LANCE: Low Rank Activation Compression for Efficient On-Device Continual Learning

Marco P. E. Apolinario[1,2] and Kaushik Roy[2]

[1] Delft University of Technology, Delft 2628 CD, Netherlands
[2] Purdue University, West Lafayette 47906, IN, USA
m.apolinariolainez@tudelft.nl, kaushik@purdue.edu

Abstract. On-device learning is essential for personalization, privacy, and long-term adaptation in resource-constrained environments. Achieving this requires efficient learning, both fine-tuning existing models and continually acquiring new tasks without catastrophic forgetting. Yet both settings are constrained by high memory cost of storing activations during backpropagation. Existing activation compression methods reduce this cost but rely on repeated low-rank decompositions, introducing computational overhead. Also, such methods have not been explored for continual learning. We propose LANCE (Low-rank Activation Compression), a framework that performs one-shot higher-order Singular Value Decomposition (SVD) to obtain a reusable low-rank subspace for activation projection. This eliminates repeated decompositions, reducing both memory and computation. Moreover, fixed low-rank subspaces further enable on-device continual learning by allocating tasks to orthogonal subspaces without storing large task-specific matrices. Experiments show that LANCE reduces activation storage by 25× to 380× depending on the architecture, while maintaining accuracy comparable to full backpropagation on CIFAR-10/100, Oxford-IIIT Pets, Flowers102, and CUB-200 datasets. On continual learning benchmarks (Split CIFAR-100, Split MiniImageNet, 5-Datasets), it performs competitively with orthogonal gradient projection methods at a fraction of the memory cost. These results position LANCE as a practical and scalable solution for efficient fine-tuning and continual learning on edge devices.[3]

1 Introduction

On-device learning can be a key enabler for personalization, privacy preservation, and rapid adaptation in resource-constrained environments. Unlike cloud-based training, on-device learning allows models to adapt directly on hardware such as smartphones, IoT devices, or embedded systems, ensuring that data remain local and inference remains low-latency [34]. Crucially, many real-world scenarios require not only efficient fine-tuning but also continual learning, where models incrementally acquire new tasks without catastrophic forgetting [14, 18, 22, 41]. The combination of efficiency and adaptability is essential for edge devices that

[3] Our code is available at https://github.com/mapolinario94/LANCE

must operate over long lifetimes while processing evolving user data streams. Yet, both fine-tuning and continual learning are severely constrained by the large memory footprint of storing intermediate activations during backpropagation. For modern deep neural networks, activation memory often exceeds parameter memory by 5–10× [6, 26], making training infeasible on low-power devices such as microcontrollers with only a few hundred kilobytes of Static Random Access Memory (SRAM) [1, 26].

Several approaches have been proposed to address this issue. Checkpointing [10] reduces memory usage at the cost of extra recomputation; reversible networks [13] eliminate the need to store activations but require specialized architectures; and quantization or activation pruning [5, 9, 17] compress or sparsify activations but may introduce approximation errors or hardware overhead. Alternatively, bio-inspired methods have explored efficient replacements for BP, such as forward-forward methods [11, 16] and local learning rules [3, 4, 12, 24, 32]. However, they usually incur accuracy degradation. More recently, low-rank approaches exploit redundancy in activations using SVD or Tucker decompositions [29,30], but these typically recompute decompositions at each training step, adding significant overhead and making them unsuitable for continual learning, where repeated factorization across tasks would require storing large task-specific matrices.

A key insight from orthogonal gradient projection methods in continual learning [2, 23, 35] is that neural activations often lie in a stable, low-rank subspace that can be reused across training. This suggests a natural question: *Can we identify a compact activation subspace and reuse it throughout fine-tuning, while simultaneously enabling efficient continual learning?*

With the above in mind, we introduce LANCE (Low-rank Activation Compression), a framework that applies one-time higher-order singular value decomposition (HOSVD) to extract a reusable low-rank activation subspace, as shown in Figure 1b. Unlike prior iterative approaches [29,30], LANCE computes the decomposition only once at the beginning of training. Subsequent activations are projected onto this fixed subspace throughout training, avoiding repeated decompositions, thereby significantly reducing memory usage, computational cost, and hardware overhead. In contrast to system-level methods such as TinyTL [6] or the 256KB training engine [26], which freeze layers or restrict updates to fit memory budgets, LANCE enables full backpropagation across all layers by directly compressing stored activations. Importantly, these fixed subspaces also allow assigning new tasks to orthogonal components, inherently supporting on-device continual learning without storing large task-specific matrices as in previous continual learning (CL) methods [2, 27, 28, 35, 36, 39, 42]. By eliminating repeated decompositions while preserving full-model fine-tuning, LANCE provides a practical and scalable solution for memory- and energy-efficient on-device continual learning.

Our main contributions are summarized as follows:

- We propose LANCE, a one-shot activation compression framework that constructs a reusable low-rank subspace via HOSVD, avoiding repeated decompositions.
- We show that LANCE enables both efficient fine-tuning and continual learning on edge devices by projecting activations onto fixed low-rank subspaces, which can be partitioned across tasks to mitigate forgetting.
- We empirically validate LANCE on fine-tuning benchmarks (CIFAR-10/100, Pets, Flowers102, CUB-200 with MCUNet, MobileNetV2, ResNet18/34) and continual learning benchmarks (Split CIFAR-100, Split MiniImageNet, 5-Datasets), demonstrating substantial memory savings while remaining within 2% of full backpropagation on standard residual networks, and presenting a controlled 3% to 13% Pareto trade-off on ultra-lightweight mobile models, and achieving competitive performance with full-rank gradient projection methods for continual learning at a fraction of the memory cost.

2 Methodology

In this section, we outline our methodology. We first describe the main memory bottleneck for on-device learning with full backpropagation (BP), then review higher-order singular value decomposition (HOSVD), and finally present our proposed method, including an overview, step-by-step pseudocode, a complexity analysis, and a convergence analysis.

2.1 Preliminaries

Training deep neural networks requires storing intermediate *activations* during the forward pass so that they can be reused for gradient computation in backpropagation, as illustrated in Figure 1a. Let $f(\mathbf{x}; \boldsymbol{\theta})$ denote a neural network with input $\mathbf{x}$ and parameters $\boldsymbol{\theta}$. During forward propagation, each layer l produces an activation tensor $\mathbf{X}^{(l)}$ that is stored in SRAM for later use in computing the gradient: $\nabla_{\boldsymbol{\theta}^{(l)}} \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \mathbf{X}^{(l)}} \frac{\partial \mathbf{X}^{(l)}}{\partial \boldsymbol{\theta}^{(l)}}$, where $\mathcal{L}$ is the loss function. The cumulative memory footprint of storing $\{\mathbf{X}^{(l)}\}_{l=1}^{L}$ across all L layers often dominates SRAM usage, especially in convolutional networks. For example, on CIFAR-100 with ResNet18, storing activations during training can consume up to 5× more memory than storing weights, quickly exhausting the limited resources of edge devices. A natural way to mitigate this bottleneck is to exploit the *low-rank structure* of activations. Instead of storing the full tensor $\mathbf{X}^{(l)}$, we can approximate it with a compressed representation and reconstruct (or use) it during backpropagation. While classical singular value decomposition (SVD) is defined for matrices, activations are inherently higher-order (e.g., batch × channels × height × width), making tensor decompositions more suitable.

