Fine-Tune Torchvision Model Case Study Rubric

DS 4002 – Spring 2025 – William Giles
Due: TBD

Submission format:

- **Upload link to GitHub repo to Canvas**

Individual Assignment

Why am I doing this? This is your opportunity to apply your skills in data science, software engineering, and data engineering to solve a real-world issue. By first understanding the context and importance of the issue you are aiming to solve with plant classification, you will then apply your data science skills to enhance a machine learning pipeline. This will also require strong programming and software engineering design decisions, as you are building functionality into an object-oriented codebase. Finally, you will get to use your data engineering skills to integrate the fine-tuned model into Fast API, tying together the entire pipeline.

- Course Learning Objective: optimize and improve an end-to-end machine learning pipeline

What am I going to do? In this assignment, you will be given various tasks to ultimately improve the pipeline that takes an image of a plant, classifies its species, and describes its edibility. Specifically, you will first review the team's past model performances and gain an understanding of the key architectural differences in the models. You will then select a new Torchvision architecture to proceed with. The next component of this case study is adding new functionality to the classifier python class, which will allow you to initialize your newly chosen model and fine-tune it on the Pl@ntNet-300K dataset. You will first download the entire dataset, and then head to classifier.py to begin coding. You will notice three TODO sections in this class, which will require you to initialize your model, implement the fine-tuning pipeline, and preprocess the data for training. Next, you will update main.py by completing the TODO sections, which will instruct you to initialize the model and use the model to generate predictions. Finally, you will run main.py to generate evaluation metrics for your model and expose the Fast API endpoint, which allows for plant classification and edibility assessment. Lastly, answer the questions found in the reflection.txt file and include your evaluation metrics.

Your final deliverables will include:

- Updated classifier.py file with new model initialization, fine-tuning pipeline, and data preprocessing pipeline
- Updated main.py file with new model initialization and prediction pipeline
- Completed reflection.txt with model evaluation metrics
- Link to GitHub repo
- List of references

Tips for success:

- Do plenty of research before you code; oftentimes, we jump into a problem and begin coding right away, in this assignment, you will be updating and refactoring a codebase, which requires a thorough understanding of the project scope and implementation details
- Think about why certain models perform better than others for this task
    - Do some models have more layers than others? How do number of layers and parameter count influence model performance? What are some novel architectures that you could try?
- If done correctly, you should not need to write hundreds of lines of code. If you take the time to understand the model documentations and the fine-tuning in pytorch document, you will see that this can be done quite efficiently
- Hint: load the data in parquet form. Why is this more efficient storage?

How will I know I have Succeeded? You will meet expectations on this case study when you follow the criteria in the rubric below.

Spec Category	Spec Details
Formatting	● Repository – A GitHub repo containing all materials ○ Submit a link to the repo ○ Everything is contained in the repo or linked to it if appropriate. ○ Contents to add/modify ▪ Data/train_dataset_parquets/ ▪ Data/test_dataset_parquets/ ▪ Data/val_dataset_parquets/ ▪ Scripts/classifier.py ▪ Scripts/main.py ▪ Output/reflection.txt ○ For code, use proper syntax and styling rules, also adhere to the predefined class structure and DO NOT add or modify any other code than what is explicitly mentioned in this rubric and the README.md
Repository	• Goal: This is a GitHub repository containing all digital materials for the case study. • Do not restructure or move around any files or folders.
Data	● Goal: This folder will hold the datasets used for training, testing, and validation

	● While it is not a requirement for submission, this folder is where you will store the data (likely in parquet form) for the project ● Note: you likely will not push the parquet data up to the repo (as it is too large), but when doing local work, make sure to keep all data in this folder
Classsifer.py	● Goal: You will modify this file to add functionality for your new Torchvision model of choice ● Update __init__ function to initialize the new Torchvision model ● Update fine_tune function to fine_tune your new model ● Update preprocess_data to process training and validation for the fine-tuning pipeline
Main.py	● Goal: You will modify this file to successfully call classifier.py to fine-tune a model, get its evaluation metrics, and expose a Fast API endpoint ● Update the specified lines to initialize and fine-tune the classifier model ● Update the specified lines to classify the test images using your new model
Reflection.txt	● Goal: Reflect on your model choice and compare its accuracy against the other models previously used (found in output/model_comparison.png ● Answer the following questions: - Did you outperform the past models? - Which ones did your model perform better on worse than? - Why are you seeing these results? - Reflect on the differences in architecture among your custom model and the other models. ● State the evaluation metrics of your fine-tuned model (found in stdout after running main.py)

Acknowledgements: Special thanks to Jess Taggart from UVA CTE for coaching on making this rubric. This structure is pulled from Streifer & Palmer (2020).