest replay-based baselines on EMBER-Class are TAMiL (63.1%), WSC (62.1%), and CLeWI (60.3%), but all three obtain lower accuracy on AZ-Class, where WSC and CLeWI reach 51.7% and 54.9%, while TAMiL reaches 52.1%. In contrast, FreeMOCA remains strong across both malware datasets.

Table 3 shows strong retention across both datasets. On EMBER-Class, FreeMOCA achieves the lowest forgetting ($F = 0.006$), highest REM (0.994), and best BWT (-0.006), indicating little degradation of earlier tasks. On AZ-Class, it records no measured forgetting ($F = 0.000$), positive BWT (0.148), and REM of 1.000, tied with CLeWI. Together with the Average Accuracy results, these metrics show that FreeMOCA preserves earlier knowledge while continuing to learn new malware classes.

5.4 Memory and Training Efficiency

Table 4 compares per-task training time and persistent cross-task storage. FreeMOCA requires only 38.05 s per task, versus 679.30 s for WSC, 3804.38 s for MalCL, and 12960.18 s for CLeWI, reducing training time by 94.4–99.7%. It also keeps an approximately constant 15.98 MB state, lower than CLeWI's 30.39 MB and far below MalCL's 284.29 MB average; WSC grows from 10.90 MB to 19.99 MB across tasks. Together with the results in Section 5.3, this shows that FreeMOCA improves predictive performance while keeping continual updates lightweight in both storage and computation.

Table 4: Memory and training-time comparison of MalCL, WSC, CLeWI, and FreeMOCA. Memory denotes persistent cross-task storage.

Metric	MalCL	WSC	CLeWI	FreeMOCA
Time (s/task)	3804.38	679.30	12960.18	**38.05**
Avg. memory (MB)	284.29	16.44	30.39	**15.98**
Memory growth (%)	86.3	45.4	0.0	**0.0**
Memory range (MB)	223.36–416.20	10.90–19.99	30.39–30.39	**15.98–15.98**

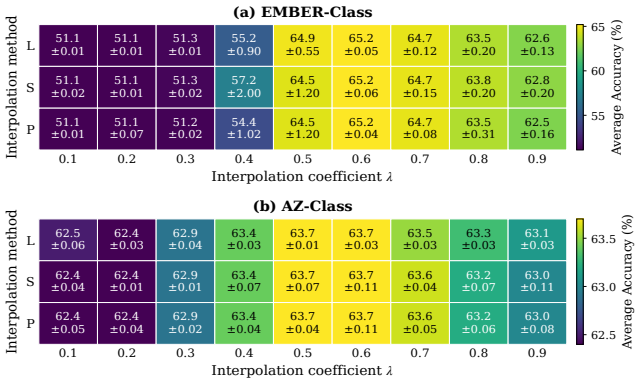


Figure 3: Sensitivity to the interpolation coefficient λ on EMBER-Class and AZ-Class using linear (L), spline (S), and polynomial (P) interpolation.

6 Ablation Studies

Stability vs. Plasticity. We study the sensitivity of FreeMOCA to the interpolation coefficient. Smaller values emphasize on the prior consolidated state, while larger values favor the newly adapted model. We select $\lambda = 0.6$ using the validation split and keep it fixed across experiments. As shown in Figure 3, this setting reaches 65.2% Average Accuracy on EMBER-Class and 63.7% on AZ-Class.

We extend our ablation in Appendix B.

7 Discussion and Limitations

We provide empirical evidence of local low-loss linear connectivity between consecutive CL solutions when successive tasks can reuse much of the existing representation. We therefore do not claim that arbitrary points along the continual trajectory are connected: abrupt distribution shifts may separate neighboring solutions and increase interpolation barriers. Connectivity and forgetting are related but not equivalent. A small barrier means that interpolation adds little extra loss even if the endpoints have already forgotten part of the past. This helps explain why connectivity can strengthen while forgetting still accumulates. Over very long sequences, local geometric compatibility may persist even as the endpoints gradually drift, limiting long-term retention.

8 Conclusion

Replay is often impractical for malware continual learning because of storage, computation, and data-access constraints. This led us to ask whether the continual learning trajectory itself can act as a form of memory. Our results support this view: shared initialization establishes linear connectivity, while sequential training trajectory strengthens it further as learning progresses. At the same time, consecutive states remain sufficiently aligned that interpolation does not require replay-based neuron matching or BatchNorm repair. Rather than storing the past explicitly, FreeMOCA reuses the geometry already present in the continual learning trajectory. The result is a replay-free consolidation mechanism that is simple, efficient, and well suited to resource-constrained settings.