Higher-Order Singular Value Decomposition (HOSVD). HOSVD (or N-mode SVD) provides a principled way to decompose a tensor into mode-specific subspaces. Given an order-d activation tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, HOSVD expresses

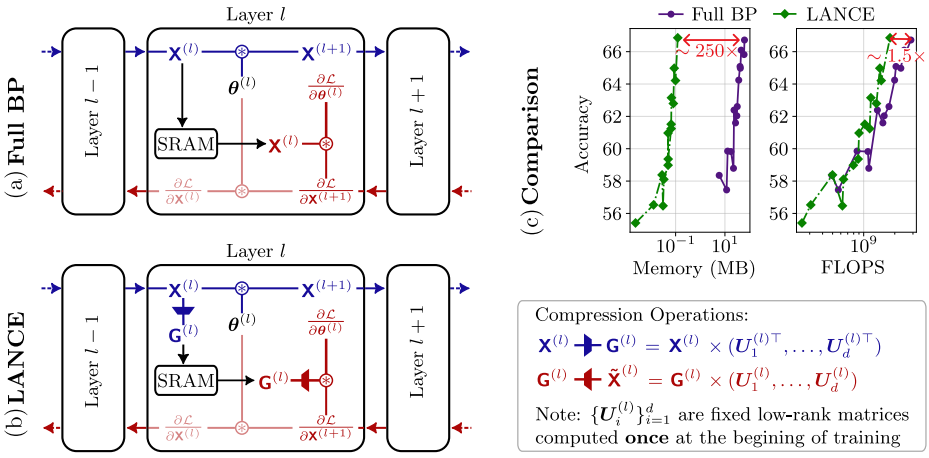


Fig. 1: Overview of LANCE for on-device training. (a) In full backpropagation (BP), the entire activation tensor $\mathbf{X}^{(l)}$ must be stored in memory for computing gradients, leading to a large memory footprint. (b) LANCE replaces $\mathbf{X}^{(l)}$ with a compressed core tensor $\mathbf{G}^{(l)}$, obtained via one-shot HOSVD using fixed low-rank matrices $\{\boldsymbol{U}_i^{(l)}\}_{i=1}^d$ computed once at the beginning of training. Only $\mathbf{G}^{(l)}$ is stored, while the factors are reused during the backward pass. (c) Pareto comparisons show that LANCE reduces memory (SRAM) usage by up to ∼250× and FLOPs by ∼1.5× relative to vanilla BP, while maintaining accuracy.

it as $\mathbf{X} = \mathbf{G} \times (\boldsymbol{U}_1, \boldsymbol{U}_2, \cdots, \boldsymbol{U}_d)$, where $\mathbf{G} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ is the core tensor, $\boldsymbol{U}_i \in \mathbb{R}^{n_i \times r_i}$ are orthonormal factor matrices for each mode i, and $\times$ denotes the tensor–matrix product. The ranks $\{r_i\}_{i=1}^d$ control the compression level, with $r_i \ll n_i$ in the low-rank regime. To compute HOSVD, the tensor $\mathbf{X}$ is *unfolded* along each mode i into a matrix $\boldsymbol{X}_i \in \mathbb{R}^{n_i \times \prod_{j \neq i} n_j}$, followed by a matrix SVD $\boldsymbol{X}_i = \boldsymbol{U}_i \boldsymbol{\Sigma}_i \boldsymbol{V}_i^\top$. Truncating to the top-$r_i$ singular vectors yields $\boldsymbol{U}^{(i)}$, which captures the dominant subspace of mode i. Thus, HOSVD produces a compact multi-linear approximation of $\mathbf{X}$ that preserves most of its structure while discarding redundancy.

2.2 Proposed Method: LANCE

To address the challenges of training DNNs on low-power devices, we propose LANCE, a one-shot HOSVD-based low-rank activation compression method for efficient on-device learning. LANCE directly targets the primary memory bottleneck in backpropagation: the need to store intermediate activations $\mathbf{X}^{(l)}$ from hidden layers. By projecting activations into a compact low-rank subspace, LANCE reduces both the memory required to store $\mathbf{X}^{(l)}$ and the number of computations needed to backpropagate through them.

Formally, given projection matrices $\{\boldsymbol{U}_i\}_{i=1}^d$ obtained via HOSVD, each activation tensor $\mathbf{X}^{(l)} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is compressed into a smaller core tensor

Algorithm 1 LANCE: Low-Rank Activation Compression for On-device Continual Learning

Require: Pretrained network $f(\mathbf{x};\boldsymbol{\theta})$ with layers $l{=}1,\dots,L$; calibration batches $\{\mathcal{B}_j\}_{j=1}^{N}$; energy thresholds ε (LANCE) and ε_{CL} (CL); *(CL only)* memory $\boldsymbol{M}^{t-1}$ from previous task

Notation: we omit layer superscripts for readability; all steps are applied per layer.
Phase I: One-Shot Subspace Calibration (offline)
for each mode $i \in \{1,\dots,d\}$ **do**
  Estimate covariance $\boldsymbol{B}_i$ from calibration activations via Equation (2)
  if $i{=}d$ **then**
    $\hat{\boldsymbol{B}}_d \leftarrow (\boldsymbol{I} - \boldsymbol{M}^{t-1}\boldsymbol{M}^{t-1\top})\,\boldsymbol{B}_d$ *// For non-CL or $t=1$, $\boldsymbol{M}^{t-1}$ is a zero matrix*
    Compute SVD, $\hat{\boldsymbol{B}}_d = \hat{\boldsymbol{U}}_d\hat{\boldsymbol{\Sigma}}_d\hat{\boldsymbol{U}}_d^{\top}$, and $\boldsymbol{B}_d = \boldsymbol{U}_d\boldsymbol{\Sigma}_d\boldsymbol{U}_d^{\top}$
    Retain top r_d columns of $\hat{\boldsymbol{U}}_d$ satisfying Equation (5); set $\boldsymbol{U}_d \leftarrow \hat{\boldsymbol{U}}_d[:,1{:}r_d]$
  else
    Compute SVD $\boldsymbol{B}_i = \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{U}_i^{\top}$
    Retain top r_i columns of $\boldsymbol{U}_i$ satisfying Equation (3)
  end if
end for
Phase II: Fine-Tuning with Low-Rank Activations (on-device)
for each minibatch $\mathcal{B}$ **do**
  Forward: project $\boldsymbol{X}^{(l)}$ to cores $\mathbf{G}^{(l)} = \mathbf{X}^{(l)} \times (\boldsymbol{U}_1^{\top},\dots,\boldsymbol{U}_d^{\top})$; store $\mathbf{G}^{(l)}$ only
  Backward: compute $\nabla_{\boldsymbol{\theta}^{(l)}}\mathcal{L}$ using $\mathbf{G}^{(l)} \times (\boldsymbol{U}_1,\dots,\boldsymbol{U}_d)$; update $\boldsymbol{\theta}^{(l)}$
end for
Phase III (CL only): Memory Construction & Update (offline, after task t)
Estimate last-mode covariance $\boldsymbol{B}_d^t$ from N_{CL} calibration mini-batches as in Equation (2)
Compute $\hat{\boldsymbol{B}}_d^t \leftarrow (\boldsymbol{I} - \boldsymbol{M}^{t-1}\boldsymbol{M}^{t-1\top})\,\boldsymbol{B}_d^t$, then SVD on $\hat{\boldsymbol{B}}_d^t$ and $\boldsymbol{B}_d^t$
Select r_d via Equation (5); and update memory: $\boldsymbol{M}^t \leftarrow \text{orth}\big([\boldsymbol{M}^{t-1}\ \ \hat{\boldsymbol{U}}^t_{d,[:,1:r_d]}]\big)$

$\mathbf{G}^{(l)} \in \mathbb{R}^{r_1\times r_2\times\cdots\times r_d}$:

$$\mathbf{G}^{(l)} \;=\; \mathbf{X}^{(l)} \times (\boldsymbol{U}_1^{(l)\top}, \boldsymbol{U}_2^{(l)\top}, \cdots, \boldsymbol{U}_d^{(l)\top}). \tag{1}$$

The compressed $\mathbf{G}^{(l)}$ is saved instead of the full activation and reused during backpropagation. Unlike prior works that recompute low-rank decompositions every iteration, LANCE computes the projection matrices only once at the beginning of fine-tuning and reuses them across all epochs. This one-shot approach avoids expensive per-iteration SVDs and is particularly well-suited for energy-constrained devices. The method, summarized in Algorithm 1, has two phases: (1) a one-shot subspace calibration performed offline, and (2) on-device fine-tuning with low-rank activations, which projects activations into low-rank subspaces during the forward pass and computes gradients during the backward pass using the compressed activations (Figure 1b). Each step is detailed below.

