

Baseline Expectations and Experimental Design

Author: Research Team

Date: January 2024

Status: Planning Phase

Table of Contents

- [1. Overview](#)
- [2. Baseline Model Goals](#)
- [3. Expected Performance](#)
- [4. Experimental Design](#)
- [5. Success Criteria](#)
- [6. Risk Assessment](#)

Overview

This document outlines the expected performance of baseline models and the experimental design for incremental learning experiments. It serves as a reference point for evaluating whether our implementation and experiments are on track.

Purpose

- Define clear success criteria before starting experiments
- Set realistic expectations based on literature
- Establish baseline performance targets
- Design rigorous experimental protocols

Baseline Model Goals

Phase 1: Initial 5-Class Model

Dataset Setup:

- Base Classes:** 5 plant disease classes (to be selected from PlantVillage)
- Training Set:** ~3,500 images (700 per class)
- Validation Set:** ~750 images (150 per class)
- Test Set:** ~750 images (150 per class)

Model Configuration:

- Architecture:** EfficientNet-B0 (primary), ResNet-18 (secondary)
- Input Size:** 224x224x3
- Pretrained:** Yes (ImageNet weights)
- Fine-tuning:** All layers

- **Optimizer:** Adam (lr=0.001)
- **Batch Size:** 32
- **Epochs:** 50 (with early stopping)

Expected Baseline Performance:

Based on literature review of similar datasets:

Metric	Expected Range	Target
Overall Accuracy	92-98%	≥95%
Per-Class Accuracy	90-98%	≥93%
Inference Time (CPU)	50-150ms	<100ms
Model Size	15-20MB	<25MB

Justification:

- PlantVillage is relatively clean and balanced
- 5 classes is manageable for initial training
- Literature shows 95%+ accuracy is achievable
- Pretrained models should transfer well

Phase 2: Initial 30-Class Model

Dataset Setup:

- **Base Classes:** 30 plant disease classes
- **Training Set:** ~21,000 images (700 per class)
- **Validation Set:** ~4,500 images (150 per class)
- **Test Set:** ~4,500 images (150 per class)

Expected Performance:

Metric	Expected Range	Target
Overall Accuracy	88-96%	≥92%
Per-Class Accuracy	85-95%	≥88%
Top-5 Accuracy	96-99%	≥97%
Inference Time (CPU)	50-150ms	<100ms

Challenges:

- More classes = more confusion potential
- Some diseases are visually similar
- Class imbalance may occur
- Longer training time required

Expected Performance

Incremental Learning Scenarios

We will test incremental learning by adding new classes to a pre-trained model. Expected forgetting rates based on literature:

Scenario 1: Add 1 New Class to 5-Class Model

Setup:

- Base model trained on 5 classes (95% accuracy)
- Add 1 new class with varying sample sizes: 10, 50, 100, 500 images
- Measure performance on both old and new classes

Expected Results:

Method	New Class Samples	Old Classes Acc.	New Class Acc.	Forgetting
Fine-tuning	10	70-80%	60-70%	15-25%
Fine-tuning	50	75-85%	75-85%	10-20%
Fine-tuning	100	80-88%	80-90%	7-15%
Fine-tuning	500	85-92%	90-95%	3-10%
LwF	10	85-90%	55-65%	5-10%
LwF	50	88-93%	70-80%	2-7%
LwF	100	90-94%	80-88%	1-5%
LwF	500	92-95%	88-93%	0-3%
EWC	10	83-88%	60-70%	7-12%
EWC	50	87-92%	72-82%	3-8%
EWC	100	89-93%	82-90%	2-6%
EWC	500	91-95%	88-94%	0-4%
Rehearsal (20/class)	10	88-93%	65-75%	2-7%
Rehearsal (20/class)	50	90-94%	78-86%	1-5%
Rehearsal (20/class)	100	92-95%	85-92%	0-3%
Rehearsal (20/class)	500	93-96%	90-95%	0-2%

Key Expectations:

1. **More samples = less forgetting:** Sufficient new data prevents overfitting
2. **Rehearsal performs best:** But requires storing old data
3. **LwF balances well:** Good trade-off between forgetting and new class learning

4. **Fine-tuning fails with few samples:** Severe forgetting expected

Scenario 2: Sequential Addition of Multiple Classes

Setup:

- Start with 5-class model
- Sequentially add 5 more classes (one at a time)
- 100 samples per new class

Expected Cumulative Forgetting:

Classes Added	Fine-tuning	LwF	EWC	Rehearsal
+1 (6 total)	12%	4%	6%	2%
+2 (7 total)	20%	8%	11%	4%
+3 (8 total)	28%	13%	16%	6%
+4 (9 total)	35%	18%	21%	9%
+5 (10 total)	42%	24%	26%	12%

