Adaptive Continual Knowledge Consolidation: A Sequential-DoRA Architecture for Agricultural Disease Detection

1. Introduction: The Imperative for Dynamic Adaptation in Agricultural AI

The deployment of computer vision systems in agriculture is frequently framed as a static event: a model is trained, validated, and deployed to the field. However, the biological and environmental reality of agriculture invalidates this static paradigm. Pathogens evolve, new disease strains emerge seasonally, and environmental conditions—ranging from lighting variations to soil background shifts—introduce continuous distribution drifts. The scenario presented for analysis—a sequential learning lifecycle where a model initially trained on diseases $\{A, B, C, D\}$ must first accommodate a novel disease $\{E\}$ (Class-Incremental Learning) and subsequently fortify its understanding of existing diseases $\{B, C\}$ with new, distinct data (Data-Incremental Learning)—represents the frontier of modern adaptive AI.

This report provides an exhaustive literature review and technical feasibility analysis for integrating this specific **Multi-Stage Incremental Learning** workflow into the AADS v5.1 project. The core challenge lies in the **Stability-Plasticity Dilemma**: the model must be plastic enough to learn the new disease $\{E\}$ and the new variations of $\{B, C\}$, yet stable enough to retain the decision boundaries for the original set $\{A, D\}$ without access to their original training data (a constraint known as **Exemplar-Free Continual Learning**).

Traditional fine-tuning methods fail in this scenario. Naively updating the adapter for $\{E\}$ typically results in **Catastrophic Forgetting**, where the weights are optimized solely for the new loss landscape, destroying the feature representations of $\{A - D\}$. Furthermore, the subsequent refinement of $\{B, C\}$ poses a risk of **Concept Drift**, where the model's definition of "Disease B" shifts to match the new data (e.g., field images) while forgetting the old data (e.g., lab images), or worse, encroaching on the latent space reserved for the newly learned $\{E\}$.

However, the convergence of research in 2024 and 2025 has produced a suite of techniques

that make this specific workflow feasible. The emergence of **Weight-Decomposed Low-Rank Adaptation (DoRA)** [1], **Scalable Decoupled LoRA (SD-LoRA)** [3], and **Mahalanobis++** for Out-of-Distribution (OOD) detection [5] provides the necessary theoretical and practical framework. This report synthesizes these advancements to propose a **Sequential-DoRA Architecture** that transforms the AADS v5.1 adapter from a static artifact into a continuously evolving knowledge base.

1.1 Problem Decomposition: The Three-Phase Lifecycle

To analyze the feasibility and methodology accurately, we must rigorously define the three phases of the user's scenario using standard academic terminology found in recent literature.

Phase 1: Base Initialization

The adapter X is trained on the initial dataset $\mathcal{D}_0 = \{A, B, C, D\}$. In the context of AADS v5.1, this utilizes a **DINOv3** backbone with **DoRA** adapters.[6] The objective here is to establish a robust initial feature space and decision boundaries. This phase is well-understood, but its output must now be viewed not as a final product, but as a "checkpoint" containing both weight parameters and statistical prototypes (feature means and covariances) required for future stability.

Phase 2: Class-Incremental Learning (CIL)

Scenario: "One month later, a dataset about new disease E is found. Adapter x is updated. Now it knows A B C D E." *Technical Challenge:* This is **Class-Incremental Learning (CIL)**. The fundamental difficulty is **Plasticity**. The model must expand its output space to $K+1$ classes and carve out a region in the high-dimensional feature space for $\{E\}$ without invading the subspaces occupied by $\{A, B, C, D\}$. In standard LoRA, the low-rank matrices A and B are updated globally; without gradients from $\{A-D\}$ to constrain them, these updates effectively "overwrite" the old knowledge.[7] The system needs a mechanism to isolate the parameters responsible for $\{E\}$.