Phase I: One-Shot Subspace Calibration (offline). We estimate the subspace for each hidden layer l using activations from N calibration mini-batches passed through the pretrained model. For each mode i, we maintain a running covariance estimate:

$$\boldsymbol{B}_i[t] \;=\; \tfrac{1}{t}\Big((t-1)\boldsymbol{B}_i[t-1] \;+\; \tfrac{1}{\prod_{j\neq i} n_j}\boldsymbol{X}_i^{(l)}\boldsymbol{X}_i^{(l)\top}\Big), \tag{2}$$

with $\boldsymbol{B}_i[0] = 0$, where $\boldsymbol{X}_i^{(l)}$ is the mode-i unfolding of $\mathbf{X}^{(l)}$. After N batches, we compute an SVD decomposition, $\boldsymbol{B}_i = \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{U}_i^\top$. We then truncate to the top-r_i singular vectors according to an energy threshold ε:

$$\frac{\sum_{j=1}^{r_i}\sigma_j}{\sum_{j=1}^{n_i}\sigma_j} \;\geq\; \varepsilon, \tag{3}$$

where σ_j is the j-th singular value of $\boldsymbol{B}_i$. This ensures that the subspace captures at least an ε fraction of activation variance. Because this decomposition is computed only once (before fine-tuning) and amortized across training, its cost is negligible relative to repeated decompositions.

Phase II: Fine-Tuning with Low-Rank Activations (on-device). During fine-tuning with BP (Figure 1b), we distinguish the forward and backward passes.

Forward pass. Each activation $\mathbf{X}^{(l)}$ is projected into the learned low-rank subspace using the precomputed matrices $\{\boldsymbol{U}_i^{(l)}\}_{i=1}^d$, following Equation (1) to obtain a core tensor $\mathbf{G}^{(l)}$. Only the compact tensor $\mathbf{G}^{(l)}$ and the fixed $\{\boldsymbol{U}_i^{(l)}\}_{i=1}^d$ need to be stored for backpropagation. This reduces the memory footprint from $\mathcal{O}(\prod_{i=1}^d n_i)$ to $\mathcal{O}(\prod_{i=1}^d r_i + \sum_{i=1}^d n_i r_i)$ per activation.

Backward pass. The gradient with respect to the weights is computed using the compressed activations. Given the gradient of the loss with respect to the layer output, $\nabla_{\mathbf{X}^{(l+1)}}\mathcal{L}$, we reconstruct the contribution with respect to the layer's parameters $\boldsymbol{\theta}^{(l)}$ by reversing the projections:

$$\nabla_{\boldsymbol{\theta}^{(l)}}\tilde{\mathcal{L}} \;=\; \frac{\partial\mathcal{L}^\top}{\partial\mathbf{X}^{(l+1)}}\left(\mathbf{G}^{(l)} \times (\boldsymbol{U}_1^{(l)}, \cdots, \boldsymbol{U}_d^{(l)})\right). \tag{4}$$

Because $\{\boldsymbol{U}_i^{(l)}\}_{i=1}^d$ are orthonormal, this operation preserves gradient directions up to the truncation error introduced by the low-rank approximation. We prove that the approximated gradient $-\nabla_{\boldsymbol{\theta}^{(l)}}\tilde{\mathcal{L}}$ is a descent direction and that the overall method converges in Sec. 2.4. Finally, gradients with respect to activations, $\nabla_{\mathbf{X}^{(l)}}\mathcal{L}$, are computed as usual by automatic differentiation; the compression only affects which forward tensors are stored (cores $\mathbf{G}^{(l)}$ vs. full $\mathbf{X}^{(l)}$), not the functional form of the backward.

2.3 LANCE for On-Device Continual Learning

We extend LANCE to the continual learning (CL) setting, where tasks arrive sequentially $t = 1, 2, \ldots, T$. After completing task t, we retain for each layer a

compact set of *important directions* on the last mode (the input dimension of the layer's weights), denoted by $\boldsymbol{M}^t$. When calibrating the HOSVD for the next task $t{+}1$, the new columns of $\boldsymbol{U}_d$ (the mode-d factors) are chosen to lie strictly in the *null space* of $\boldsymbol{M}^t$. This ensures that task-$t{+}1$ learns from directions orthogonal to those already used by previous tasks, thereby preventing interference and mitigating forgetting.

For clarity, we omit the explicit layer superscript in the following notation; all operations (memory construction, calibration, and updates) are applied independently at each layer.

Phase III: Memory construction and update. At the end of training on task t, we estimate the subspace $\boldsymbol{M}^t$ by accumulating the covariance of activations along the last mode, $\boldsymbol{B}_d^t$, as in Equation (2) using N_{CL} calibration mini-batches. An SVD decomposition of $\boldsymbol{B}_d^t$ yields a basis of principal directions, and we retain the top r_d components that capture at least an energy fraction ε_{CL}. To ensure orthogonality with previously stored directions, we project $\boldsymbol{B}_d^t$ onto the complement of $\boldsymbol{M}^{t-1}$: $\hat{\boldsymbol{B}}_d^t = \boldsymbol{B}_d^t - \boldsymbol{M}^{t-1}\boldsymbol{M}^{t-1\top}\boldsymbol{B}_d^t$, where $\boldsymbol{M}^0$ is initialized as empty. We select the smallest r_d such that:

$$\frac{\sum_{j=1}^{n_d}\sigma_j - \sum_{j=r_d}^{n_d}\hat{\sigma}_j}{\sum_{j=1}^{n_d}\sigma_j} \;\geq\; \varepsilon_{\text{CL}}. \tag{5}$$

The corresponding top-r_d eigenvectors form $\hat{\boldsymbol{U}}_d^t$, which are merged with the previous memory to update: $\boldsymbol{M}^t \leftarrow \text{orth}\big([\boldsymbol{M}^{t-1}\ \hat{\boldsymbol{U}}^t_{d,[:,1:r_d]}]\big)$.

Null-space constrained calibration. For a new task $t{+}1$, we update the one-shot calibration of LANCE by incorporating the memory constraint only in mode-d; all other modes follow the procedure in Section 2.2. We compute the covariance $\boldsymbol{B}_d$ for the new task, remove contributions along $\boldsymbol{M}^t$, $\hat{\boldsymbol{B}}_d = \boldsymbol{B}_d - \boldsymbol{B}_d\boldsymbol{M}^t\boldsymbol{M}^{t\top}$, and select the top r_d eigenvectors of $\hat{\boldsymbol{B}}_d$ that satisfy an energy threshold ε similar to Equation (5). The resulting low-rank factors used during fine-tuning are $\{\boldsymbol{U}_i^{(l)}\}_{i=1}^{d-1} \cup \hat{\boldsymbol{U}}^{(l)}_{d,[:,1:r_d]}$, where $\hat{\boldsymbol{U}}^{(l)}_{d,[:,1:r_d]}$ are the retained singular vectors of $\hat{\boldsymbol{B}}_d$. This guarantees that activations of the new task are compressed into a subspace orthogonal to those of previous tasks, thereby reducing catastrophic forgetting.

