Assignment Report: Transfer Learning with Pre-trained CNNs

**Introduction**

Transfer learning has become a powerful technique in deep learning, particularly in computer vision tasks, where training large models from scratch can be computationally expensive and require vast amounts of labeled data. The core idea of transfer learning is to leverage knowledge learned by a model on a large dataset, such as ImageNet, and apply it to a different but related task with limited data.

In this project, we explore transfer learning using the ResNet-18 architecture pre-trained on ImageNet. By reusing the pre-trained model's learned features, we significantly reduce the training time and computational resources required to build an image classifier for a subset of the Food-101 dataset.

**Dataset**

For this project, we used the Food-101 dataset, a large-scale image dataset containing 101 food categories. To simplify the task and reduce computational requirements, we randomly selected 15 classes out of the 101 for focused experimentation. This approach allowed us to train and evaluate models efficiently while ensuring a manageable scope. To ensure reproducibility and consistency across all team members’ experiments, we set a random seed for Python, NumPy, and PyTorch. This guaranteed that the same 15-class subset was used across all runs.

Preprocessing Steps

*-> Image Resizing: All images were resized to 224×224 pixels to match the input size required by the ResNet-18 model.*

*-> Normalization: Images were normalized with a mean and standard deviation of [0.5, 0.5, 0.5], scaling pixel values to the range [-1, 1].*

*-> Data Loading: Filtered datasets were wrapped in PyTorch DataLoaders with a batch size of 32, enabling efficient batch processing and shuffling during training.*

Additionally, the dataset configuration and a sample of images were logged to Weights & Biases (W&B) for visualization, tracking, and reproducibility.

**Methodology**

Our approach to this assignment followed a structured transfer learning workflow designed to progressively improve model performance while practicing good experiment tracking and team collaboration. We utilized the ResNet-18 architecture pre-trained on ImageNet, adapting it for our 15-class Food-101 subset.

Overall Workflow: The assignment was completed in three main phases:

Phase 1

* Loaded the pre-trained ResNet-18 model.
* Replaced the final fully connected (FC) layer with a new layer outputting 15 classes.
* Froze all layers except the final FC layer.
* Ran two experiments each with different hyperparameter combinations to evaluate performance.

Phase 2

* Loaded the best checkpoint from Phase 1.
* Unfroze Layer4 and FC layer for fine-tuning while keeping earlier layers frozen.
* Conducted two fine-tuning experiments with different hyperparameters per run.

Phase 3

* Performed random sweeps using Weights & Biases (W&B) to automatically test multiple hyperparameter combinations.
* Explored optimizers, learning rates, weight decay, batch sizes, and activation functions.
* Ran 10 sweep runs each selecting the best configuration.

Weights & Biases Logging:

To maintain consistency and ensure reproducibility we standardized all logging functions. Every run logged the following to W&B:

* *Dataset details*
* *Model architecture and parameters*
* *Hyperparameters*
* *Epoch-wise metrics (Loss, Accuracy, Macro F1-Score)*
* *Confusion matrices*
* *Final Metrics (Best F1 & Accuracy for each run)*
* *Model artifacts (best-performing checkpoints saved and versioned)*

The model artifacts made it possible for any team member to pick up, fine-tune, or experiment further on the best models from others.

