AADS-ULoRA v5.5–Independent Multi-Crop Continual Learning Architecture

Quick Start Adapter Guide

Continuous Learning for Multi-Crop Agricultural Disease Detection

With Independent Adapters, DoRA, SD-LoRA, CONEC-LoRA, and Dynamic OOD Detection

Agricultural AI Development Team

March 2026–Version

Abstract

Quick start guide for AADS-ULoRA v5.5 Independent Multi-Crop Continual Learning Architecture. This guide provides a streamlined 12-week timeline for implementing practical agricultural diagnostic capabilities: Simple Crop Router → Independent Crop Adapters → Per-Crop Continual Learning with Dynamic OOD Detection. Building on v5.4's independent adapters, v5.5 introduces dynamic per-class Mahalanobis thresholds for enhanced novelty detection. Each crop maintains its own lifecycle (Base → CIL → DIL) without requiring coordination with other crops, while automatically adapting OOD detection to each disease class's variability.

Contents

1 When to Use Independent Multi-Crop Architecture

1.1 Use v5.5 If:

✓ You need multiple crops with independent update schedules

✓ You want to add new diseases to one crop without affecting others

✓ You cannot store all historical training data (rehearsal-free)

✓ You want simple, practical deployment over theoretical complexity

✓ You need proven, literature-backed continual learning methods

✓ You want automatic OOD threshold adaptation (no manual tuning)

✓ You need per-class OOD sensitivity (different thresholds per disease)

1.2 Use v5.4 If:

× You prefer fixed, manually-tuned OOD thresholds

× You have very limited validation data for threshold computation

1.3 Use v5.3 If:

× You need cross-crop knowledge transfer (e.g., tomato → pepper)

× You have strong ML background and want cutting-edge research

× You can invest 6+ months in complex global orchestration

2 12-Week Implementation Timeline

Table 1: v5.5 Independent Multi-Crop 12-Week Timeline

Week	Phase	Activities
1-2	Setup	Environment, DINOv3 access, crop router training
3-4	First Crop (Tomato)	Phase 1 base training with DoRA, compute prototypes and dynamic OOD thresholds, validate $\geq$95% accuracy
5	Phase 2 (CIL)	Add new disease to tomato via SD-LoRA, validate retention $\geq$90%, update OOD thresholds
6	Phase 3 (DIL)	Fortify tomato with domain-shifted data via CONEC-LoRA, update OOD thresholds
7-8	Second Crop (Pepper)	Independent Phase 1 initialization, validate standalone
9-10	Third Crop (Corn)	Add third independent crop, verify no interference
11	Integration	Full pipeline with routing, dynamic OOD, all crops working independently
12	Demo & Docs	Deploy Gradio demo, document system architecture and OOD statistics

3 Adapter Specification

```python
adapter_spec_v55 = {
    # Identity
    'adapter_id': 'aads_multicrop_independent_v55',
    'architecture': 'independent_multicrop_dynamic_ood',

    # Crop Router Configuration (Simple)
    'crop_router': {
        'type': 'resnet50_classifier',  # Or DINOv3 linear probe
        'training_data': 'plantclef_crops',
        'target_accuracy': 0.98
    },

    # Per-Crop Adapter Configuration
    'per_crop': {
        'model_name': 'facebook/dinov2-giant',
        'use_dora': True,
        'lora_r': 32,
        'lora_alpha': 32,
        'loraplus_lr_ratio': 16,
        'phase1_epochs': 50,
        'phase2_epochs': 20,
        'phase3_epochs': 15
    },

    # OOD Detection (Dynamic Mahalanobis)
    'ood_detection': {
        'method': 'dynamic_mahalanobis',
        'threshold_factor': 2.0,  # k sigma (95% confidence)
        'min_val_samples_per_class': 10,  # Minimum for statistics
        'fallback_threshold': 25.0  # If insufficient validation data
    },

    # Performance Targets
    'targets': {
        'crop_routing_accuracy': 0.98,
        'phase1_accuracy': 0.95,
        'phase2_retention': 0.90,
        'phase3_retention': 0.85,
        'ood_auroc': 0.92,
        'ood_false_positive_rate': 0.05
    }
}
```