Observations:

- Forgetting accumulates with each new class
- Rehearsal degrades as exemplar budget per class decreases
- LwF and EWC may struggle with many sequential tasks

Mobile Deployment Performance

After training, models will be optimized for mobile deployment:

Optimization Techniques:

1. **Quantization:** FP32 → INT8
2. **Pruning:** Remove 30% of weights
3. **Model export:** Convert to mobile format

Expected Mobile Performance (Android Mid-Range Device):

Model	Original Size	Optimized Size	Inference Time	Accuracy Drop
EfficientNet-B0	18MB	5MB	80-120ms	<1%
ResNet-18	44MB	11MB	100-150ms	<1%
MobileNetV2	14MB	4MB	50-80ms	<0.5%

Target Specifications:

- Model size: <10MB
 - Inference time: <150ms
 - Accuracy drop: <2%
 - Memory usage: <500MB
-

Experimental Design

Variables

Independent Variables:

1. Incremental Learning Method:

- Fine-tuning (baseline)
- Learning without Forgetting (LwF)
- Elastic Weight Consolidation (EWC)
- Rehearsal (with varying memory budgets)

2. Number of New Class Training Samples:

- 10, 50, 100, 500 images

3. Base Model Size:

- 5 classes
- 30 classes

4. Model Architecture:

- EfficientNet-B0 (primary)
- ResNet-18 (secondary comparison)

Dependent Variables:

1. Accuracy Metrics:

- Overall accuracy
- Per-class accuracy
- Old classes accuracy
- New class accuracy
- Top-5 accuracy

2. Forgetting Metrics:

- Absolute forgetting (accuracy drop)
- Relative forgetting (percentage drop)
- Per-class forgetting

3. Efficiency Metrics:

- Training time (seconds)

- Inference time (milliseconds)
- Model size (MB)
- Memory usage (MB)

Controlled Variables:

- Input image size: 224x224
 - Batch size: 32
 - Learning rate schedule
 - Random seed (for reproducibility)
 - Data augmentation parameters
-

Experimental Matrix

Experiment 1: Single Class Addition

- Base: 5-class model
- Add: 1 new class
- Samples: [10, 50, 100, 500]
- Methods: [Fine-tuning, LwF, EWC, Rehearsal]
- Seeds: [42, 123, 456] (3 runs each)
- **Total runs:** 4 methods × 4 sample sizes × 3 seeds = 48 experiments

Experiment 2: Multiple Class Addition

- Base: 5-class model
- Add: 5 classes sequentially
- Samples per class: 100
- Methods: [Fine-tuning, LwF, EWC, Rehearsal]
- Seeds: [42, 123, 456]
- **Total runs:** 4 methods × 5 incremental steps × 3 seeds = 60 experiments

Experiment 3: Large Base Model

- Base: 30-class model
- Add: 3 new classes (one at a time)
- Samples per class: [50, 100]
- Methods: [Fine-tuning, LwF, Rehearsal]
- Seeds: [42, 123, 456]
- **Total runs:** 3 methods × 3 classes × 2 sample sizes × 3 seeds = 54 experiments

Total Experiments: 162 runs

Estimated Time:

- Training time per run: ~30 minutes
 - Total training time: ~81 hours
 - Wall-clock time: ~4-5 days (with parallelization)
-

Data Collection

For each experiment, collect:

Performance Metrics:

- Confusion matrix (full)
- Per-class precision, recall, F1
- Overall accuracy, top-5 accuracy
- Inference time statistics (mean, std, min, max)

Incremental Learning Metrics:

- Old classes accuracy before/after
- New class accuracy
- Catastrophic forgetting measure
- Per-class accuracy changes

Model Information:

- Model size (parameters, file size)
- Training time
- Memory usage
- Hyperparameters used

Reproducibility Information:

- Random seed
 - Dataset split
 - Augmentation parameters
 - Environment (hardware, software versions)
-

Statistical Analysis

Comparison Methodology:

1. **Mean \pm Standard Deviation:** Report across 3 random seeds
2. **Statistical Significance:** Use paired t-test or Wilcoxon signed-rank test
3. **Effect Size:** Calculate Cohen's d for practical significance
4. **Confidence Intervals:** 95% CI for main results

Visualization:

- Line plots: Accuracy vs. number of samples
 - Bar plots: Method comparison
 - Heatmaps: Confusion matrices
 - Box plots: Forgetting distribution
-

Success Criteria

Minimum Viable Research (Must Have)

1. Baseline Model Performance:

- ☐ 5-class model achieves $\geq 93\%$ accuracy
- ☐ 30-class model achieves $\geq 90\%$ accuracy
- ☐ Inference time $< 150\text{ms}$ on CPU