Discussion. Both memory update and null-space constrained calibration are performed *offline*, so they do not affect the efficiency of on-device fine-tuning. The method requires storing only the compact memory matrices $\{\boldsymbol{M}^{(l)}\}$ and applying inexpensive projections during calibration. By construction, interference with past tasks is eliminated along the stored subspaces, and forgetting is mitigated without replay buffers or large task-specific models. In contrast to prior gradient-projection methods [2, 27, 28, 35, 36], which explicitly use $\{\boldsymbol{M}^{(l)}\}$ for online gradient projection and must store such matrices in SRAM during training, our approach achieves continual learning naturally by projecting activations into fixed low-rank subspaces that lie in the null space of previous tasks, retaining LANCE's memory efficiency.

2.4 Convergence Analysis

We provide a brief convergence analysis showing that LANCE yields valid descent directions and converges to projected stationary points. Full proofs are deferred to Appendix B.

Assumption A1 (Orthogonal truncation). *For every layer l and mode i, the calibration bases $\boldsymbol{U}_i^{(l)}$ have orthonormal columns; hence $\mathsf{P}^{(l)} := \boldsymbol{U}_1^{(l)}\boldsymbol{U}_1^{(l)\top} \otimes \cdots \otimes \boldsymbol{U}_d^{(l)}\boldsymbol{U}_d^{(l)\top}$ is an orthogonal projector.*

Assumption A2 (Smoothness). *The empirical loss $\mathcal{L}(\boldsymbol{\theta})$ is L-smooth.*

Assumption A3 (Layer linearity in inputs). *Each trainable layer is linear in its input during backpropagation (dense or convolutional after unfolding). Nonlinearities are piecewise linear and treated as fixed locally.*

Assumption A4 (Energy capture). *One-shot HOSVD retains a fraction $\varepsilon \in (0, 1]$ of activation energy per mode, inducing a bound on gradient leakage outside the retained input subspace (used in Proposition 1).*

Theorem 1 (Projected gradient & descent). *Under Assumptions A1 and A3, for any layer l with input projector $\mathsf{P}^{(l)}$, the LANCE weight gradient equals the right-Frobenius orthogonal projection of the full gradient: $\nabla_{\boldsymbol{W}}\mathcal{L}_{\mathrm{LANCE}} = \nabla_{\boldsymbol{W}}\mathcal{L}_{\mathrm{full}}\,\mathsf{P}^{(l)}$. Consequently, $\langle \nabla_{\boldsymbol{W}}\mathcal{L}_{\mathrm{LANCE}}, \nabla_{\boldsymbol{W}}\mathcal{L}_{\mathrm{full}}\rangle = \|\nabla_{\boldsymbol{W}}\mathcal{L}_{\mathrm{LANCE}}\|_F^2 \geq 0$, so $-\nabla_{\boldsymbol{W}}\mathcal{L}_{\mathrm{LANCE}}$ is a descent direction unless it vanishes.*

Theorem 2 (Monotone decrease & projected stationarity). *Let Q be the block projector that right-multiplies each layer's weight gradient by its $\mathsf{P}^{(l)}$. Under Assumption A2, the LANCE update $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta\,\mathsf{Q}\,\nabla\mathcal{L}(\boldsymbol{\theta}_k)$ with $\eta \in (0, 1/L]$ satisfies $\mathcal{L}(\boldsymbol{\theta}_{k+1}) \leq \mathcal{L}(\boldsymbol{\theta}_k) - \frac{\eta}{2}\|\mathsf{Q}\nabla\mathcal{L}(\boldsymbol{\theta}_k)\|_2^2$, hence $\|\mathsf{Q}\nabla\mathcal{L}(\boldsymbol{\theta}_k)\|_2 \to 0$ and every limit point $\boldsymbol{\theta}_\star$ obeys $\mathsf{Q}\nabla\mathcal{L}(\boldsymbol{\theta}_\star) = \mathbf{0}$.*

Proposition 1 (Stationarity gap vs. truncation). *If along the iterates $\|(I - \mathsf{Q})\nabla\mathcal{L}(\boldsymbol{\theta})\|_2 \leq C\sqrt{1-\varepsilon}$ for some $C > 0$, then any limit point $\boldsymbol{\theta}_\star$ satisfies $\|\nabla\mathcal{L}(\boldsymbol{\theta}_\star)\|_2 \leq C\sqrt{1-\varepsilon}$.*

Insight. Together, these results show that LANCE behaves like full BP restricted to the low-rank activation subspace: updates are valid descent directions, the loss decreases monotonically, and iterates converge to projected stationary points. The residual gap depends only on the energy threshold ε, so higher ε yields solutions closer to true stationary points, while lower ε trades accuracy for efficiency in a quantifiable way.

3 Experimental Evaluation

We evaluate LANCE in two complementary settings: (i) *single-task fine-tuning*, where a pretrained model is adapted to a target dataset on an edge device, and (ii) *continual learning*, where multiple tasks arrive sequentially and must be learned without catastrophic forgetting. Our experiments aim to answer the following questions:

Q1 Does the one-shot subspace calibration preserve gradient fidelity in practice?
Q2 How effective is LANCE at reducing activation memory and computational cost during on-device fine-tuning, while maintaining accuracy?
Q3 Can LANCE leverage fixed low-rank subspaces to achieve competitive performance on continual learning benchmarks, while offering superior memory efficiency?

Experimental Setup. (i) Datasets: For single-task fine-tuning, we use CIFAR-10/100 [20], Oxford-IIIT Pets [33], Flowers-102 [31], CUB-200 [40], and ImageNet [21], covering a range of dataset sizes and complexities. For continual learning, we evaluate on Split CIFAR-100 (20 tasks), Split MiniImageNet (20 tasks), and the 5-Datasets benchmark (CIFAR-10, MNIST, SVHN, Fashion-MNIST, and notMNIST), following common CL protocols. (ii) Metrics: We report the following metrics: classification accuracy in the target datasets; memory usage, measured as peak activation storage in MB (estimated in Appendix C); and computational cost in FLOPs. For continual learning benchmarks, we additionally report the average final accuracy over all tasks (ACC) and the Backward Transfer (BWT), which quantifies the forgetting of previously learned tasks when new tasks are introduced, as defined in Appendix A.2.

3.1 Results: Single-task Fine-Tuning Efficiency

We first evaluate LANCE on fine-tuning the last 2 or 4 layers of MCUNet [25], MobileNetV2 [37], ResNet18 [15] and ResNet34 [15]. All models are pretrained on ImageNet-1k. For LANCE, the offline Phase I calibration uses N=100 mini-batches with an energy threshold ε=0.7. Fine-tuning runs for 50 epochs.

Answer to Q1. To assess gradient fidelity, we compute the angle between LANCE gradients ($\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{LANCE}}$) and full BP gradients ($\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\text{BP}}$), normalized as $\nabla_{\boldsymbol{\theta}}\mathcal{L}/||\nabla_{\boldsymbol{\theta}}\mathcal{L}||_F$, on one mini-batch at the start of each epoch. Figure 2 shows that LANCE gradients remain within $\sim$70° of the full gradients, with angles stabilizing over time. This empirically validates Theorem 1, confirming that LANCE preserves gradient fidelity and maintains valid descent directions. This observation is consistent with the ablation in Sec. 3.3, where gradient fidelity remains stable across a wide range of compression ratios controlled by the energy threshold ε.