AI USE STATEMENT

We used generative AI tools to assist with editing and polishing the manuscript for clarity and readability, including rephrasing and improving the presentation of author-written text. We did not use generative AI tools to generate synthetic data, develop the theoretical or conceptual framework, formulate or prove mathematical claims, propose hypotheses, design the research methodology or experiments, implement the proposed method, process the datasets, or interpret the experimental results. All AI-assisted text was reviewed and revised by the authors. The authors take full responsibility for the final content of the paper, including all technical claims, analyses, and conclusions.

REPRODUCIBILITY STATEMENT

We provide the methodological, implementation, and experimental details needed to reproduce our results. The paper describes datasets, task construction, training hyperparameters, replay budget, evaluation metrics, and the FreeMOCA consolidation procedure, while model architecture and extended results are provided in the appendix A and B, respectively. We also provide anonymized source code with the submission to facilitate reproduction of the reported experiments.

ETHICS STATEMENT

This work studies continual learning for malware classification using existing benchmark datasets and does not involve human subjects or personal data. The proposed method is intended for defensive malware detection and research. We do not develop or distribute malware, and the method does not enable malware generation or deployment. However, malware research can have dual-use implications, so the presented techniques should be used responsibly.

REFERENCES

Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git Re-Basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations (ICLR)*, 2023.

Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. Androzoo: Collecting millions of android apps for the research community. In *International Conference on Mining Software Repositories (MSR)*, 2016.

Hyrum S Anderson and Phil Roth. EMBER: an open dataset for training static pe malware machine learning models. *arXiv:1804.04637*, 2018.

Elahe Arani, Fahad Sarfraz, and Bahram Zonooz. Learning fast, learning slow: A general continual learning method based on complementary learning system. *arXiv preprint arXiv:2201.12604*, 2022.

Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, and Konrad Rieck. Drebin: Effective and explainable detection of android malware in your pocket. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

Prashant Bhat, Bahram Zonooz, and Elahe Arani. Task-aware information routing from common representation space in lifelong learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Alexander Dylan Bodner, Antonio Santiago Tepsich, Jack Natan Spolski, and Santiago Pourteau. Convolutional kolmogorov-arnold networks. *arXiv preprint arXiv:2406.13155*, 2024.

Djohan Bonnet, Kellian Cottart, Tifenn Hirtzlin, Tarcisius Januel, Thomas Dalgaty, Elisa Vianello, and Damien Querlioz. Bayesian continual learning and forgetting in neural networks. *Nature Communications*, 16(1):9614, 2025.

Dongkyu Cho, Taesup Moon, Rumi Chunara, Kyunghyun Cho, and Sungmin Cha. Forget forgetting: Continual learning in a world of abundant memory. In *International Conference on Learning Representations (ICLR)*, 2026.

Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning (ICML)*, 2020.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 1999.

Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. In *International Conference on Computer Vision (ICCV)*, 2023.

Daniele Ghiani, Daniele Angioni, Giorgio Piras, Angelo Sotgiu, Luca Minnei, Srishti Gupta, Maura Pintor, Fabio Roli, and Battista Biggio. Regression-aware continual learning for android malware detection. *IEEE Transactions on Information Forensics and Security*, 2026.

Yiling He, Junchi Lei, Hongyu She, Shuo Shao, Xinran Zheng, Yiping Liu, Zhan Qin, and Lorenzo Cavallaro. Retrofit: Continual learning with controlled forgetting for binary security detection and analysis. In *USENIX Security Symposium*, 2026.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*, 2018.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences (PNAS)*, 2017.

Jędrzej Kozal, Jan Wasilewski, Bartosz Krawczyk, and Michał Woźniak. Continual learning with weight interpolation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Mei Li, Yuxiang Lu, Qinyan Dai, Suizhi Huang, Yue Ding, and Hongtao Lu. BECAME: Bayesian continual learning with adaptive model merging. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *International Conference on Machine Learning (ICML)*, 2025.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Dilan Gorur, Razvan Pascanu, and Hassan Ghasemzadeh. Linear mode connectivity in multitask and continual learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Jimin Park, AHyun Ji, Minji Park, Mohammad Saidur Rahman, and Se Eun Oh. Malcl: Leveraging gan-based generative replay to combat catastrophic forgetting in malware classification. In *AAAI Conference on Artificial Intelligence*, 2025.