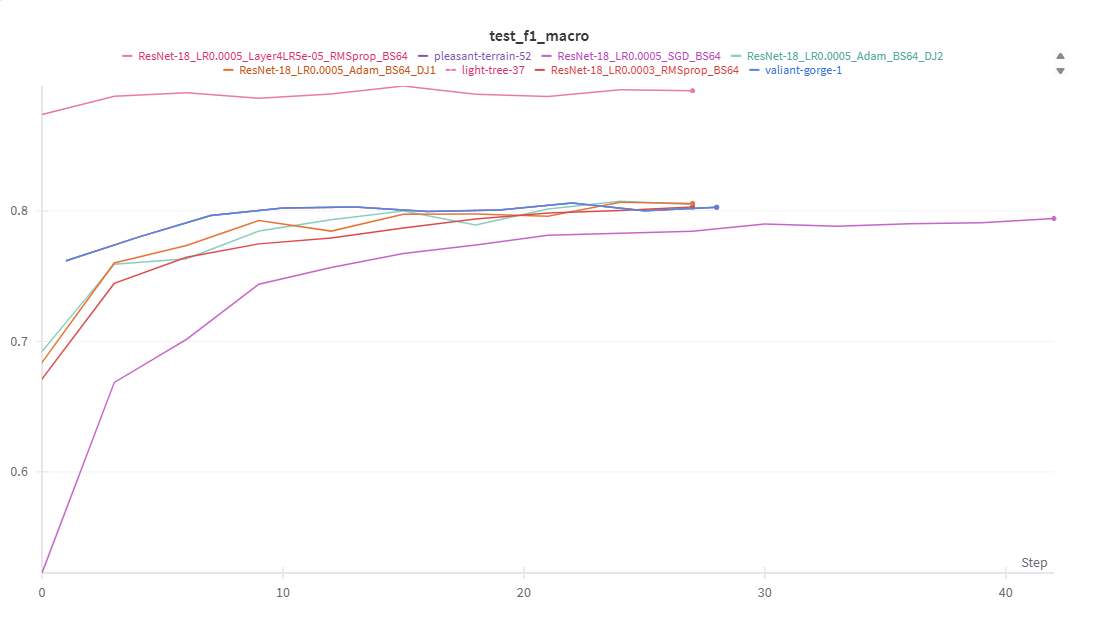
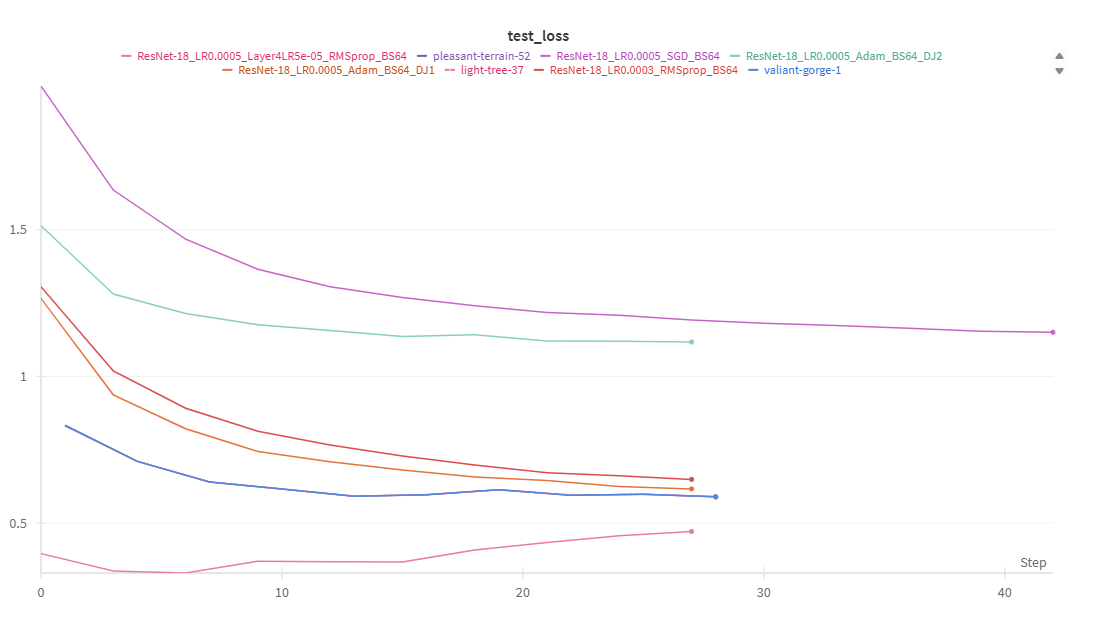
**Summary of Hyperparameter Categories Explored**