Answer to Q2. Across all datasets and models, LANCE reduces activation storage by up to 250× relative to BP while maintaining competitive accuracy. Table 2 reports results on CIFAR-10/100 and Pets and CUB200. Compared

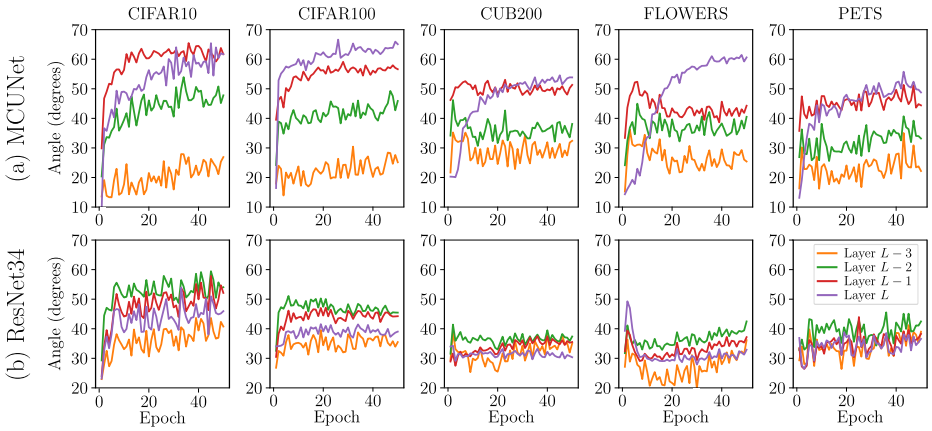


Fig. 2: Gradient alignment between full BP and LANCE. We plot the angle between true gradients and LANCE-projected gradients across epochs for different fine-tuning tasks. LANCE consistently produces gradients within 70° of the true gradient, and angles stabilize as training progresses, indicating preserved descent directions.

Table 1: ImageNet-scale fine-tuning on Split ImageNet-1k. We pretrain each model on 500 ImageNet classes (Split A) and fine-tune on the remaining 500 classes (Split B).

	ResNet18			ResNet34		
Method	Acc ↑	MB ↓	TFLOPs ↓	Acc ↑	MB ↓	TFLOPs ↓
BP (2 layers)	60.70	24.50	0.059	60.77	24.50	0.059
BP (4 layers)	64.28	61.25	0.090	64.21	49.00	0.118
LANCE ($\epsilon = 0.8$, 2 layers)	57.04	2.48	0.039	56.57	1.58	0.038
LANCE ($\epsilon = 0.8$, 4 layers)	59.76	5.79	0.062	60.56	3.87	0.078
LANCE ($\epsilon = 0.9$, 2 layers)	59.18	7.12	0.048	58.59	4.49	0.046
LANCE ($\epsilon = 0.9$, 4 layers)	62.20	16.59	0.075	62.66	10.79	0.093

to iterative low-rank methods such as ASI [29], or HOSVD [30] where a full SVD is recomputed at each step, LANCE achieves comparable compression with higher accuracy and similar FLOPs, confirming that repeated decompositions are unnecessary. Although BP attains higher accuracy when the same number of layers are fine-tuned, LANCE's extreme memory efficiency enables fine-tuning more layers under strict SRAM budgets. For example, fine-tuning 4 layers with LANCE reaches the accuracy of 2-layer BP while still fitting within microcontroller constraints. These results demonstrate that fixed low-rank subspaces retain sufficient representational power for efficient on-device fine-tuning.

Scalability to ImageNet-Scale Fine-Tuning. To assess the scalability of LANCE to larger datasets and higher-capacity models, we conduct an ImageNet-scale experiment following the protocol used in prior work [29]. We split ImageNet-1k into two disjoint subsets of 500 classes (Split A and Split B). A model is first pretrained on Split A and then fine-tuned on Split B using either full BP or

Table 2: Performance and efficiency comparison. Results are reported in terms of accuracy, memory usage, and TFLOPs. † denote the results from the respective original paper.

Mdl.	Method	# Layers	CIFAR10			CIFAR100			Pets			CUB200		
			Acc ↑	MB ↓	TFLOPS ↓	Acc ↑	MB ↓	TFLOPS ↓	Acc ↑	MB ↓	TFLOPS ↓	Acc ↑	MB ↓	TFLOPS ↓
MCUNet	BP	2	75.85	11.71	0.000	52.47	11.71	0.000	56.30	11.71	0.000	33.05	11.71	0.000
		4	82.91	17.57	0.001	58.38	17.57	0.001	60.01	17.57	0.001	34.13	17.57	0.001
	HOSVD	2	70.91	0.04	1.930	48.00	0.05	1.930	56.11	0.06	1.93	31.22	0.06	1.93
		4	75.63	0.12	2.590	52.34	0.10	2.590	57.89	0.17	2.59	31.72	0.14	2.59
	LANCE	2	70.64	0.04	0.000	47.57	0.04	0.000	57.26	0.06	0.000	31.44	0.05	0.000
		4	76.23	0.09	0.000	52.59	0.09	0.000	58.05	0.14	0.000	30.87	0.13	0.000
MobileNetV2	BP	2	91.72	30.62	0.01	73.43	30.62	0.01	89.94	30.62	0.01	64.37	30.62	0.01
		4	92.23	57.42	0.01	74.69	57.42	0.01	90.32	57.42	0.01	65.86	57.42	0.01
	HOSVD	2	85.67	0.13	11.87	67.28	0.14	11.87	89.58	0.13	11.87	58.11	0.13	11.87
		4	89.03	0.16	22.86	69.33	0.15	22.86	89.91	0.23	22.86	60.30	0.16	22.86
	ASI †	2	85.09	0.26	0.01	61.03	0.15	0.01	88.13	0.15	0.01	46.44	0.26	0.01
		4	88.03	0.63	0.06	66.18	0.63	0.06	88.91	0.76	0.06	50.69	0.63	0.06
	LANCE	2	84.16	0.15	0.01	60.69	0.16	0.01	89.69	0.14	0.01	58.18	0.14	0.01
		4	88.99	0.17	0.01	68.79	0.17	0.01	90.40	0.19	0.01	61.21	0.15	0.01
ResNet18	BP	2	92.69	24.5	0.05	75.69	24.5	0.06	89.17	24.5	0.06	62.28	24.5	0.06
		4	94.01	61.25	0.09	77.07	61.25	0.09	89.07	61.25	0.09	60.80	61.25	0.09
	HOSVD	2	92.17	0.71	6.12	74.57	0.72	6.12	89.47	0.92	6.12	60.02	0.73	6.12
		4	93.16	1.16	15.56	75.35	1.21	15.56	88.71	1.77	15.56	59.26	1.41	15.56
	ASI †	2	90.77	1.24	0.04	69.70	0.90	0.04	88.75	2.01	0.04	56.77	1.51	0.04
		4	91.90	1.89	0.06	71.93	1.61	0.06	88.44	4.34	0.07	55.73	3.50	0.07
	LANCE	2	91.8	0.75	0.03	74.09	0.77	0.03	89.26	0.94	0.03	59.87	0.73	0.03
		4	92.89	1.11	0.05	75.7	01.16	0.05	89.12	1.78	0.05	59.50	1.36	0.05
ResNet34	BP	2	93.48	24.50	0.06	76.19	24.50	0.06	91.22	24.50	0.06	64.10	24.50	0.06
		4	94.37	49.00	0.12	77.50	49.00	0.12	91.60	49.00	0.12	64.22	49.00	0.12
	HOSVD	2	92.70	0.42	6.12	75.31	0.45	6.12	90.89	0.48	6.12	60.82	0.38	6.12
		4	93.80	0.92	12.25	76.01	0.95	12.25	91.11	1.10	12.25	62.28	0.84	12.25
	ASI †	2	90.09	0.49	0.03	69.66	0.44	0.03	91.09	0.81	0.04	58.77	0.60	0.03
		4	91.54	1.24	0.07	70.60	1.00	0.07	91.41	2.05	0.07	58.85	1.58	0.07
	LANCE	2	92.62	0.46	0.03	74.89	0.51	0.03	90.70	0.45	0.03	62.35	0.39	0.03
		4	93.56	1.01	0.06	75.94	1.02	0.06	90.92	1.05	0.06	63.01	0.80	0.06

LANCE. Table 1 reports results for ResNet18 and ResNet34 when fine-tuning the last 2 or 4 layers. The trends closely mirror those observed on smaller downstream datasets: LANCE achieves large reductions in activation memory (up to $\sim 4\times$) and maintains accuracy within $\sim$1–2% of full BP, with only modest differences in training FLOPs. These results demonstrate that LANCE remains effective and memory-efficient even at ImageNet scale, confirming that the one-shot low-rank subspace generalizes well to large, high-variance visual distributions.