Jimin Park, Haeseung Jeon, AHyun Ji, Aritran Piplai, Mohammad Saidur Rahman, and Se Eun Oh. Tabcl: Continual malware classification with tabular-aware generation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 547–559. Springer, 2026.

Haomiao Qiu, Miao Zhang, Ziyue Qiao, and Liqiang Nie. Train with perturbation, infer after merging: A two-stage framework for continual learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Mohammad Saidur Rahman, Scott Coull, and Matthew Wright. On the limitations of continual learning for malware classification. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022.

Mohammad Saidur Rahman, Scott Coull, Qi Yu, and Matthew Wright. MADAR: Efficient continual learning for malware analysis with distribution-aware replay. In *Conference on Applied Machine Learning in Information Security (CAMLIS)*, 2025.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

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

Albin Soutif-Cormerais, Antonio Carta, and Joost Van de Weijer. Improving online continual learning performance and stability with temporal ensembles. In *Conference on Lifelong Learning Agents*, pp. 828–845. PMLR, 2023.

Zafir Stojanovski, Karsten Roth, and Zeynep Akata. Momentum-based weight interpolation of strong zero-shot models for continual learning. *arXiv preprint arXiv:2211.03186*, 2022.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

Gido M Van de Ven, Hava T Siegelmann, and Andreas S Tolias. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 2020.

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

Haitao Wen, Haoyang Cheng, Heqian Qiu, Lanxiao Wang, Lili Pan, and Hongliang Li. Optimizing mode connectivity for class incremental learning. In *International Conference on Machine Learning (ICML)*, 2023.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*, 2017.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, 2017.

Peiyan Zhang, Yuchen Yan, Chaozhuo Li, Senzhang Wang, Xing Xie, Guojie Song, and Sunghun Kim. Continual learning on dynamic graphs via parameter isolation. In *International ACM SIGIR conference on research and development in information retrieval (SIGIR)*, pp. 601–611, 2023.

A Additional Method Details and Theoretical Analysis

A.1 FreeMOCA Algorithm

Algorithm 1 FreeMOCA: replay-free recursive consolidation.

Input: task datasets $\{\mathcal{D}_t\}_{t=1}^{T}$, initial model θ_0
Parameter: interpolation coefficient $\lambda \in [0, 1]$
Output: consolidated model θ_T^c

$\theta_1^a \leftarrow \mathrm{Train}(\theta_0, \mathcal{D}_1)$
$\theta_1^c \leftarrow \theta_1^a$
for $t = 2$ to T **do**
  Expand the classifier for the new classes in $\mathcal{D}_t$
  $\theta_t^a \leftarrow \mathrm{Train}(\theta_{t-1}^c, \mathcal{D}_t)$
  Construct $\bar{\theta}_{t-1}^c$ by retaining the shared parameters of θ_{t-1}^c and copying the new classifier parameters from θ_t^a
  $\theta_t^c \leftarrow (1-\lambda)\bar{\theta}_{t-1}^c + \lambda\theta_t^a$
end for
return θ_T^c

A.2 Parameter-Space Interpolation

Given two compatible parameter states θ_a and θ_b, linear interpolation constructs

$$\theta(\alpha) = (1-\alpha)\theta_a + \alpha\theta_b, \qquad \alpha \in [0, 1]. \tag{16}$$

In FreeMOCA, the endpoints are the dimension-matched preceding state $\bar{\theta}_{t-1}^c$ and the adapted state θ_t^a. New classifier parameters are already matched through Eq. 2.

We use linear interpolation in the core method because it requires only these two states. Polynomial and spline interpolation, which require additional states to construct the path, are evaluated only as alternative interpolation schemes in the ablation study.