Listing 1: v5.5 Adapter Configuration

4 Phase-by-Phase Quick Start

4.1 Phase 0: Crop Router Setup (Weeks 1-2)

```bash
# Train simple crop classifier
python train_crop_router.py \
    --data_dir ./data/plantclef_crops/ \
    --crops tomato,pepper,corn \
    --output_dir ./models/crop_router/

# Validation: Crop classification accuracy >= 98%
python evaluate_router.py \
    --router ./models/crop_router/ \
```

```
    --test_data ./data/crops_test/
```

Listing 2: Train Simple Crop Classifier

4.2 Phase 1: First Crop Initialization (Weeks 3-4)

```
# Initialize tomato adapter with Phase 1 DoRA training
# IMPORTANT: Include validation data for OOD threshold computation
python train_phase1_crop.py \
    --crop_name tomato \
    --data_dir ./data/tomato/phase1/ \
    --val_dir ./data/tomato/val/ \
    --classes healthy,early_blight,late_blight,leaf_mold \
    --output_dir ./adapters/tomato/

# Validation: Clean accuracy >= 95%
python evaluate_crop.py \
    --adapter ./adapters/tomato/phase1/ \
    --test_data ./data/tomato/test

# Check computed OOD thresholds
python inspect_ood_stats.py \
    --adapter ./adapters/tomato/phase1/
# Output shows per-class thresholds:
#   healthy: mean=5.2, std=1.1, threshold=7.4
#   early_blight: mean=8.5, std=2.3, threshold=13.1
#   ...
```

Listing 3: Initialize Tomato Adapter with Dynamic OOD

4.3 Phase 2: Add New Disease (Week 5)

```
# Add new disease to tomato via SD-LoRA
python train_phase2_cil.py \
    --crop_name tomato \
    --phase1_checkpoint ./adapters/tomato/phase1/ \
    --new_class_data ./data/tomato/septoria_leaf_spot/ \
    --new_class_name septoria_leaf_spot \
    --output_dir ./adapters/tomato/phase2/

# Validation: Old class retention >= 90%
python evaluate_retention.py \
    --adapter ./adapters/tomato/phase2/ \
    --test_data ./data/tomato/test_all_classes

# Verify OOD thresholds updated for new class
python inspect_ood_stats.py \
    --adapter ./adapters/tomato/phase2/
# Should show new class with computed threshold
```

Listing 4: SD-LoRA Class Increment with OOD Update

4.4 Phase 3: Fortify Existing Classes (Week 6)

```
# Fortify tomato with domain-shifted data
python train_phase3_dil.py \
    --crop_name tomato \
    --phase2_checkpoint ./adapters/tomato/phase2/ \
    --fortification_data ./data/tomato/domain_shift/ \
    --target_classes early_blight,late_blight \
```

```
    --output_dir ./adapters/tomato/phase3/

# OOD thresholds automatically updated for fortified classes
python inspect_ood_stats.py \
    --adapter ./adapters/tomato/phase3/
```

Listing 5: CONEC-LoRA Fortification with OOD Update

4.5 Add Second Crop Independently (Weeks 7-8)

```
# Pepper adapter - completely independent from tomato
python train_phase1_crop.py \
    --crop_name pepper \
    --data_dir ./data/pepper/phase1/ \
    --val_dir ./data/pepper/val/ \
    --classes healthy,bacterial_spot,powdery_mildew \
    --output_dir ./adapters/pepper/

# Note: No LEBA transfer, no ELLA coordination
# Each crop is self-contained with its own OOD statistics
```