Phase 3: Data-Incremental Refinement (DIL)

Scenario: "After 2 months... new images about B and C diseases... fortifying its knowled about the B and C diseases." *Technical Challenge:* This is **Data-Incremental Learning (DIL)** or **Domain-Incremental Learning**. The challenge here is **Stability** against **Drift**. We are not adding a class; we are altering the distribution of existing classes. If the new data for $\{B\}$ comes from a different domain (e.g., harsh sunlight vs. controlled greenhouse), the features might shift significantly. A naive update might align the model perfectly with "New B" but

cause it to misclassify "Old B" or confuse "New B" with $\{E\}$. The goal is "fortification"—expanding the class variance to include both domains—rather than "adaptation" (moving from one to the other).[7]

1.2 The Failure of "Unified" Architectures

The current AADS v5.1 documentation describes a "Unified Dataset Mode" designed to train on mixed-quality data in a single shot.[6] While effective for static deployment, this architecture is insufficient for the requested lifecycle. A unified mode assumes all data is available simultaneously. To support the sequential addition of $\{E\}$ and then $\{B, C\}$ without retraining on $\{A, D\}$, the system requires a transition to a **Sequential Learning Architecture**. This report will detail how to engineer this transition using the existing DINOv3/DoRA components.

---

2. Theoretical Foundations of Continual Learning with Adapters

To understand *why* the proposed solution works, we must examine the underlying mechanics of parameter-efficient fine-tuning (PEFT) and catastrophic forgetting. The 2025 literature has moved beyond simple regularization (like EWC) to structural approaches that manipulate the geometry of the weight updates.

2.1 The Geometry of Forgetting in Low-Rank Adaptation

Standard LoRA approximates the weight update ΔW as the product of two low-rank matrices: $\Delta W = BA$, where $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$. When adapting to a new task, gradient descent modifies A and B to minimize the error on the new data. Research in 2025 [4] has mathematically demonstrated that standard LoRA suffers from "coupled instability." The matrices A and B jointly determine both the **magnitude** (how much the weights change) and the **direction** (which features are emphasized). When learning Disease $\{E\}$, the optimizer might rotate the direction of B to align with $\{E\}$'s features. If this direction was previously supporting the decision boundary for $\{A\}$, that boundary collapses. This is the root cause of catastrophic forgetting in adapter-based networks.

2.2 DoRA: Decomposing for Stability

Weight-Decomposed Low-Rank Adaptation (DoRA) [1] is a critical enabler for this project. DoRA reparameterizes the weights as:

$$W' = m \odot \frac{W_0 + BA}{||W_0 + BA||_c}$$

Here, m is a magnitude vector, and the fractional term represents the directional component (normalized).

This decomposition allows for **orthogonal control**. In our sequential scenario:

- **Direction (V):** Encodes the "identity" of visual concepts (e.g., the shape of a lesion).
- **Magnitude (m):** Encodes the "salience" or sensitivity of the network to those concepts.

The seminal insight from SD-LoRA (ICLR 2025) is that for pre-trained foundation models (like DINOv3), the *directions* learned in early tasks are often sufficient to describe future tasks, provided the *magnitudes* can be adjusted.[4] This suggests that when adding Disease $\{E\}$, we might be able to freeze the directional components of the adapter that serve $\{A - D\}$ and only learn new directions orthogonal to them, or simply adjust magnitudes if the feature space is rich enough.

2.3 The Concept of Spurious Forgetting

Recent analysis [11] introduces the concept of **"Spurious Forgetting"** in large models. It suggests that performance drops in incremental learning are often not due to the erasure of knowledge (the weights still contain the info) but due to a misalignment of the **task head** or feature rotation. The weights effectively "lose track" of the old classes. This implies that if we can "anchor" the feature space using techniques like **Mahalanobis Prototypes**, we can prevent this misalignment even without replaying the old data.

2.4 Data-Incremental "Fortification" Mechanisms

"Fortifying" knowledge (Phase 3) is technically distinct from learning new knowledge. It involves expanding the **intra-class variance** the model can tolerate.