A.3 Proof of Proposition 1

Recall

$$\mathcal{S}_t = \left\{\bar{\theta}_{t-1}^c + \alpha d_t : \alpha \in [0, 1]\right\}. \tag{17}$$

Define

$$g_t(\alpha) = \mathcal{L}_{\le t}\left(\bar{\theta}_{t-1}^c + \alpha d_t\right), \qquad \alpha \in [0, 1]. \tag{18}$$

Its second derivative is

$$g_t''(\alpha) = d_t^\top \nabla^2 \mathcal{L}_{\le t}\left(\bar{\theta}_{t-1}^c + \alpha d_t\right) d_t. \tag{19}$$

By the Hessian bound in Proposition 1,

$$|g_t''(\alpha)| \le \beta_t \|d_t\|_2^2. \tag{20}$$

For a twice continuously differentiable scalar function, the error relative to its endpoint chord satisfies

$$|g_t(\alpha) - [(1-\alpha)g_t(0) + \alpha g_t(1)]| \le \frac{1}{2}\alpha(1-\alpha) \sup_{s\in[0,1]} |g_t''(s)|. \tag{21}$$

Therefore,

$$g_t(\alpha) - [(1-\alpha)g_t(0) + \alpha g_t(1)] \le \frac{\beta_t}{2}\alpha(1-\alpha)\|d_t\|_2^2. \tag{22}$$

Since $\max_{\alpha\in[0,1]} \alpha(1-\alpha) = 1/4$,

$$B_t \le \frac{\beta_t}{8}\|d_t\|_2^2, \tag{23}$$

which proves Proposition 1.

The result is local to consecutive states. It does not imply linear connectivity between arbitrary models along the continual trajectory.

A.4 Proof of Proposition 2

Consider any parameter block shared throughout tasks $1, \dots, t$. For this block, classifier expansion does not affect the recurrence, and Eq. 4 becomes

$$\theta_t^c = (1-\lambda)\theta_{t-1}^c + \lambda\theta_t^a, \qquad t \geq 2, \tag{24}$$

with $\theta_1^c = \theta_1^a$.

We show by induction that

$$\theta_t^c = \sum_{k=1}^{t} \alpha_{k,t}\theta_k^a, \tag{25}$$

where

$$\alpha_{1,t} = (1-\lambda)^{t-1}, \qquad \alpha_{k,t} = \lambda(1-\lambda)^{t-k}, \quad 2 \leq k \leq t. \tag{26}$$

For $t = 1$, the result follows from $\theta_1^c = \theta_1^a$. Assume it holds after task $t-1$. Substituting into Eq. 24,

$$\begin{aligned} \theta_t^c &= (1-\lambda)\sum_{k=1}^{t-1} \alpha_{k,t-1}\theta_k^a + \lambda\theta_t^a \\ &= \sum_{k=1}^{t-1} (1-\lambda)\alpha_{k,t-1}\theta_k^a + \lambda\theta_t^a. \end{aligned} \tag{27}$$

Thus,

$$\alpha_{k,t} = (1-\lambda)\alpha_{k,t-1}, \quad k < t, \qquad \alpha_{t,t} = \lambda, \tag{28}$$

which gives Eq. 26.

The coefficients are nonnegative because $0 \leq \lambda \leq 1$, and

$$\sum_{k=1}^{t} \alpha_{k,t} = (1-\lambda)\sum_{k=1}^{t-1} \alpha_{k,t-1} + \lambda = 1. \tag{29}$$

Hence θ_t^c is a convex combination of the adapted states, proving Proposition 2.

Parameters introduced at later tasks. Suppose a classifier block q is first introduced at task $s > 1$. By Eq. 2,

$$(\theta_s^c)^{(q)} = (\theta_s^a)^{(q)}. \tag{30}$$

For $t > s$, it follows the same recurrence, giving

$$(\theta_t^c)^{(q)} = \sum_{k=s}^{t} \alpha_{k,t}^{(s)} (\theta_k^a)^{(q)}, \tag{31}$$

where

$$\alpha_{s,t}^{(s)} = (1-\lambda)^{t-s}, \qquad \alpha_{k,t}^{(s)} = \lambda(1-\lambda)^{t-k}, \quad s < k \leq t. \tag{32}$$

Again, $\alpha_{k,t}^{(s)} \geq 0$ and $\sum_{k=s}^{t} \alpha_{k,t}^{(s)} = 1$.

Thus, each parameter block begins accumulating trajectory information from the task at which it is introduced. This is an exact parameter-space identity; it does not imply perfect functional retention of every earlier task.