Listing 6: Independent Pepper Adapter

5 Production Deployment

```
from aads_v55 import IndependentMultiCropPipeline

# Initialize pipeline
pipeline = IndependentMultiCropPipeline(config)

# Register crops independently
# OOD statistics loaded automatically
pipeline.register_crop('tomato', './adapters/tomato/')
pipeline.register_crop('pepper', './adapters/pepper/')
pipeline.register_crop('corn', './adapters/corn/')

# Production inference
while True:
    image, metadata = receive_from_sensors()

    # Simple routing -> adapter prediction with dynamic OOD
    result = pipeline.process_image(image, metadata)

    if result['action'] == 'INFERENCE':
        # Standard diagnosis with confidence and OOD info
        display_result(result)
        # result includes:
        # - disease prediction
        # - confidence score
        # - mahalanobis_distance
        # - dynamic_threshold
        # - ood_score (distance / threshold)

    elif result['action'] == 'TRIGGER_PHASE2':
        # New disease detected in specific crop
        send_alert(f"New disease in {result['crop']}")
        send_alert(f"OOD score: {result['ood_score']:.2f}")
        accumulate_samples(result['crop'], image)

    elif result['action'] == 'PHASE2_COMPLETE':
        # Crop adapter updated with new OOD thresholds
```

```
        send_alert(f"Adapter updated: {result['crop']}")
        send_alert(f"New OOD stats computed for added class")
```

Listing 7: Production Pipeline with Dynamic OOD

6 Success Criteria

Table 2: v5.5 Success Criteria

Phase	Metric	Target
Crop Router	Crop classification accuracy	$\geq 98\%$
Per-Crop Phase 1	Clean accuracy	$\geq 95\%$
Per-Crop Phase 2	Old class retention	$\geq 90\%$
Per-Crop Phase 3	Protected class retention	$\geq 85\%$
OOD Detection	AUROC	≥ 0.92
OOD Detection	False positive rate	$\leq 5\%$
System-wide	Independence	Zero interference
Dynamic OOD	Per-class threshold validity	All classes have stats

7 Comparison: v5.2 vs v5.3 vs v5.4 vs v5.5

Table 3: Architectural Comparison Across Versions

Feature	v5.2	v5.3	v5.4	v5.5
Crops	Single	Multi (coupled)	Multi (independent)	Multi (independent)
Cross-crop transfer	N/A	LEBA + ELLA	None	None
OOD Detection	Fixed	Meta-OOD adaptive	Fixed Mahalanobis	**Dynamic Mahalanob**
Threshold type	Global	Adaptive global	Global fixed	**Per-class dynamic**
Manual tuning	Required	Less	Required	**Minimal**
Complexity	Medium	Very High	Medium	Medium
Implementation time	8 weeks	20+ weeks	12 weeks	12 weeks
Best for	Single crop	Research	Production	**Production + Robust C**

8 Troubleshooting

8.1 Crop Router: Low Accuracy

Symptom: Crop classification $< 95\%$
Solution:

- Use pre-trained PlantCLEF model or increase training data
- Ensure images contain clear leaf morphology
- Check for class imbalance in training data

8.2 Phase 2 SD-LoRA: High Forgetting

Symptom: Old class retention $< 85\%$
Solution:

- Verify lora_A and lora_B are frozen
- Reduce learning rate for Phase 2 (try 5e-5 instead of 1e-4)
- Increase number of Phase 2 epochs
- Check that old class data loader is properly configured

8.3 OOD Detection: Poor Calibration

Symptom: Too many false positives or missed detections
Solution:

- **High false positives:** Increase threshold_factor (try 2.5 for 99% confidence)
- **Missed detections:** Decrease threshold_factor (try 1.5 for 87% confidence)
- Ensure sufficient validation samples per class (≥ 10)
- Check that validation data represents true in-distribution variation

8.4 Dynamic OOD: Missing Class Statistics

Symptom: OOD stats show 0.0 for some classes
Solution:

- Ensure validation data has examples of all classes
- Check class indexing matches between training and validation
- Use fallback threshold (25.0) if insufficient data

References

[1] Liu, S., et al. (2024). DoRA: Weight-Decomposed Low-Rank Adaptation. *ICML 2024*.

[2] Wu, Y., et al. (2025). SD-LoRA: Scalable Decoupled Low-Rank Adaptation for Class Incremental Learning. *ICLR 2025*.

[3] Paeedeh, N., et al. (2025). Continual Knowledge Consolidation LoRA for Domain Incremental Learning. *arXiv:2510.16077*.

[4] Lee, K., et al. (2018). A Simple Unified Framework for Detecting Out-of-Distribution Samples. *NeurIPS 2018*.

[5] Gong, D., et al. (2024). Self-Expansion of Pre-trained Models with Mixture of Adapters for Continual Learning. *arXiv:2403.18886*.