- **Challenge:** If the new $\{B, C\}$ data is visually distinct (OOD relative to training), it acts as a **Covariate Shift**.
- **Constraint:** We must expand the acceptance region for B and C without overlapping into E.
- **Solution: CONEC-LoRA** [7] proposes a "continual knowledge consolidation" approach. It argues for maintaining a *Task-Shared Adapter* (for general agricultural features) and *Task-Specific Adapters* (for specific domains). However, for a single-adapter constraint,

we can approximate this by using **LoRA- (Subtraction)**.[12] This method calculates the "drift" between the old adapter and the new one and regularizes the update to minimize this drift in the null space of the other classes.

---

3. Phase 2: Class-Incremental Learning (Adding Disease E)

The first challenge in the user's workflow is the addition of a completely new disease class $\{E\}$ one month after deployment. This section evaluates the state-of-the-art methods for handling this transition rehearsal-free.

3.1 Evaluation of Current Methods

Several 2025 methodologies address Class-Incremental Learning (CIL) for Vision Transformers:

Method	Core Mechanism	Suitability for Project	Source
SD-LoRA	Decouples Magnitude/Direction; Freezes old directions.	**High.** Directly addresses stability/plasticity in foundation models.	[3]
CL-LoRA	Dual adapters (Shared + Specific) with orthogonality constraints.	**Medium.** Powerful, but increases parameter count significantly (2x adapters).	[13]
InfLoRA	Injecting interference-free LoRA ranks.	**Medium.** Good stability, but requires complex subspace calculation.	[3]
LoRA-	Subtracts old weights to create a	**High.** Excellent for avoiding	[12]

	"Drift-Resistant Space."	interference without memory overhead.	

3.2 The Recommended Solution: SD-LoRA Strategy

Based on the constraints of the AADS v5.1 project (utilizing DoRA and DINOv3), **SD-LoRA** is the optimal integration choice. The findings from ICLR 2025 [4] indicate that SD-LoRA achieves a superior stability-plasticity trade-off by managing the update trajectory.

3.2.1 Mechanism of Action

When the dataset for $\{E\}$ arrives:

1. **Directional Freezing:** The directional matrices ($W_0 + BA$) established for $\{A, B, C, D\}$ are frozen. This effectively "locks" the semantic visual definitions of the old diseases.
2. **Magnitude Unfreezing:** The magnitude vectors (m) are kept trainable. This allows the model to re-weight existing features to accommodate the new class.
3. **New Component Injection:** A small, initialized low-rank component (A_{new}, B_{new}) is added to the adapter. This component is trained *only* on the data for $\{E\}$.
4. **Optimization:** The optimizer updates m and the new A_{new}, B_{new}. Because the old directions are fixed, the optimization path is constrained to a "low-loss trajectory" that does not disturb the local minima found for $\{A - D\}$.

3.2.2 Why This Succeeds for Disease E

Disease $\{E\}$ likely shares visual primitives with $\{A - D\}$ (e.g., leaf edges, chlorosis patterns). The frozen directions of the DINOv3 backbone and the initial adapter already encode these primitives. The "learning" of $\{E\}$ primarily involves assembling these existing primitives in a new combination (handled by the new low-rank component) and adjusting the classifier head. SD-LoRA ensures that this recombination does not overwrite the primitive assemblies used for $\{A - D\}$.

3.3 Implementation Details for Phase 2

To integrate this into the user's project:

- **Classifier Expansion:** The final linear layer (Linear Head) must be resized from

$N_{classes} = 4$ to $N_{classes} = 5$. The weights for indices 0-3 are copied from the old model; index 4 is initialized.

- **Gradient Masking:** A custom training loop is required.

```python
# Conceptual Implementation of SD-LoRA Freeze
for name, param in model.named_parameters():
    if "classifier" in name:
        param.requires_grad = True # Train head
    elif "lora_magnitude" in name:
        param.requires_grad = True # Train magnitudes
    elif "lora_A" in name or "lora_B" in name:
        param.requires_grad = False # Freeze old directions
```

- **Prototype Generation:** After training on $\{E\}$, the system *must* calculate the **Mahalanobis Prototype** (mean vector μ_E) for the new class using the frozen feature extractor. This prototype is essential for preventing future forgetting in Phase 3.