A.5 Proof of Proposition 3

Fix an earlier task $j < t$ and define

$$f(s) = \mathcal{L}_j^{(t)}\left(\bar{\theta}_{t-1}^c + s d_t\right), \qquad s \in [0,1]. \tag{33}$$

Then

$$f(0) = \mathcal{L}_j^{(t)}(\bar{\theta}_{t-1}^c), \qquad f(\lambda) = \mathcal{L}_j^{(t)}(\theta_t^c). \tag{34}$$

By the chain rule,

$$f'(s) = \nabla \mathcal{L}_j^{(t)} \left(\bar{\theta}_{t-1}^c + s d_t\right)^\top d_t, \tag{35}$$

and

$$f''(s) = d_t^\top \nabla^2 \mathcal{L}_j^{(t)} \left(\bar{\theta}_{t-1}^c + s d_t\right) d_t. \tag{36}$$

Taylor's theorem around $s = 0$ gives

$$f(\lambda) = f(0) + \lambda f'(0) + \int_0^\lambda (\lambda - s) f''(s)\, ds. \tag{37}$$

Using

$$\left\| \nabla \mathcal{L}_j^{(t)}(\bar{\theta}_{t-1}^c) \right\|_2 \le \epsilon_{j,t}, \tag{38}$$

Cauchy–Schwarz gives

$$f'(0) \le \epsilon_{j,t} \|d_t\|_2. \tag{39}$$

Likewise, the curvature assumption on $\mathcal{S}_t$ implies

$$|f''(s)| \le \beta_{j,t} \|d_t\|_2^2. \tag{40}$$

Substituting into Eq. 37,

$$f(\lambda) - f(0) \le \lambda \epsilon_{j,t} \|d_t\|_2 + \frac{\beta_{j,t}}{2} \lambda^2 \|d_t\|_2^2. \tag{41}$$

The total change in earlier-task loss also includes classifier expansion. Recall

$$\Delta_{j,t}^{\exp} = \mathcal{L}_j^{(t)}(\bar{\theta}_{t-1}^c) - \mathcal{L}_j^{(t-1)}(\theta_{t-1}^c). \tag{42}$$

This term captures the effect of introducing the new classifier outputs while holding the previously shared parameters fixed. Therefore,

$$\begin{aligned} &\mathcal{L}_j^{(t)}(\theta_t^c) - \mathcal{L}_j^{(t-1)}(\theta_{t-1}^c) \\ &= \Delta_{j,t}^{\exp} + f(\lambda) - f(0). \end{aligned} \tag{43}$$

Combining this identity with Eq. 41 yields

$$\begin{aligned} \mathcal{L}_j^{(t)}(\theta_t^c) - \mathcal{L}_j^{(t-1)}(\theta_{t-1}^c) \le \Delta_{j,t}^{\exp} &+ \lambda \epsilon_{j,t} \|d_t\|_2 \\ &+ \frac{\beta_{j,t}}{2} \lambda^2 \|d_t\|_2^2, \end{aligned} \tag{44}$$

which proves Proposition 3.

Comparison with direct sequential carry-forward. Carrying θ_t^a directly to the next task corresponds to $\lambda = 1$. Define

$$Q_{j,t}(\lambda) = \lambda \epsilon_{j,t} \|d_t\|_2 + \frac{\beta_{j,t}}{2} \lambda^2 \|d_t\|_2^2. \tag{45}$$

For $0 \le \lambda \le 1$,

$$\begin{aligned} Q_{j,t}(1) - Q_{j,t}(\lambda) = (1 - \lambda) \epsilon_{j,t} \|d_t\|_2 &\\ + \frac{\beta_{j,t}}{2} (1 - \lambda^2) \|d_t\|_2^2 \ge 0. & \end{aligned} \tag{46}$$

Thus, for the same adapted endpoint and local geometry, interpolation reduces the shared-parameter contribution to the upper bound relative to directly carrying the adapted model forward. This does not imply that smaller λ is always preferable, since reducing λ also limits current-task adaptation.

Table 5: Straight-line interpolation barriers on the cumulative test set. Lower values indicate stronger linear connectivity. Results are mean ± standard deviation over three seeds.