3.2 Results: Continual Learning

We compare LANCE against a broad set of continual learning baselines: regularization based methods (EWC [19], HAT [38]), replay-based approaches (A-GEM [7], ER_Res [8]), and projection-based methods (GPM [35], TRGP [28], CUBER [27], SGP [36], SD [42], LMSP [39], CODE-CL [2]). For reference, we also report multitask performance, which serves as an upper bound where all tasks are trained jointly. We follow the same experimental setup as [2]: a 5-layer AlexNet for Split CIFAR-100, and a reduced ResNet18 for Split MiniImageNet

Table 3: Performance comparison on continual image classification datasets using multi-head networks. Accuracy and BWT (mean ± std) are reported over five trials. † denotes the results taken from [35] and ‡ denote the results from the respective original papers.

Method	Split CIFAR100			Split MiniImageNet			5-Datasets		
	ACC (%) ↑	BWT (%) ↑	Mem (MB) ↓	ACC (%) ↑	BWT (%) ↑	Mem (MB) ↓	ACC (%) ↑	BWT (%) ↑	Mem (MB) ↓
Multitask†	79.58 ± 0.54	–	–	69.46 ± 0.62	–	–	91.54 ± 0.28	–	–
EWC †	68.80 ± 0.88	-2 ± 1	–	52.01 ± 2.53	-12 ± 3	–	88.64 ± 0.26	-4 ± 1	–
HAT †	72.06 ± 0.50	0 ± 0	–	59.78 ± 0.57	-3 ± 0	–	91.32 ± 0.18	-1 ± 0	–
A-GEM †	63.98 ± 1.22	-15 ± 2	–	57.24 ± 0.72	-12 ± 1	–	84.04 ± 0.33	-12 ± 1	–
ER_Res †	71.73 ± 0.63	-6 ± 1	–	58.94 ± 0.85	-7 ± 1	–	80.31 ± 0.22	-4 ± 0	–
TRGP+SD ‡	75.5 ± 0.35	-0.96 ± 0.09	–	65.8 ± 0.16	-0.49 ± 0.08	–	–	–	–
LMSP ‡	74.21	+0.94	–	64.2	+1.55	–	93.78	+0.07	–
GPM †	72.48 ± 0.54	-0.9	22.27	60.41 ± 0.61	-0.9	127.37	91.22 ± 0.20	-1.0	70.33
TRGP ‡	74.64 ± 0.32	-0.9 ± 0.01	78.86	61.78 ± 0.60	-0.5 ± 0.60	543.22	93.56 ± 0.10	-0.04 ± 0.01	181.57
CUBER ‡	75.54 ± 0.22	$+0.13 \pm 0.08$	326.79	62.67 ± 0.74	$+0.23 \pm 0.15$	604.61	93.48 ± 0.10	-0.00 ± 0.02	196.91
SGP ‡	76.05 ± 0.43	-1	22.27	62.83 ± 0.33	-1	127.37	–	–	–
CODE-CL‡	77.21 ± 0.32	-1.1 ± 0.28	33.80	71.16 ± 0.32	-1.1 ± 0.3	283.82	93.51 ± 0.13	-0.11 ± 0.01	116.72
LANCE	71.52 ± 0.27	-0.17 ± 0.34	2.32	59.68 ± 1.17	-0.99 ± 0.78	32.96	90.76 ± 0.31	-1.02 ± 0.15	34.96

and the 5-Datasets benchmark. During the first task, models are trained without constraints using full BP. For subsequent tasks, the model from the previous task serves as initialization, and LANCE applies its subspace-constrained training.

Answer to Q3. LANCE achieves accuracy competitive with full-rank gradient projection methods such as GPM and CODE-CL, while using drastically less memory. As summarized in Tab. 3, on Split CIFAR-100 LANCE reaches ∼71% average accuracy (ACC) while reducing activation storage by more than 10× compared to GPM. Similar trends are observed on Split MiniImageNet and the 5-Datasets benchmark, where LANCE maintains stable performance without rehearsal memory. These results confirm that fixed low-rank subspaces are sufficient to retain representational capacity across tasks, while providing substantial memory savings for resource-constrained devices.

3.3 Ablations and Analysis

We conduct ablations on two key hyperparameters: (i) the energy threshold ε (evaluated on CUB-200), and (ii) the number of calibration batches N (evaluated on CIFAR-100 with ε=0.7). Results for MCUNet and ResNet34 are shown in Figure 3. As expected, higher ε increases accuracy but also grows memory usage exponentially, highlighting the need to balance compression and accuracy according to device memory budgets. For calibration size N, accuracy remains stable across a wide range, while memory usage grows with N. Remarkably, even N=2 calibration batches are sufficient to yield stable subspaces, underscoring the robustness of the one-shot calibration phase.

We additionally evaluate how different compression ratios affect gradient fidelity by measuring the angle between full-BP gradients and LANCE-projected gradients across several values of ε, shown in Figure 4. As expected, more aggressive compression (lower ε) leads to larger angles, but the degradation is smooth

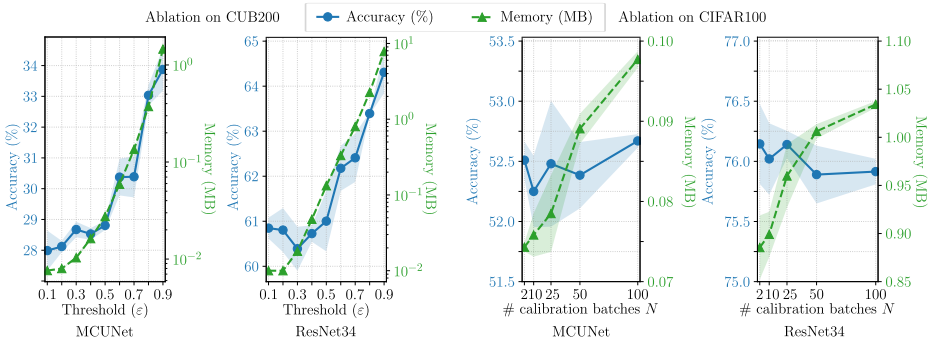
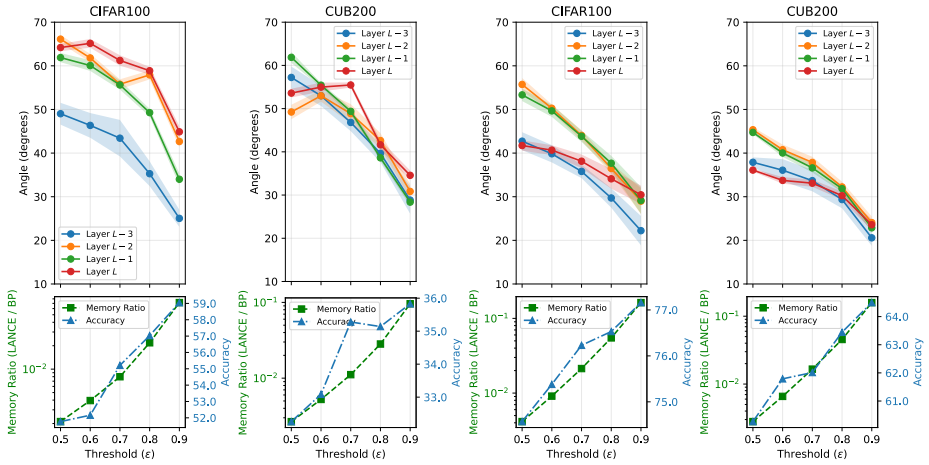


Fig. 3: Ablation studies of LANCE. (Left) Effect of the energy threshold ε on CUB-200 using MCUNet and ResNet34. Accuracy improves steadily as ε increases, but memory usage grows exponentially, illustrating the trade-off between accuracy and compression. (Right) Effect of the number of calibration batches N on CIFAR-100. Accuracy remains stable even for very small N, while memory increases with larger calibration sets. These results show that stable subspaces can be obtained with as few as N=2 calibration batches, and practical choices of ε (e.g., 0.7) provide a good balance between accuracy and memory.

and controlled. Even at the strongest compression settings, reducing activation storage by up to two orders of magnitude, LANCE maintains gradient alignment within $\sim 70°$ of full BP, indicating that the one-shot subspace preserves the dominant descent directions required for stable optimization.

