WOC_NITK _2023

Porting ProgressiveMultitask Model from Tensorflow to PyTorch

Task

Git Commands Explained:

1. **`git init`**
   - **Purpose:** Initializes a new Git repository in the current directory.
   - **Usage:** Run `git init` in the project's root directory to start version control.

2. **`git clone`**
   - **Purpose:** Creates a copy of an existing Git repository, including its history.
   - **Usage:** Use `git clone <repository_url>` to copy a remote repository to your local machine.

3. **`` `git add` ``**
   - **Purpose:** Stages changes for commit.
   - **Usage:** Use `git add <file>` or `git add .` to stage specific files or all changes.

4. **`git commit`**
   - **Purpose:** Records staged changes in a new commit.
   - **Usage:** Execute `git commit -m "Commit message"` to create a commit with a message.

5. **`git push`**
   - **Purpose:** Sends committed changes to a remote repository.
   - **Usage:** Use `git push <remote> <branch>` to push your local commits to the remote.

6. **`git pull`**
   - **Purpose**: Fetches changes from a remote repository and merges them.
   - **Usage:** `git pull` updates your local repository with changes from the remote.

7. **`git fetch`**
   - **Purpose:** Downloads changes from a remote repository without merging.
   - **Usage:** `git fetch <remote>` fetches changes from the specified remote without automatically merging them.

Merge Conflict in Git:

A merge conflict happens when Git can't automatically combine changes from different branches. To resolve:

1. Identify conflicted files using `git status`.
2. Open the conflicted file and manually resolve conflicts.
3. Remove conflict markers (<<<<<<<, =======, >>>>>>) and make the file correct.
4. Use `git add <file>` to mark the resolved file.
5. Create a merge commit with `git commit`.
6. Push the merged changes to the remote repository with `git push`.

Best Practices for Git Commit Messages:

- Keep messages concise (50-72 characters) in the subject line.
- Use the imperative mood (e.g., "Add feature" not "Added feature").
- Provide a detailed explanation in the body, including the "why" and "how."
- Reference issues or commits with #<issue_number> or Refs #<issue_number>.

Purpose of .gitignore Files:

`.gitignore` files specify files or directories that Git should ignore. It's used to exclude files like build artifacts and logs. For example, to ignore log files and the `node_modules` directory in a Node.js project, create a `.gitignore` with this content:

```
*.log
node_modules/
```

This keeps your repo clean by not tracking unnecessary files.