Method	EMBER-Class	AZ-Class
Independent Init.	0.0664 ± 0.0008	0.0567 ± 0.0002
Shared Init.	0.0496 ± 0.0013	0.0427 ± 0.0006
Warm Start	0.0255 ± 0.0054	0.0179 ± 0.0015
FreeMOCA Adapt	0.0231 ± 0.0041	0.0167 ± 0.0017
FreeMOCA Consolidated	0.0119 ± 0.0027	0.0052 ± 0.0008

Accumulation across tasks. For an earlier task $j < T$, telescoping the one-step changes gives

$$\begin{aligned} &\mathcal{L}_j^{(T)}(\theta_T^c) - \mathcal{L}_j^{(j)}(\theta_j^c) \\ &\leq \sum_{t=j+1}^{T} \Delta_{j,t}^{\exp} + \lambda \sum_{t=j+1}^{T} \epsilon_{j,t} \|d_t\|_2 \\ &\quad + \frac{\lambda^2}{2} \sum_{t=j+1}^{T} \beta_{j,t} \|d_t\|_2^2. \end{aligned} \tag{47}$$

The bound separates classifier expansion, first-order sensitivity to subsequent updates, and curvature effects. It does not guarantee monotonic retention: these quantities and the task-induced displacements may vary throughout the sequence. None is required by FreeMOCA during training.

A.6 Classifier Architecture

We use the same lightweight 1D CNN architecture as (Park et al., 2025) for both datasets, with input dimensions of 2,381 for EMBER-Class and 2,439 for AZ-Class. The feature extractor comprises three Conv1D layers with 512, 256, and 128 channels, together with Batch Normalization, ReLU, dropout ($p = 0.5$), and max-pooling. The resulting representation is passed to a fully connected classifier followed by Batch Normalization, dropout, and ReLU. In Class-IL, the output layer is expanded as new classes arrive while preserving the parameters associated with previously observed classes.

B Additional Experiments and Ablations

B.1 Additional Evaluation Beyond Malware

To test FreeMOCA beyond malware, we evaluate it on a 5-task MNIST Class-IL sequence with two classes per task. FreeMOCA reaches 61.54% Average Accuracy, compared with 60.15% for replay-based MalCL (Park et al., 2025). Despite the larger separation between successive MNIST class distributions, FreeMOCA remains competitive without storing past samples.

B.2 Numerical Connectivity Results

Table 5 summarizes the straight-line barriers underlying the transition-wise curves in Figure 2. Barriers are evaluated on the cumulative test set containing all tasks observed up to each transition and averaged over three random seeds.

The numerical results confirm the same hierarchy observed in Figure 2: shared initialization reduces the barrier relative to independent training, while sequential warm-starting reduces it further. FreeMOCA Adapt remains close to ordinary Warm Start, whereas the retained consolidated states occupy the lowest-barrier portion of the local trajectory.

B.3 Are CLeWI's Alignment and Repair Necessary?

Table 6 isolates the effect of CLeWI's alignment and repair components (Kozal et al., 2024). Removing either component improves performance, and removing both increases Average Accuracy

Table 6: Ablation of neuron alignment and normalization repair in CLeWI on EMBER-Class and AZ-Class. Results are Average Accuracy (%).

Variant	Alignment	Repair	EMBER-Class	AZ-Class
Full CLeWI	✓	✓	60.3 ± 2.3	54.9 ± 1.4
w/o Alignment	×	✓	63.4 ± 1.8	55.3 ± 1.3
w/o Repair	✓	×	63.7 ± 0.7	58.5 ± 1.9
Replay + Direct Interpolation	×	×	67.9 ± 2.2	62.6 ± 1.2

Table 7: Sensitivity of FreeMOCA to task ordering. Randomized results are mean ± standard deviation over three Class-IL orders. The dissimilar order maximizes the distance between consecutive task centroids.

Task Order	EMBER-Class	AZ-Class
Randomized orders	65.2 ± 0.05	63.7 ± 0.03
Dissimilar order	22.86	67.86

from 60.3% to 67.9% on EMBER-Class and from 54.9% to 62.6% on AZ-Class. Once both components are removed, however, the resulting configuration is no longer the CLeWI method; it reduces to replay-assisted direct interpolation. These results indicate that warm-started states already support direct interpolation, while the remaining EMBER gain likely comes from replay rather than alignment or repair.

B.4 Task Ordering and Abrupt Shifts

We next test whether FreeMOCA depends on task ordering. Across three randomized Class-IL orders, FreeMOCA achieves $65.2 \pm 0.05\%$ Average Accuracy on EMBER-Class and $63.7 \pm 0.03\%$ on AZ-Class, indicating limited sensitivity to ordinary reordering.