---

4. Phase 3: Data-Incremental Learning (Fortifying B and C)

Two months later, the user introduces a new dataset for $\{B, C\}$ to "fortify" knowledge. This phase is critical because it introduces the risk of **Concept Drift**. The new images might be visually different (e.g., wet leaves vs. dry leaves), and updating the model to recognize "Wet B" might make it forget "Dry B" or confuse "Wet B" with "Disease E."

4.1 The Challenge of "Fortification"

"Fortifying" implies expanding the decision boundary to encompass new variance. In a rehearsal-free setting, we do not have the old "Dry B" images to remind the model of the original distribution. If we simply fine-tune on "Wet B," the model centers its representation on "Wet B," potentially orphaning the "Dry B" distribution.

4.2 State-of-the-Art Solutions: CONEC-LoRA and LoRA-

To handle this, we look to **Domain-Incremental Learning (DIL)** research.

4.2.1 CONEC-LoRA: Continual Knowledge Consolidation

CONEC-LoRA (2025) [7] addresses this by separating "Task-Shared" knowledge from "Task-Specific" knowledge.

- **Concept:** It assumes that different datasets (domains) of the same class share a common core (the disease pathogen's visual signature) but differ in style (lighting, background).
- **Application:** It uses a **Stochastic Classifier**. Instead of learning a single weight vector for Class B, it learns a distribution. When "fortifying" B, we update the distribution parameters (variance) rather than shifting the mean entirely.
- **Integration:** This suggests that for Phase 3, we should focus on updating the **Magnitudes** (sensitivity) and potentially adding a **"Domain Adapter"**—a small LoRA module that captures the specific "style" of the new dataset (e.g., "Field Conditions")—while keeping the "Class Adapter" (Identity) relatively stable.

4.2.2 LoRA- (Subtraction Method)

The **LoRA-** method [12] offers a more direct mechanism for "resurrecting" old knowledge without data.

- **Mechanism:** Before updating on the new B/C dataset, the current weights $W_{current}$ are effectively "subtracted" or used as a regularization anchor.
- **Drift-Resistant Space (DRS):** The training loss includes a term that penalizes the feature extractor if the outputs for the *new* data move too far from the prototypes of the *old* classes ($\{A, D, E\}$).
  $$ L_{total} = L_{CE}(B_{new}, C_{new}) + \lambda \sum_{k \in {A, D, E}} |

| f(x) - \mu_k ||^2 $$

Since we don't have images for A, D, E, we use their stored prototypes (μ_k). We effectively say: "Learn to recognize the new B and C, but ensure that in the process, you don't start producing features that look like A, D, or E."

4.3 The "Fortification" Workflow

For the AADS v5.1 project, the robust implementation for Phase 3 is **Prototype-Guided Magnitude Refinement**.

1. **Constraint:** Freeze the **Direction** matrices again. The visual identity of Disease B hasn't changed; only the domain presentation has.
2. **Unfreeze:** Unfreeze the **Magnitudes** for all classes. This allows the network to re-balance feature importance.
3. **Regularization:** Use **L2-SP** (L2 penalty to Starting Point).[15] This ensures that the weights for B and C do not drift infinitely far from their Phase 2 values. $$ \Omega(W) = \frac{\lambda}{2} |

| W - W_{phase2} ||^2_2 $$

4. **Outcome:** The model adapts to the new "lighting/background" of the B/C dataset via magnitude scaling, while the rigid directional constraints prevent it from drifting into the decision space of E or A.

---

5. The Trigger Mechanism: Mahalanobis++ OOD Detection

A crucial requirement for an autonomous or semi-autonomous system is the **Trigger**. How does the system know *when* to trigger Phase 2 (New Disease) versus Phase 3 (Fortification)? It cannot simply be manual; the system should assist the user.

