Reproducibility and Reusability

Objective:

The primary objective of this assignment was to ensure reproducibility and reusability of machine learning models by building an automated pipeline. This was achieved through Docker for consistent environment setup and GitHub Actions to automate the workflow, enabling continuous integration (CI) and easy model retraining.

Steps and Approach:

1. Repository Setup:

We began by cloning our GitHub repository and navigating into it. This created a local copy of our project, which would serve as the foundation for building a Docker image and setting up an automated CI pipeline.

2. Docker Environment Configuration:

We created a `Dockerfile` to define a consistent environment for running our machine learning code. Using Docker ensures that our code runs in the same environment each time, regardless of external dependencies or changes to the host system.

The `Dockerfile` specified a lightweight Python 3.9 environment, installed essential libraries and dependencies, and copied our project files into the container.

Finally, we configured the Docker container to execute our `train.py` script, which runs the training process whenever the container is started.

3. Building and Running the Docker Image:

Using Docker, we built an image from the `Dockerfile`. This process converted our code and dependencies into a single, reproducible package.

Running the container allowed us to execute the training script (`train.py`) inside a consistent environment, confirming that our model training would work independently of the local machine's configuration.

4. Setting Up GitHub Actions for CI:

To automate the pipeline, we created a `.github/workflows` directory and added a workflow file named `ci.yml`.

This workflow was designed to trigger automatically whenever code was pushed to the main branch or a pull request was made.

Within the workflow, we configured multiple steps:

- **Checkout:** Retrieves the latest code from the repository.

- **Python Setup:** Installs Python 3.9.
- **Dependency Installation:** Installs required packages using `requirements.txt`.
- **Training Execution:** Runs `train.py` to train the model in the pipeline environment.

By setting up this workflow, we ensured that each push or pull request automatically triggers a new model training process, verifying the functionality and consistency of the code.

5. Committing and Pushing the Workflow to GitHub:

We committed the workflow file to the repository, which initiated the pipeline.

Observing the workflow's execution in the GitHub Actions tab provided insights into each step, allowing us to track success or identify errors in real time. The detailed logs enabled us to quickly troubleshoot any issues in the CI pipeline.

Results and Outcomes:

1. Reproducibility:

By using Docker, we created a reproducible environment that can be used by anyone to replicate the same results on different systems. This ensures that our model training process remains stable and consistent across environments.

2. Automation:

The GitHub Actions pipeline allowed us to automate the workflow, making it possible to trigger the training process automatically upon every change in the code. This process enables continuous integration (CI), allowing us to verify that all code changes work as expected before they're merged.

3. Increased Efficiency:

The automated CI pipeline saves time and reduces human error by executing the necessary steps (checking code, installing dependencies, running the training script) without manual intervention. This workflow also enables a faster development cycle, as each code update or bug fix can immediately be tested and verified.

4. Scalability and Version Control:

With this setup, our codebase is easily scalable. We can add additional steps for model evaluation, deployment, or experiment tracking in the future. Moreover, as the pipeline is connected to the repository, each version of the code can be tied to a corresponding model version, allowing for easy rollbacks or comparisons.

Conclusion:

This assignment successfully demonstrated how to leverage Docker and GitHub Actions to automate a machine learning pipeline. The setup guarantees consistent and reproducible model training, automates routine processes through CI, and provides a foundation for scaling into more advanced MLOps practices like continuous deployment, model versioning, and automated testing.