We then construct a more severe sequence by maximizing the input-space distance between consecutive task centroids. As shown in Table 7, performance remains strong on AZ-Class but drops substantially on EMBER-Class. This is consistent with FreeMOCA's bounded-task-drift requirement. AZ-Class uses sparse DREBIN features organized into semantic categories, such that even malware-family groups with distant centroids may retain reusable feature structure across tasks. By contrast, EMBER-Class relies on hashed PE, header, and function-call features, for which deliberately dissimilar family groups can induce a substantially larger change in the learned decision boundary. These transitions increase parameter displacement and interpolation barriers, potentially moving consecutive solutions outside the locally connected region required for reliable consolidation.

B.5 Adaptive Layer-Wise Consolidation

The core FreeMOCA method uses a single global interpolation coefficient λ. As an optional extension, we also examine whether consolidation can be adjusted separately across layers. For each shared layer ℓ, let

$$d_t^{(\ell)} = (\theta_t^a)^{(\ell)} - (\bar{\theta}_{t-1}^c)^{(\ell)}, \qquad s_{t,\ell} = \frac{1}{n_\ell} \left\| d_t^{(\ell)} \right\|_1, \tag{48}$$

where n_ℓ is the number of parameters in layer ℓ. We normalize the layer shifts within each task,

$$\hat{s}_{t,\ell} = \frac{s_{t,\ell} - s_t^{\min}}{s_t^{\max} - s_t^{\min} + \epsilon}, \tag{49}$$

and assign

$$\lambda_{t,\ell} = \lambda_{\min} + (1 - \hat{s}_{t,\ell})(\lambda_{\max} - \lambda_{\min}). \tag{50}$$

The shared parameters are then consolidated as

$$(\theta_t^c)^{(\ell)} = (1 - \lambda_{t,\ell})(\bar{\theta}_{t-1}^c)^{(\ell)} + \lambda_{t,\ell}(\theta_t^a)^{(\ell)}. \tag{51}$$

Table 8: Ablation of the adaptive layer-wise rule in FreeMOCA. Values are Average Accuracy (%).

Layer-wise signal			Adaptive coefficient range		
Configuration	**EMBER**	**AZ**	**Range**	**EMBER**	**AZ**
Inverse displacement (FreeMOCA)	**66.30**	**66.10**	$[0.1, 0.9]$	27.90	60.31
Direct displacement	34.90	59.40	$[0.2, 0.8]$	29.68	60.36
Gradient norm	32.19	60.36	$[0.3, 0.7]$	**66.30**	**66.10**
Activation drift	33.75	60.27	$[0.4, 0.6]$	40.52	60.40
			$[0.5, 0.5]$	64.90	63.70

Thus, layers that change more during adaptation retain more of the preceding consolidated state, while stable layers place more weight on the adapted model. Newly introduced classifier parameters are copied directly from θ_t^a, as in the core method.

Alternative Layer Signals. We further test whether the benefit comes from layer-wise weighting itself or from the inverse-displacement rule.

Direct Displacement. Using the same displacement score,

$$s_{t,\ell} = \frac{1}{n_\ell} \left\| (\theta_t^a)^{(\ell)} - (\bar{\theta}_{t-1}^c)^{(\ell)} \right\|_1, \tag{52}$$

we reverse the mapping so that larger displacements receive larger interpolation coefficients. This variant reaches 34.90% on EMBER-Class and 59.40% on AZ-Class.

Gradient Norm. We replace parameter displacement with the average RMS gradient magnitude over current-task mini-batches,

$$g_{t,\ell} = \mathbb{E}_b \left[\frac{\|\nabla_{\theta^{(\ell)}} \mathcal{L}_b\|_2}{\sqrt{n_\ell}} \right]. \tag{53}$$

Layers with larger gradient norms receive smaller interpolation coefficients. This variant reaches 32.19% on EMBER-Class and 60.36% on AZ-Class.

Activation Drift. We measure the relative change between the preceding and adapted representations,

$$a_{t,\ell} = \mathbb{E}_x \left[\frac{\left\| h_\ell(x; \theta_t^a) - h_\ell(x; \bar{\theta}_{t-1}^c) \right\|_2}{\left\| h_\ell(x; \bar{\theta}_{t-1}^c) \right\|_2 + \epsilon} \right]. \tag{54}$$

This variant reaches 33.75% on EMBER-Class and 60.27% on AZ-Class.