|  |  |
| --- | --- |
| **Hyperparameter** | **Experiment Values** |
| Optimizer | Adam, AdamW, RMSprop, SGD |
| Learning Rates | FC: 0.001, 0.0005, 0.0003  Layer4: 0.0001, 0.00005 |
| Batch Sizes | 32, 64 |
| Epochs | 5, 10, 15, 20 |
| Weight Decay | 0, 0.0001, 0.0005 (in sweep phase) |
| Activation Functions | ReLU, LeakyReLU (in sweep phase) |
| Separate LR for layers | Yes — Layer4 vs FC layers |

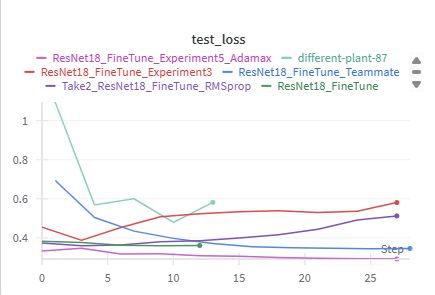
As part of our exploration, we experimented with several hyperparameters to understand how they influenced the model’s behavior and performance. Choosing the right optimizer was crucial — while Adam, AdamW, and RMSprop helped with faster and smoother convergence due to their adaptive nature, SGD gave us a good baseline but required careful tuning to avoid slow learning. We tested various learning rates, assigning lower values to the pre-trained layers (like Layer4) to preserve learned features and higher rates to the new FC layer for quicker adaptation. Batch size also played a role — smaller batches like 32 offered better generalization, while larger ones like 64 made the training more stable but increased memory usage. We experimented with different epochs, finding that more epochs helped convergence but sometimes risked overfitting. In the sweep phase, we also tested weight decay for regularization — useful for controlling overfitting but tricky because high values could hurt learning. Another interesting experiment was adding ReLU and LeakyReLU activations to see if they improved the model’s ability to learn complex patterns, especially avoiding dead neurons. Overall, these hyperparameter choices shaped how well the model learned, showing us the trade-offs between speed, generalization, and stability at every step.