2. Incremental Learning Implementation:

- ☐ All 4 methods implemented and working
- ☐ Reproducible results (consistent across seeds)
- ☐ Clear documentation of approach

3. Experimental Results:

- ☐ Complete Experiment 1 (single class addition)
- ☐ Statistically significant differences between methods
- ☐ Catastrophic forgetting clearly demonstrated

4. Mobile Deployment:

- ☐ Model runs on Android device
- ☐ Inference time $< 200\text{ms}$ on mobile
- ☐ Working prototype app

5. Documentation:

- ☐ Results documented with tables and figures
- ☐ Code is clean and commented
- ☐ README with reproduction instructions

Stretch Goals (Nice to Have)

1. Extended Experiments:

- ☐ Complete all 162 planned experiments
- ☐ Test with real field images (beyond PlantVillage)
- ☐ Compare additional architectures (MobileNetV3)

2. Advanced Analysis:

- ☐ Feature space visualization (t-SNE/UMAP)
- ☐ Similarity analysis between classes
- ☐ Ablation studies on hyperparameters

3. Enhanced Deployment:

- ☐ iOS application
- ☐ Model quantization and pruning
- ☐ On-device learning capability

4. Publication:

- ☐ Technical report/paper draft
- ☐ Open-source release with documentation
- ☐ Benchmark dataset contribution

Risk Assessment

Technical Risks

Risk	Probability	Impact	Mitigation
Baseline accuracy lower than expected	Medium	High	Try different architectures, more training
CUDA/GPU issues	Medium	Medium	Ensure CPU fallback works
Burn framework limitations	Low	High	Have PyTorch reference implementation
Mobile deployment failures	Medium	Medium	Start testing early, simplify if needed
Experiment time overrun	High	Medium	Prioritize key experiments, parallelize

Research Risks

Risk	Probability	Impact	Mitigation
No significant difference between methods	Low	High	Ensure experimental design is sound
Results not reproducible	Medium	High	Use fixed seeds, document everything
Insufficient data for conclusions	Low	Medium	Plan minimum sample sizes upfront
Literature comparison difficulties	Medium	Low	Use same dataset/splits as prior work

Project Management Risks

Risk	Probability	Impact	Mitigation
Scope creep	High	Medium	Stick to MVP, move extras to stretch goals
Time management	Medium	High	Weekly milestones, buffer time
Hardware failure	Low	High	Regular backups, cloud alternatives
Documentation lag	High	Medium	Document as you go, not at the end

Validation Strategy

Sanity Checks

Before trusting results, verify:

1. Overfitting Check:

- Train accuracy shouldn't be 100% while val is low
- Gap between train/val should be <10%

2. Forgetting Sanity:

- Fine-tuning should show clear forgetting
- Rehearsal with all data should have minimal forgetting
- Results should align with literature trends

3. Implementation Verification:

- Reproduce known results from papers (if possible)
- Compare Rust implementation with PyTorch reference
- Test on simple synthetic data first

Reproducibility Checklist

- ☐ Random seeds fixed and documented
- ☐ Dataset splits saved and versioned
- ☐ All hyperparameters logged
- ☐ Environment documented (OS, Rust version, dependencies)
- ☐ Code version tracked (git commit hash)
- ☐ Results can be regenerated from saved models

Timeline Estimates

Baseline Training (Weeks 1-2)

- 5-class model: 2-3 hours
- 30-class model: 8-12 hours
- Hyperparameter tuning: 1-2 days
- Validation and testing: 1 day

Incremental Learning Implementation (Weeks 3-5)

- Fine-tuning: Already done (baseline)
- LwF implementation: 2-3 days
- EWC implementation: 3-4 days
- Rehearsal implementation: 2-3 days
- Testing and debugging: 2-3 days

Experiments (Weeks 6-9)

- Experiment 1: 1.5 days
 - Experiment 2: 2 days
 - Experiment 3: 1.5 days
 - Buffer for reruns: 2 days
 - Analysis: 2-3 days
-

Next Steps

1. Immediate (Week 1):

- ☐ Finalize class selection for 5 and 30-class models
- ☐ Prepare dataset splits
- ☐ Set up experiment tracking system
- ☐ Create config files for baseline training

2. Short-term (Weeks 2-3):

- ☐ Train and validate baseline models
- ☐ Document baseline performance
- ☐ Begin incremental learning implementation

3. Medium-term (Weeks 4-8):

- ☐ Run all planned experiments
- ☐ Collect and organize results
- ☐ Perform statistical analysis

4. Long-term (Weeks 9-12):

- ☐ Mobile deployment
- ☐ Write technical documentation
- ☐ Prepare visualizations and paper draft

Last Updated: January 2024

Next Review: After baseline training completion