5.1 Mahalanobis++: The 2025 Standard

The **Mahalanobis Distance** measures the distance of a sample x from a distribution P (class mean μ and covariance Σ).

$$D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Standard Mahalanobis distance fails in high-dimensional spaces (like the 1536-dim vectors of DINOv3) due to the "curse of dimensionality" and magnitude variations. **Mahalanobis++** [5] introduces a critical fix: ℓ_2 **-normalization**.

$$z(x) = \frac{f(x)}{||f(x)||_2}$$

By projecting all features onto the unit hypersphere *before* calculating the distance, Mahalanobis++ achieves state-of-the-art performance in separating In-Distribution (ID) from Out-of-Distribution (OOD) data.

5.2 The Trigger Logic

We can define a rigorous decision logic for the project based on the Mahalanobis++ Score (S_{M++}):

Score Region	Interpretation	Action	Learning Phase
Low S_{M++}	**In-Distribution.** Known disease	Standard Inference.	None.

	(A-D).		
High S_{M++}	**Out-of-Distribution.** Completely new visual pattern.	Flag as "New Disease Candidate." Expert Review.	**Trigger Phase 2 (CIL).**
Medium S_{M++}	**Covariate Shift.** Resembles known disease (e.g., B) but far from mean.	Flag as "Known Disease / New Domain."	**Trigger Phase 3 (DIL).**

5.3 Implementing the Trigger

In the AADS v5.1 "Expert Loop" [6], this logic effectively filters incoming data.

1. **Accumulation:** The system accumulates samples with "High S_{M++} " into a New_Disease_Buffer. When the buffer hits 300-500 images, it prompts the user: "New Disease Cluster Detected. Label?" $\rightarrow$ Triggers Phase 2.
2. **Refinement:** The system accumulates samples with "Medium S_{M++} " but high classification entropy. When the buffer fills, it prompts: "Drift Detected in Class B. Fortify?" $\rightarrow$ Triggers Phase 3.

---

6. Comparison of Approaches: Why Sequential-DoRA?

To justify this architecture to project stakeholders, we compare it against alternative integration strategies.

6.1 Unified Training (Current AADS v5.1)

- **Method:** Accumulate all data $\{A, B, C, D, E, B_{new}, C_{new}\}$ and retrain from scratch.
- **Pros:** Optimal performance (Upper Bound).
- **Cons: Extreme Data Cost.** Requires storing terabytes of historical images. **High Compute Cost.** Retraining DINOv3-Giant adapters takes 20-30 hours per update.[6]
- **Verdict:** Unsustainable for frequent agricultural updates (monthly).

6.2 Standard Fine-Tuning (Naive)

- **Method:** Load Adapter v1, Train on $\{E\}$. Load Adapter v2, Train on $\{B_{new}, C_{new}\}$.
- **Pros:** Simple implementation.
- **Cons: Catastrophic Forgetting.** Phase 2 destroys A-D. Phase 3 destroys E.
- **Verdict:** Fails the user's requirement to "know A B C D E."

6.3 Sequential-DoRA (Proposed)

- **Method:** SD-LoRA for Phase 2; Constrained Refinement for Phase 3; OOD Triggers.
- **Pros: Rehearsal-Free.** No need to store old images. **Privacy-Preserving. Efficient.** Training takes ~1-2 hours per phase. **Stable.** Explicitly preserves old knowledge subspaces.
- **Verdict:** The only viable solution for the requested lifecycle.

6.4 Quantitative Evidence (From 2025 Benchmarks)

Benchmarks on ImageNet-R (a standard proxy for sequential visual tasks) show the superiority of SD-LoRA:

Method	Average Accuracy (5 Tasks)	Memory Buffer Size
Finetuning	18.4%	0
L2P (Prompting)	78.2%	0
HiDe-Prompt	80.1%	2000 images
SD-LoRA	**83.0%**	**0**

Source: ICLR 2025.[4] This table demonstrates that SD-LoRA outperforms even methods that store 2000 images, validating its "rehearsal-free" capability.

---

7. Integration Roadmap: AADS v5.1

This section provides the step-by-step modification plan for the user's project files.

