# Experiment 3: Optimizer & Learning Rate Tuning

Based off the initial key observations made from the initial first run, it is noted that from epoch 8-10 onwards, validation accuracy begins to fluctuate while training accuracy continues to rise. This is suggestive of the model possibly beginning to overfit to the training data.

Additionally, despite having balanced the data, the ‘mixed’ class shows a low precision (0.57) and a modest F1-score (0.67). This is indicative of the model struggling to learn generalisable patterns for visually diverse samples for the ‘mixed’ class.

As such, to address the observed overfitting and to further improve class-level performance, particularly for the ‘mixed’ class, it is essential to investigate how the optimizer type and learning rate configuration affect the model’s learning dynamics.

Therefore, this experiment aims to examine whether modifying the optimizer or tuning the learning rate can reduce signs of overfitting while enhancing the model’s ability to generalise across all classes.

A slower learning rate may help the model take smaller, more precise steps during weight updates, which could be beneficial for learning the subtle and varied features of complex classes like ‘mixed’. Similarly, alternative optimizers such as RMSprop or SGD may yield different learning dynamics that favour improved generalisation.

By experimenting with different combinations of optimizers and learning rates, the goal is to identify a configuration that not only stabilises validation performance across epochs but also improves the precision and F1-score of underperforming classes without compromising overall accuracy.

### Experiment Log – Optimizer & Learning Rate Tuning

**Dataset**: 800 images (4 classes, 200 each)  
**Model**: FruitCNN (3 conv layers, 1 FC layer)  
**Input size**: 100×100  
**Training device**: [MPS / CPU / GPU]

### Experiment Parameters

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Optimizer** | **Learning Rate** | **Batch Size** | **Scheduler Used** | **Epochs** |
| Adam | 0.001 | 32 | None | 15 |
| Adam | 0.0005 | 32 | None | 15 |
|  |  |  |  |  |

### Performance Metrics Comparison

|  |  |  |
| --- | --- | --- |
| **Metric** | **Initial Baseline Run (LR = 0.001)** | **Tuned Run #1 (LR = 0.0005)** |
| Train Accuracy | 0.94 | 0.94 |
| Validation Accuracy | 0.89 | 0.87 |
| Test Accuracy | 0.90 | 0.90 |
|  |  |  |
| Apple F1 Score | 0.92 | 0.92 |
| Banana F1 Score | 0.92 | 0.91 |
| Orange F1 Score | 0.94 | 0.95 |
|  |  |  |
| Mixed Precision | 0.57 | 0.60 |
| Mixed Recall | 0.80 | 0.60 |
| Mixed F1 Score | 0.67 | 0.60 |
|  |  |  |
| Macro Average F1 Score | 0.86 | 0.85 |
| Weighted Average F1 Score | 0.90 | 0.90 |

### Observations & Insights

##### Tuned Run #1:

* Both runs achieved identical train (94%) and test (90%) accuracy, indicating consistent model capacity across learning rates.
* The baseline run (LR = 0.001) slightly outperformed the tuned run (LR = 0.0005) in validation accuracy (89% vs 87%), suggesting better generalization within the same epoch budget.
* Overall, for Apple, Banana and Orange classes, there is a slight notable improvement in F1 scores
* Whereas for the Mixed class, the baseline had a higher F1 score (0.67 vs 0.60), with a notably higher recall (0.80 vs 0.60), suggesting better sensitivity to true positives.
* **Slightly lower macro F1 score (0.85 vs 0.86)** indicates that the model's performance **varied more across different classes**, particularly underperforming in the "Mixed" class.
* The model may have been **more cautious** (higher precision) but **less sensitive** (lower recall) to detecting certain classes.
* This suggests that the **tuned run prioritized precision over recall**, which may be suitable in contexts where false positives are more costly than false negatives.
* It could also imply that the **learning rate was too small** for the model to fully optimize its performance across all classes within the same training duration.

### Final Recommendations