3.4 Runtime Validation on Edge Hardware

While our memory analysis (Sec. 3.1) demonstrates LANCE's ability to fit within strict microcontroller SRAM budgets, we evaluated the framework on a Raspberry Pi 3 Model B+ (ARM Cortex-A53) to validate its practical runtime efficiency. Serving as an accessible edge proxy, this hardware allows us to isolate and measure the relative computational overhead of the competing methods without artificial bottlenecking. We fine-tuned an MCUNet model for five iterations on CIFAR-10 using a batch size of 128. Each configuration was run for five independent trials, and we report mean end-to-end latency across 1–10 trainable layers. As shown in Figure 5, LANCE consistently achieves the lowest forward, backward, and total training time. Relative to LANCE, BP is 3–9% slower, ASI is 19–76% slower, and iterative HOSVD is 2.1×–4.4× slower. This demonstrates that one-shot subspace calibration effectively eliminates the overhead of repeated decompositions, yielding tangible latency speedups that inherently scale down to even tighter energy-constrained microcontrollers.



(a) MCUNet (b) ResNet34

Fig. 4: Gradient fidelity across compression ratios. We vary the energy threshold ε controlling the effective memory compression of LANCE and report the angle (mean $\pm$ std) between full-BP gradients and LANCE-projected gradients over the final 10 epochs (out of 50). As ε decreases, compression becomes more aggressive and gradient alignment degrades accordingly. Nevertheless, even under memory reductions of up to two orders of magnitude, LANCE maintains gradient directions within $\sim 70^\circ$ of full BP, indicating preserved descent directions despite strong compression.

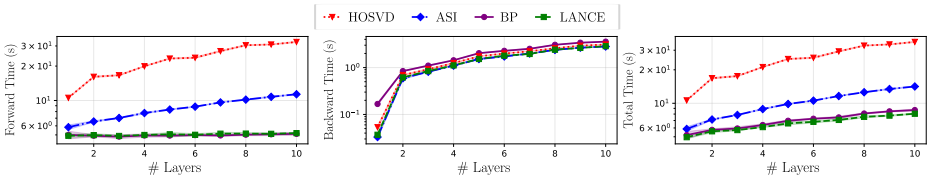


Fig. 5: End-to-end fine-tuning latency on a Raspberry Pi 3B+ (MCUNet on CIFAR-10, batch size 128). Averaged over five trials, LANCE consistently achieves the lowest forward, backward, and total training times across all layer depths.

4 Related Work

Storing intermediate activations is the primary memory bottleneck for on-device training [6, 26]. Prior work has reduced this cost through recomputation (e.g., checkpointing [10]), architectural modifications (e.g., reversible networks [13]), or lossy compression and sparsity [5,9,17], but often at the expense of extra computation, accuracy, or hardware complexity. A complementary line of work exploits low-rank structure in activations [29, 30]. Closest to our method is ASI [29], which incrementally updates subspaces during training; in contrast, LANCE performs a one-shot HOSVD at initialization and reuses the resulting subspaces throughout fine-tuning, avoiding repeated decompositions. Recent CL work such

as LMSP [39] also uses low-rank representations but focuses on reducing the computational cost of gradient projection, still requiring multiple per-task low-rank memory components; by contrast, LANCE targets the dominant activation-memory bottleneck directly. This simple design not only reduces compute and hardware overhead but also enables a continual learning extension: fixed subspaces naturally support null-space allocation across tasks without storing large task-specific matrices. System-level methods such as TinyTL [6], MCUNet [25], and the 256KB training engine [26] co-design architectures and update rules to fit extreme SRAM budgets, often by freezing layers or pruning gradient paths. LANCE is complementary to these approaches: instead of restricting the optimization, it preserves full backpropagation while directly compressing activations. Finally, in continual learning, regularization methods [19,38], replay-based approaches [7, 8], and gradient-projection methods [2, 27, 28, 35, 36, 42] all mitigate forgetting by controlling parameter updates or storing exemplars. LANCE differs by operating directly in the activation subspace: new tasks are assigned to the orthogonal complement of previously used directions during calibration. This yields performance competitive with gradient-projection methods while requiring significantly less memory, and without rehearsal buffers.

5 Conclusion

We introduced LANCE, a one-shot low-rank activation compression framework for efficient on-device learning. By calibrating reusable activation subspaces with a single HOSVD, LANCE eliminates repeated decompositions and reduces both memory and computational cost. Experiments show that LANCE achieves up to 380× memory savings while maintaining accuracy close to full backpropagation across diverse datasets and models. Crucially, the fixed low-rank subspaces naturally extend to continual learning, where LANCE matches the performance of gradient projection methods at a fraction of their memory footprint. Together, these results establish LANCE as a practical and scalable approach for enabling fine-tuning and continual learning on resource-constrained edge devices.

Acknowledgments

This work was supported in part by the Center for Co-design of Cognitive Systems (CoCoSys), one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program, and in part by the Department of Energy (DoE).

References

1. Ankit, A., Hajj, I.E., Chalamalasetti, S.R., Agarwal, S., Marinella, M., Foltin, M., Strachan, J.P., Milojicic, D., Hwu, W.M., Roy, K.: Panther: A programmable architecture for neural network training harnessing energy-efficient reram. IEEE Transactions on Computers **69**(8), 1128–1142 (2020). https://doi.org/10.1109/TC.2020.2998456

2. Apolinario, M.P.E., Choudhary, S., Roy, K.: CODE-CL: Conceptor-based gradient projection for deep continual learning. In: 2025 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 775–784 (October 2025). https://doi.org/10.1109/ICCV51701.2025.00080
3. Apolinario, M.P.E., Roy, A., Roy, K.: LLS: Local Learning Rule for Deep Neural Networks Inspired by Neural Activity Synchronization . In: 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). pp. 7807–7816 (Mar 2025). https://doi.org/10.1109/WACV61041.2025.00758
4. Apolinario, M.P.E., Roy, K., Frenkel, C.: TESS: A scalable temporally and spatially local learning rule for spiking neural networks. In: 2025 International Joint Conference on Neural Networks (IJCNN). pp. 1–9 (2025). https://doi.org/10.1109/IJCNN64981.2025.11227652
5. Barley, D., Fröning, H.: Compressing the backward pass of large-scale neural architectures by structured activation pruning. arXiv preprint arXiv:2311.16883 (2023)
6. Cai, H., Gan, C., Zhu, L., Han, S.: Tinytl: reduce memory, not parameters for efficient on-device learning. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20 (2020)
7. Chaudhry, A., Ranzato, M., Rohrbach, M., Elhoseiny, M.: Efficient Lifelong Learning with A-GEM. International Conference on Learning Representations (2019)
8. Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P.K., Torr, P.H., Ranzato, M.: On Tiny Episodic Memories in Continual Learning. arXiv:1902.10486 (2 2019), https://arxiv.org/abs/1902.10486v4
9. Chen, J., Zheng, L., Yao, Z., Wang, D., Stoica, I., Mahoney, M., Gonzalez, J.: Actnn: Reducing training memory footprint via 2-bit activation compressed training. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 1803–1813. PMLR (18–24 Jul 2021), https://proceedings.mlr.press/v139/chen21z.html
10. Chen, T., Xu, B., Zhang, C., Guestrin, C.: Training deep nets with sublinear memory cost. arXiv preprint arXiv:1604.06174 (2016)
11. Dellaferrera, G., Kreiman, G.: Error-driven input modulation: Solving the credit assignment problem without a backward pass. In: Proceedings of the 39th International Conference on Machine Learning. vol. 162, pp. 4937–4955. PMLR (17–23 Jul 2022)
12. Frenkel, C., Lefebvre, M., Bol, D.: Learning Without Feedback: Fixed Random Learning Signals Allow for Feedforward Training of Deep Neural Networks. Frontiers in Neuroscience **15**, 629892 (2 2021). https://doi.org/10.3389/FNINS.2021.629892/BIBTEX
13. Gomez, A.N., Ren, M., Urtasun, R., Grosse, R.B.: The reversible residual network: Backpropagation without storing activations. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 30. Curran Associates, Inc. (2017)
14. Hadsell, R., Rao, D., Rusu, A.A., Pascanu, R.: Embracing change: Continual learning in deep neural networks. Trends in Cognitive Sciences **24**(12), 1028–1040 (2020). https://doi.org/https://doi.org/10.1016/j.tics.2020.09.004
15. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778 (2016). https://doi.org/10.1109/CVPR.2016.90
16. Hinton, G.: The forward-forward algorithm: Some preliminary investigations. arXiv preprint arXiv:2212.13345 **2**(3), 5 (2022)