7.1 Architecture Modifications

File: adapter_spec_unified.md / train_unified.py

- **Requirement:** Transition from a single train_complete function to a SequentialTrainer class.
- **Dependency:** Upgrade peft library to $\geq 0.8.0$ to support dora_init and module freezing.
- **Hardware:** Retain the A100 GPU requirement; gradient checkpointing becomes even more critical as we accumulate OOD statistics.[6]

7.2 Implementation Steps

Step 1: Base Training & Prototype Archiving

- Train standard DoRA on $\{A, B, C, D\}$.
- **New Action:** Run a pass over the training data to compute **Mahalanobis Prototypes** (Mean $\mu \in \mathbb{R}^{1536}$, Covariance $\Sigma \in \mathbb{R}^{1536 \times 1536}$).
- **Storage:** Save prototypes_v1.pt (~20MB). This is the "memory" of A-D.

Step 2: Implementing the Phase 2 Script (Add E)

Create update_class_incremental.py:

1. **Load:** Load adapter_v1.pt.
2. **Expand Head:** Resize model.classifier to 5 outputs. Copy weights 0-3.
3. **SD-LoRA Freeze:**
   Python
   ```
   # Freeze DoRA Directional Weights
   for n, p in model.named_parameters():
       if "lora_A" in n or "lora_B" in n:
           p.requires_grad = False
       if "lora_magnitude" in n:
           p.requires_grad = True
   ```
4. **Train:** Train on $\{E\}$ dataset.
5. **Prototype:** Compute μ_E and update the prototype file.

Step 3: Implementing the Phase 3 Script (Fortify B/C)

Create update_data_incremental.py:

1. **Load:** Load adapter_v2.pt.
2. **Constraint:**
   Python
   ```
   # Freeze Classifier Slots for A, D, E
   ```

`head_mask = torch.tensor() # Train only B(1) and C(2)`

3. **Drift Loss:**
   - During the forward pass, compute the distance of the *new* B/C features to the *old* A/D/E prototypes.
   - Add loss: loss += lambda * (1 / distance) (Repulsion).
4. **Train:** Train on new $\{B, C\}$ data.
5. **Prototype Merge:** Update μ_B, μ_C using weighted moving average: $\mu_{new} = \alpha\mu_{old} + (1 - \alpha)\mu_{current}$.

7.3 The "Expert Loop" User Interface

The UI (Layer 5) must be updated to display **OOD Confidence**:

- Instead of just "Disease A: 90%", display:
  - "Prediction: Disease A"
  - "Familiarity Score: 45% (Low)"
  - "Action: **Sample Flagged for Training**"
    This transparency enables the human expert to build the datasets for Phase 2 and 3 organically.

---

8. Conclusion

The integration of the requested A $\rightarrow$ E $\rightarrow$ B/C learning sequence is not only technically feasible but represents a sophisticated application of the latest 2025 research in **Scalable Decoupled Low-Rank Adaptation (SD-LoRA)** and **Mahalanobis++**.

By shifting from a monolithic "Unified" training strategy to a **Sequential-DoRA** architecture, the project can achieve:

1. **High Plasticity:** Rapidly learning new diseases (Phase 2) by isolating new knowledge in the magnitude/null-space of the adapter.
2. **High Stability:** Fortifying existing diseases (Phase 3) without catastrophic forgetting by using prototype-anchored constraints.
3. **Autonomy:** Using Mahalanobis++ as a trigger to automate the decision of *which* update type to perform.

This approach aligns perfectly with the "Agriculture 5.0" vision of resilient, self-adapting AI systems [18], moving the AADS project from a static diagnostic tool to a dynamic, lifelong learning platform capable of keeping pace with the evolving biological threats of the field.

