**Phase 4: Optimizing EfficientNet & Further Refinements**

After **Phase 3** revealed the **limitations of EfficientNetB0**, this phase focuses on:  
✅ **Revising EfficientNet integration** to resolve training issues.  
✅ **Testing alternative pre-trained architectures** (EfficientNetV2S, EfficientNetB7).  
✅ **Introducing new dataset formats** for improved consistency.  
✅ **Enhancing model evaluation with additional performance metrics** (precision, recall, F1-score).

**4.1 EfficientNetV2S Implementation & Performance Evaluation**

Following **EfficientNetB0’s poor performance**, further research identified **EfficientNetV2S** as a strong alternative.

✅ **Why EfficientNetV2S?**

* **More efficient scaling** than EfficientNetB0.
* **Higher accuracy at reduced computational cost**.
* Designed for **better feature extraction on diverse datasets**.

**New Features in 4.1**

1. **Pathlib Integration** – Replaced OS-based directory handling for **cleaner file management**.
2. **Pre-Split Dataset Usage** – Ensured **consistent training/validation/test splits**.
3. **Automated Logging (CSVLoggerCallback)** – Simplified **result tracking**.

**Results & Analysis**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| epoch | train\_loss | train\_accuracy | val\_loss | val\_accuracy |
| 1 | 1.1358641386032104 | 0.6073740124702454 | 0.8104385733604431 | 0.7142857313156128 |
| 2 | 0.7446956634521484 | 0.7466169595718384 | 0.6739667654037476 | 0.7778649926185608 |
| 3 | 0.6397733688354492 | 0.7766228914260864 | 0.6440829038619995 | 0.779434859752655 |
| 4 | 0.5722954273223877 | 0.7983918190002441 | 0.5954170227050781 | 0.7849293351173401 |
| 5 | 0.5141079425811768 | 0.8254559636116028 | 0.6367316246032715 | 0.7786499261856079 |
| 6 | 0.4740147888660431 | 0.8336929082870483 | 0.6287651062011719 | 0.784144401550293 |
| 7 | 0.45026329159736633 | 0.8450676798820496 | 0.5533455014228821 | 0.8006279468536377 |
| 8 | 0.4089750051498413 | 0.8558540940284729 | 0.5666468739509583 | 0.802982747554779 |
| 9 | 0.38659313321113586 | 0.8636987805366516 | 0.5268163681030273 | 0.8218210339546204 |
| 10 | 0.3500184118747711 | 0.8758580088615417 | 0.5276644825935364 | 0.8163265585899353 |

|  |  |
| --- | --- |
| loss | 0.5741644501686096 |
| accuracy | 0.820588231086731 |

✅ **Best results so far on the mushroom dataset** (previous best: **76.3%**).  
✅ **Reduced overfitting compared to previous models**.  
✅ **Loss values significantly lower than earlier EfficientNet implementations**.

However, confusion matrix does not reflect the accuracy and loss data, indicating some issue, possibly with generalisation.

A blue squares with white text

AI-generated content may be incorrect.

📌 **Next Steps:** Compare **EfficientNetV2S vs. EfficientNetB7** for further improvements.

**4.2 EfficientNetB7 Integration & Performance Comparison**

The next test substituted **EfficientNetB7** for **EfficientNetV2S** while keeping hyperparameters identical.

✅ **Why EfficientNetB7?**

* **Largest & most powerful EfficientNet variant**.
* **Greater feature extraction depth**, especially for **complex datasets**.

**Results & Analysis**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| epoch | train\_loss | train\_accuracy | val\_loss | val\_accuracy |
| 1 | 1.1454973220825195 | 0.5922729969024658 | 0.8210660815238953 | 0.7260596752166748 |
| 2 | 0.744627058506012 | 0.7428907752037048 | 0.709729015827179 | 0.7653061151504517 |
| 3 | 0.6187347173690796 | 0.7811335325241089 | 0.7632706761360168 | 0.7386185526847839 |
| 4 | 0.5369474291801453 | 0.8134928345680237 | 0.5996302366256714 | 0.7998430132865906 |
| 5 | 0.4531058967113495 | 0.8407530784606934 | 0.646831750869751 | 0.7880690693855286 |
| 6 | 0.408512145280838 | 0.8633065223693848 | 0.6600754261016846 | 0.784144401550293 |
| 7 | 0.3473513722419739 | 0.8772308230400085 | 0.6196209788322449 | 0.795918345451355 |
| 8 | 0.31165143847465515 | 0.8897823095321655 | 0.5546718239784241 | 0.8202511668205261 |
| 9 | 0.28371602296829224 | 0.9029221534729004 | 0.5934820175170898 | 0.8116169571876526 |
| 10 | 0.24702267348766327 | 0.9123357534408569 | 0.5574294328689575 | 0.8233909010887146 |

|  |  |
| --- | --- |
| loss | 0.5930107831954956 |
| accuracy | 0.8176470398902893 |

* **Similar performance to EfficientNetV2S**, but slightly higher overfitting risk.
* **No significant gain over EfficientNetV2S**, despite its larger architecture.
* **Confusion matrix analysis revealed class imbalances still affecting predictions**.