17. Jiang, Z., Chen, X., Huang, X., Du, X., Zhou, D., Wang, Z.: Back razor: Memory-efficient transfer learning by self-sparsified backpropogation. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022), https://openreview.net/forum?id=mTXQIpXPDbh
18. Ke, Z., Liu, B., Ma, N., Xu, H., Shu, L.: Achieving forgetting prevention and knowledge transfer in continual learning. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. NIPS '21, Curran Associates Inc., Red Hook, NY, USA (2021)
19. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences of the United States of America **114**(13), 3521–3526 (3 2017)
20. Krizhevsky, A.: Learning Multiple Layers of Features from Tiny Images (2009)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems **25** (2012)
22. Kudithipudi, D., Aguilar-Simon, M., Babb, J., Bazhenov, M., Blackiston, D., Bongard, J., Brna, A.P., Chakravarthi Raja, S., Cheney, N., Clune, J., Daram, A., Fusi, S., Helfer, P., Kay, L., Ketz, N., Kira, Z., Kolouri, S., Krichmar, J.L., Kriegman, S., Levin, M., Madireddy, S., Manicka, S., Marjaninejad, A., McNaughton, B., Miikkulainen, R., Navratilova, Z., Pandit, T., Parker, A., Pilly, P.K., Risi, S., Sejnowski, T.J., Soltoggio, A., Soures, N., Tolias, A.S., Urbina-Meléndez, D., Valero-Cuevas, F.J., van de Ven, G.M., Vogelstein, J.T., Wang, F., Weiss, R., Yanguas-Gil, A., Zou, X., Siegelmann, H.: Biological underpinnings for lifelong learning machines. Nature Machine Intelligence 2022 4:3 **4**(3), 196–210 (3 2022). https://doi.org/10.1038/s42256-022-00452-0
23. Liang, Y.S., Li, W.J.: Inflora: Interference-free low-rank adaptation for continual learning. In: 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 23638–23647 (2024). https://doi.org/10.1109/CVPR52733.2024.02231
24. Lillicrap, T.P., Cownden, D., Tweed, D.B., Akerman, C.J.: Random feedback weights support learning in deep neural networks. arXiv preprint arXiv:1411.0247 (2014)
25. Lin, J., Chen, W.M., Lin, Y., Cohn, J., Gan, C., Han, S.: MCUNet: Tiny Deep Learning on IoT Devices. In: 34th Conference on Neural Information Processing Systems (7 2020)
26. Lin, J., Zhu, L., Chen, W.M., Wang, W.C., Gan, C., Han, S.: On-device training under 256kb memory. In: Annual Conference on Neural Information Processing Systems (NeurIPS) (2022)
27. Lin, S., Yang, L., Fan, D., Zhang, J.: Beyond not-forgetting: continual learning with backward knowledge transfer. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA (2022)
28. Lin, S., Yang, L., Fan, D., Zhang, J.: TRGP: Trust Region Gradient Projection for Continual Learning. International Conference on Learning Representations (2022)
29. Nguyen, L.T., Quélennec, A., Nguyen, V.T., Tartaglione, E.: Beyond low-rank decomposition: A shortcut approach for efficient on-device learning. In: Singh, A., Fazel, M., Hsu, D., Lacoste-Julien, S., Berkenkamp, F., Maharaj, T., Wagstaff, K.,

Zhu, J. (eds.) Proceedings of the 42nd International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 267, pp. 46196–46210. PMLR (13–19 Jul 2025), https://proceedings.mlr.press/v267/nguyen25i.html
30. Nguyen, L.T., Quélennec, A., Tartaglione, E., Tardieu, S., Nguyen, V.T.: Activation map compression through tensor decomposition for deep learning. In: The Thirty-eighth Annual Conference on Neural Information Processing Systems (2024)
31. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian Conference on Computer Vision, Graphics and Image Processing. pp. 722–729 (2008). https://doi.org/10.1109/ICVGIP.2008.47
32. Nøkland, A.: Direct feedback alignment provides learning in deep neural networks. In: Advances in Neural Information Processing Systems (NeurIPS). vol. 29 (2016)
33. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Cats and dogs. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3498–3505 (2012). https://doi.org/10.1109/CVPR.2012.6248092
34. Ren, H., Anicic, D., Li, X., Runkler, T.: On-device online learning and semantic management of tinyml systems. ACM Trans. Embed. Comput. Syst. **23**(4) (Jun 2024). https://doi.org/10.1145/3665278
35. Saha, G., Garg, I., Roy, K.: Gradient Projection Memory for Continual Learning. International Conference on Learning Representations (2021)
36. Saha, G., Roy, K.: Continual Learning with Scaled Gradient Projection. Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023 **37**, 9677–9685 (6 2023). https://doi.org/10.1609/AAAI.V37I8.26157
37. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted Residuals and Linear Bottlenecks . In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4510–4520. IEEE Computer Society, Los Alamitos, CA, USA (Jun 2018). https://doi.org/10.1109/CVPR.2018.00474
38. Serra, J., Suris, D., Miron, M., Karatzoglou, A.: Overcoming catastrophic forgetting with hard attention to the task. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4548–4557. PMLR (10–15 Jul 2018), https://proceedings.mlr.press/v80/serra18a.html
39. Shang, J., Shao, S., Tong, T., Yang, F., Chen, Y., Jiao, Y., Liu, J., Gao, Y.: Divide and orthogonalize: Efficient continual learning with local model space projection. In: Chiappa, S., Magliacane, S. (eds.) Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence. Proceedings of Machine Learning Research, vol. 286, pp. 3766–3786. PMLR (21–25 Jul 2025), https://proceedings.mlr.press/v286/shang25a.html
40. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset (2011)
41. Wang, L., Zhang, X., Su, H., Zhu, J.: A Comprehensive Survey of Continual Learning: Theory, Method and Application. IEEE Transactions on Pattern Analysis and Machine Intelligence **46**(08), 5362–5383 (8 2024). https://doi.org/10.1109/TPAMI.2024.3367329
42. Zhao, Z., Zhang, Z., Tan, X., Liu, J., Qu, Y., Xie, Y., Ma, L.: Rethinking gradient projection continual learning: Stability/plasticity feature space decoupling. In: 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3718–3727 (2023). https://doi.org/10.1109/CVPR52729.2023.00362