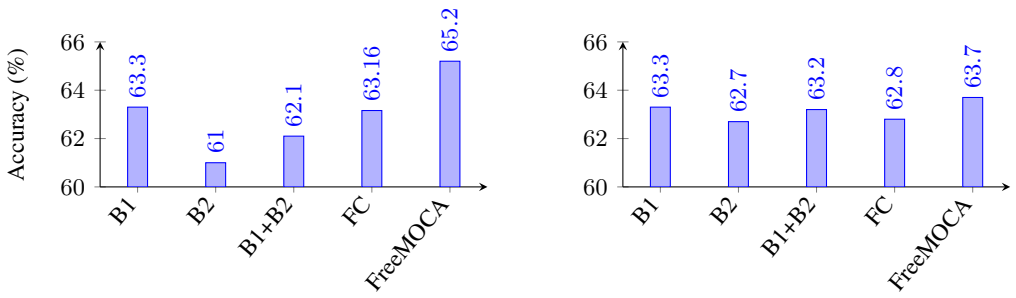
Among the adaptive variants, inverse displacement performs best, reaching 66.30% on EMBER-Class and 66.10% on AZ-Class. These results are reported as an optional extension; the core FreeMOCA method throughout the main paper uses the simpler global interpolation coefficient and obtains 65.2% and 63.7%, respectively.

Sensitivity to the Adaptive Range. The interval $[\lambda_{\min}, \lambda_{\max}]$ determines how much the layer-wise coefficients may vary. The best tested range is $[0.3, 0.7]$, reaching 66.30% on EMBER-Class and 66.10% on AZ-Class. These experiments serve only to characterize the optional adaptive extension and are not used in the core FreeMOCA results.

B.6 Selective Block Interpolation

We examine whether the benefit of consolidation is concentrated in a specific part of the network. Using the same fixed interpolation coefficient for all variants in this ablation, we interpolate Block 1, Block 2, both blocks, the classifier, and the full model while leaving the remaining parameters unchanged.

Figure 4 shows that the benefit of consolidation is distributed across the network rather than confined to one component. Full-model interpolation performs best on both datasets, reaching 65.2% on EMBER-Class and 63.7% on AZ-Class.



(a) EMBER-Class (b) AZ-Class

Figure 4: Selective block interpolation on EMBER-Class and AZ-Class. Among individual components, Block 1 performs best, while full-model interpolation gives the highest accuracy overall.

Table 9: Robustness of FreeMOCA on EMBER-Class to initialization, optimization, learning-rate scheduling, and backbone architecture.

Initialization		**Optimizer**		**LR Scheduler**	
Setting	**Acc.**	**Setting**	**Acc.**	**Setting**	**Acc.**
Seed 1	64.19	SGD	**69.09**	StepLR	64.59
Seed 2	65.10	Adam	61.78	MultiStepLR	64.69
Seed 3	66.32	AdamW	64.29	CosineAnnealingLR	**68.27**
Seed 4	68.79				
Seed 5	69.21				

Backbone Architecture					
Backbone	**None**	**FreeMOCA**	**Backbone**	**None**	**FreeMOCA**
ViT	18.23	**60.43**	10-layer CNN	16.02	**56.66**
CKAN	13.42	**52.01**			

B.7 Backbone and Optimization Robustness

We evaluate whether FreeMOCA is sensitive to random initialization, optimization choices, and backbone architecture on EMBER-Class.

Initialization. Across five random initializations, performance remains within a similar range, showing that FreeMOCA does not depend on a particular starting point.

Optimization Choices. Performance varies with the optimizer and learning-rate schedule. SGD reaches 69.09% Average Accuracy, compared with 61.78% for Adam and 64.29% for AdamW. Among the tested schedulers, CosineAnnealingLR reaches 68.27%, while StepLR and MultiStepLR obtain 64.59% and 64.69%, respectively. Thus, optimization affects absolute accuracy, but the benefit of consolidation persists across the tested settings.

Backbone Architecture. We also replace the default three-layer CNN with ViT, a ten-layer CNN, and CKAN (Bodner et al., 2024). FreeMOCA improves sequential fine-tuning from 18.23% to 60.43% with ViT, from 16.02% to 56.66% with the ten-layer CNN, and from 13.42% to 52.01% with CKAN. Since other CL baselines are not re-evaluated on these architectures, we restrict this comparison to sequential fine-tuning.