A blue squares with white text

AI-generated content may be incorrect.

📌 **Next Steps:**

1. Address class imbalance via **dataset balancing** or **class weighting**.
2. Introduce **additional evaluation metrics** for deeper model insights.

**4.3 LeafSnap Dataset Expansion & Classification Report Integration**

To test EfficientNet’s scalability beyond mushrooms, the **LeafSnap dataset was reintroduced**, this time combining **lab and field images** for greater diversity.

**New Features in 4.3**

✅ **Pre-Split Dataset** – Ensured a **fixed test set** across all experiments.  
✅ **Classification Report** – Provided **Precision, Recall, F1-score** for better performance interpretation.  
✅ **Confusion Matrix Storage in CSV** – Enabled **tabular analysis** of misclassified images.

**Results & Analysis**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch | train\_loss | train\_accuracy | val\_loss | val\_accuracy |
| 1 | 0.6041288375854492 | 0.819915235042572 | 0.09760291874408722 | 0.9787685871124268 |
| 2 | 0.09363394975662231 | 0.9751059412956238 | 0.06681859493255615 | 0.9766454100608826 |
| 3 | 0.05764128640294075 | 0.9867584705352783 | 0.10196834802627563 | 0.9639065861701965 |
| 4 | 0.03619294613599777 | 0.9920550584793091 | 0.058931030333042145 | 0.9787685871124268 |
| 5 | 0.08439228683710098 | 0.9692796468734741 | 0.09217304736375809 | 0.966029703617096 |
| 6 | 0.043456628918647766 | 0.9862288236618042 | 0.030875694006681442 | 0.9893842935562134 |
| 7 | 0.017142845317721367 | 0.9973517060279846 | 0.04061240330338478 | 0.9830148816108704 |
| 8 | 0.016026057302951813 | 0.9968220591545105 | 0.07946506887674332 | 0.9681528806686401 |
| 9 | 0.018259607255458832 | 0.9941737055778503 | 0.03020964190363884 | 0.9851379990577698 |
| 10 | 0.010882783681154251 | 0.9978813529014587 | 0.04407785087823868 | 0.9830148816108704 |

|  |  |
| --- | --- |
| loss | accuracy |
| 0.007476646453142166 | 1.0 |

A screenshot of a crossword puzzle

AI-generated content may be incorrect.

✅ **Perfect test accuracy (100%)**, indicating **strong learning capability**.  
✅ **Exceptionally low loss (0.007), best results in the entire project**.  
✅ **However, conflicting results emerged in the classification report**.

**Classification Report Issue**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class | Precision | Recall | F1-Score | Support |
| acer\_negundo | 0.1 | 0.1 | 0.1 | 10.0 |
| acer\_palmatum | 0.0 | 0.0 | 0.0 | 6.0 |
| aesculus\_pavi | 0.0 | 0.0 | 0.0 | 7.0 |
| asimina\_triloba | 0.0 | 0.0 | 0.0 | 10.0 |
| cercis\_canadensis | 0.0 | 0.0 | 0.0 | 9.0 |
| chionanthus\_virginicus | 0.0 | 0.0 | 0.0 | 9.0 |
| gleditsia\_triacanthos | 0.0 | 0.0 | 0.0 | 8.0 |
| ilex\_opaca | 0.091 | 0.091 | 0.091 | 11.0 |
| liriodendron\_tulipifera | 0.1 | 0.1 | 0.1 | 10.0 |
| ostrya\_virginiana | 0.2 | 0.2 | 0.2 | 10.0 |
| prunus\_sargentii | 0.0 | 0.0 | 0.0 | 6.0 |
| ptelea\_trifoliata | 0.273 | 0.273 | 0.273 | 11.0 |
| quercus\_montana | 0.1 | 0.1 | 0.1 | 10.0 |
| styrax\_japonica | 0.0 | 0.0 | 0.0 | 6.0 |
| ulmus\_pumila | 0.0 | 0.0 | 0.0 | 6.0 |
| macro avg | 0.058 | 0.058 | 0.058 | 129.0 |
| weighted avg | 0.070 | 0.070 | 0.070 | 129.0 |

🚨 **Issue:**

* **Despite high test accuracy, precision/recall scores were extremely low**.
* **Confusion matrix confirmed poor class differentiation**, meaning the model **memorized training data** but failed to generalize.

📌 **Next Steps:** Investigate **whether the test set was mistakenly included in training** or if **dataset imbalance is distorting results**.

**Key Takeaways from Phase 4**

✅ **EfficientNetV2S outperformed other architectures**, achieving **82.1% accuracy** on real-world datasets.  
✅ **EfficientNetB7 did not provide significant improvements**, despite its larger architecture.  
✅ **Expanding LeafSnap to include real-world images produced outstanding accuracy**, but also revealed **issues with true generalization**.  
✅ **Classification report analysis showed discrepancies**, indicating a need for **dataset balancing & proper evaluation**.

**Next Steps → Future Work**

📌 **Investigate classification report anomalies** to ensure genuine generalization.  
📌 **Test additional techniques (class weighting, fine-tuning) to improve predictions on underrepresented classes**.  
📌 **Explore dataset augmentation strategies for LeafSnap** to improve real-world performance.