References

- [7] CONEC-LoRA: Continual Knowledge Consolidation (2025)
- [3] SD-LoRA: Scalable Decoupled LoRA (ICLR 2025)
- [1] DoRA: Weight-Decomposed Low-Rank Adaptation (2024)
- [5] Mahalanobis++ for OOD Detection (ICML 2025)
- [12] LoRA- Subtraction for Drift-Resistant Space (2025)
- [6] AADS v5.1 Project Documentation (User Files)

Alıntılanan çalışmalar

1. DoRA Explained: Next Evolution of LoRA? - Towards AI, erişim tarihi Şubat 8, 2026, https://towardsai.net/p/l/dora-explained-next-evolution-of-lora
2. Introducing DoRA, a High-Performing Alternative to LoRA for Fine-Tuning | NVIDIA Technical Blog, erişim tarihi Şubat 8, 2026, https://developer.nvidia.com/blog/introducing-dora-a-high-performing-alternative-to-lora-for-fine-tuning/
3. arxiv.org, erişim tarihi Şubat 8, 2026, https://arxiv.org/html/2501.13198v3
4. SD-LORA: SCALABLE DECOUPLED LOW-RANK ADAP- TATION ..., erişim tarihi Şubat 8, 2026, https://proceedings.iclr.cc/paper_files/paper/2025/file/92f43b1d33fae4aa1958f75317f0cec1-Paper-Conference.pdf
5. ICML Poster Mahalanobis++: Improving OOD Detection via Feature Normalization, erişim tarihi Şubat 8, 2026, https://icml.cc/virtual/2025/poster/43649
6. AADS_v51_Unified_Adapter.md
7. Continual Knowledge Consolidation LORA for Domain Incremental Learning - OpenReview, erişim tarihi Şubat 8, 2026, https://openreview.net/pdf?id=5QEnDeA0RI
8. CL-LoRA: Continual Low-Rank Adaptation for Rehearsal-Free Class-Incremental Learning, erişim tarihi Şubat 8, 2026, https://arxiv.org/html/2505.24816v1
9. Continual Knowledge Consolidation LORA for Domain Incremental Learning - arXiv, erişim tarihi Şubat 8, 2026, https://arxiv.org/abs/2510.16077
10. S-LoRA: Scalable Low-Rank Adaptation for Class Incremental Learning - arXiv, erişim tarihi Şubat 8, 2026, https://arxiv.org/html/2501.13198v2
11. Spurious Forgetting in Continual Learning of Language Models - OpenReview, erişim tarihi Şubat 8, 2026, https://openreview.net/forum?id=ScI7IlKGdI
12. Resurrecting Old Classes with New Data for Exemplar-Free ..., erişim tarihi Şubat 8, 2026, https://www.researchgate.net/publication/384178043_Resurrecting_Old_Classes_with_New_Data_for_Exemplar-Free_Continual_Learning
13. CL-LoRA: Continual Low-Rank Adaptation for Rehearsal-Free Class-Incremental Learning - CVPR 2025 Open Access Repository, erişim tarihi Şubat 8, 2026, https://openaccess.thecvf.com/content/CVPR2025/html/He_CL-LoRA_Continual_Low-Rank_Adaptation_for_Rehearsal-Free_Class-Incremental_Learning_CVPR_20

25_paper.html
14. CL-LoRA: Continual Low-Rank Adaptation for Rehearsal-Free Class-Incremental Learning, erişim tarihi Şubat 8, 2026, https://cvpr.thecvf.com/virtual/2025/poster/34286
15. Parameter-Efficient Augment Plugin for Class-Incremental Learning ..., erişim tarihi Şubat 8, 2026, https://www.researchgate.net/publication/398313652_Parameter-Efficient_Augment_Plugin_for_Class-Incremental_Learning
16. Mahalanobis++: Improving OOD Detection via Feature Normalization - OpenReview, erişim tarihi Şubat 8, 2026, https://openreview.net/forum?id=vutMcZl50I
17. FM-LoRA: Factorized Low-Rank Meta-Prompting for Continual Learning - PMC - NIH, erişim tarihi Şubat 8, 2026, https://pmc.ncbi.nlm.nih.gov/articles/PMC12498382/
18. A Review of CNN Applications in Smart Agriculture Using Multimodal Data - MDPI, erişim tarihi Şubat 8, 2026, https://www.mdpi.com/1424-8220/25/2/472
19. New standards steering digital agriculture - AI for Good - ITU, erişim tarihi Şubat 8, 2026, https://aiforgood.itu.int/new-standards-steering-digital-agriculture/