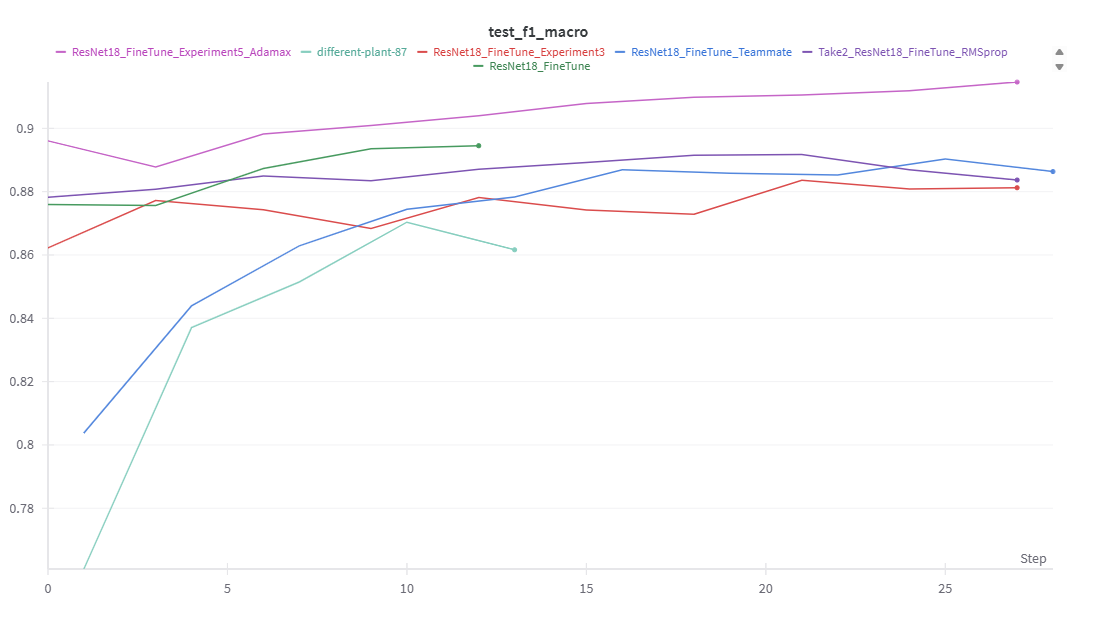
**Model Performance & Metrics**

Phase 1 Metrics

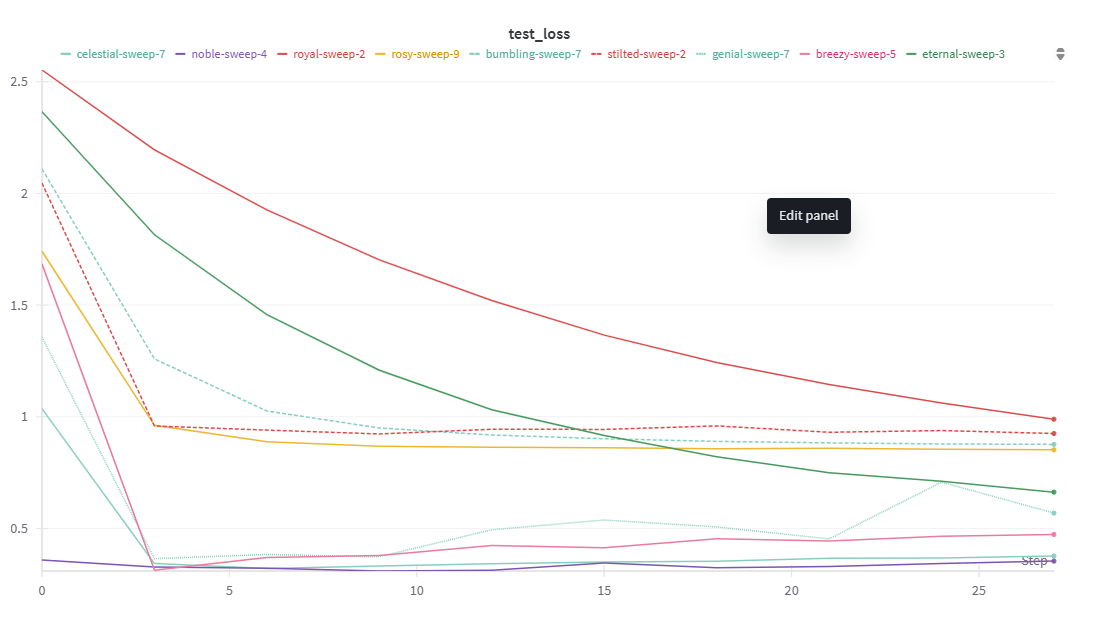


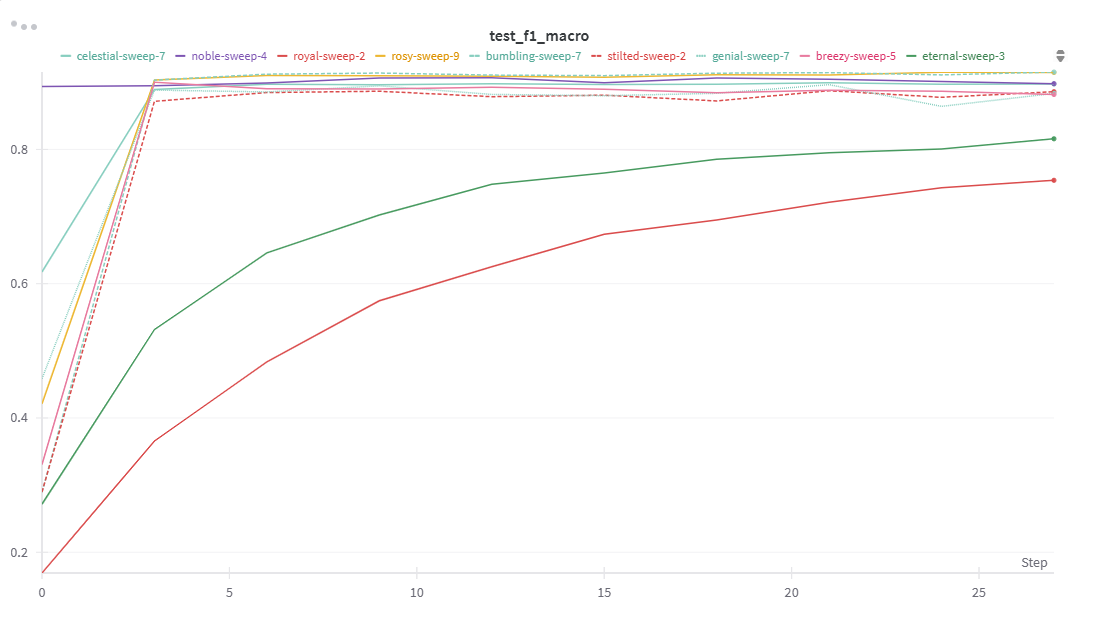
Phase 2 Metrics





Phase 3 Metrics





One of the most important takeaways from this assignment was understanding the impact of unlocking deeper layers during fine-tuning. In the first phase, when only the final fully connected layer was trained, the model relied heavily on pre-trained features. While this worked reasonably well, the performance was limited because the model couldn’t fully adapt to the unique characteristics of our Food-101 subset. Once we unfroze Layer4 along with the final layer, the model started learning more task-specific patterns, especially capturing subtle differences between similar food classes. This adjustment helped improve both accuracy and F1-score.

**Challenges**

Throughout the project, we faced several challenges common in deep learning workflows, especially when working with limited resources. One major challenge was the dataset complexity. Even though we worked with only 15 classes from Food-101, the dataset still contained a large number of high-resolution images with subtle visual differences between classes, making the classification task non-trivial. Another significant challenge was computational limitations. Since we worked on Google Colab with T4 GPUs, we had to carefully manage memory and runtime. Larger batch sizes and full fine-tuning runs quickly pushed the limits of available GPU memory. To handle this, we adjusted batch sizes, optimized data loading, and monitored usage closely to avoid crashes or forced runtime restarts. Manual hyperparameter tuning was also time-consuming and sometimes frustrating. Testing different combinations manually increased experimentation time, especially when certain settings did not perform as expected. The introduction of W&B sweeps helped address this by automating the exploration and reducing repetitive manual efforts.

**Conclusion & Future Improvements**

This project gave us hands-on experience in applying transfer learning using ResNet-18 for image classification on a subset of the Food-101 dataset. By following a structured approach—feature extraction, fine-tuning, and hyperparameter sweeps—we were able to understand how different training strategies and parameter choices impact model performance. Using Weights & Biases (W&B) throughout the project allowed us to track every experiment, compare runs easily, and manage model artifacts. One of the key takeaways was how powerful model artifact logging is—not just for reproducibility, but also for collaboration. Any team member could pick up a saved model, fine-tune it further, or test new hyperparameters without starting from scratch.

For future work, there are several exciting directions we could explore. One improvement would be experimenting with deeper architectures like ResNet-50 or ResNet-101 to see if the added complexity leads to better feature extraction and overall accuracy, especially given the dataset’s fine-grained classes. Since we have model artifacts saved, we could also use these as a starting point for those deeper models—loading the trained weights and continuing from there rather than beginning with generic ImageNet weights.

Overall, this project laid a strong foundation in transfer learning workflows and collaborative model development, which can be scaled up for more complex tasks and larger datasets in future projects.

Guardians Of The Galaxy.