**WEEK-1 DL/ML TRACK**

**Artificial Intelligence (AI)**

**Definition:** AI is the capability of machines to perform tasks that typically require human intelligence. These tasks include understanding and processing natural language, recognizing images, making decisions, and solving problems.

**History:**

* **1950s-1960s:** The conceptual foundation of AI was laid during this period. Key figures include:
  + **Alan Turing:** Proposed the Turing Test to determine machine intelligence.
  + **John McCarthy:** Coined the term "Artificial Intelligence" in 1956.
  + **Marvin Minsky and Seymour Papert:** Early AI researchers exploring how machines could simulate human thinking.

**Techniques:**

* **Rule-based systems:** Use predefined rules to make decisions.
* **Decision trees:** Models that split data into branches to make decisions.
* **Optimization algorithms:** Techniques to find the best solution from a set of possible solutions.
* **Machine Learning (ML) and Deep Learning (DL):** Subfields of AI focusing on learning from data.
* **Natural Language Processing (NLP):** Enables machines to understand and respond to human language.
* **Computer vision:** Allows machines to interpret and make decisions based on visual input.

**Applications:**

* **Healthcare:** AI assists in diagnosis, drug discovery, and personalized treatment plans.
* **Finance:** AI helps in fraud detection, algorithmic trading, and credit scoring.
* **Transportation:** AI powers autonomous vehicles, traffic management systems, and logistics.
* **Education:** AI provides personalized learning experiences and automates grading.
* **Entertainment:** AI is used in game development, content recommendation, and special effects.

**Machine Learning (ML)**

**Definition:** ML is a subset of AI where machines learn from data without being explicitly programmed. The goal is to enable machines to improve their performance over time as they are exposed to more data.

**History:**

* **1980s:** The emergence of ML as a distinct field within AI, focusing on algorithms that can learn from and make predictions based on data.

**Techniques:**

* **Supervised learning:** Involves training a model on labeled data (e.g., regression and classification tasks).
* **Unsupervised learning:** Involves finding patterns in unlabeled data (e.g., clustering and dimensionality reduction).
* **Reinforcement learning:** Involves training models to make a sequence of decisions by rewarding desired behaviors.
* **Neural networks:** Computational models inspired by the human brain, used to recognize patterns in data.

**Applications:**

* **Image and speech recognition:** Identifying objects in images or transcribing spoken words.
* **Natural Language Processing (NLP):** Understanding and generating human language.
* **Recommendation systems:** Suggesting products or content based on user behavior.
* **Fraud detection:** Identifying fraudulent activities in financial transactions.
* **Predictive maintenance:** Predicting equipment failures to perform timely maintenance.

**Deep Learning (DL)**

**Definition:** DL is a subfield of ML that focuses on using neural networks with many layers (deep neural networks) to analyze complex patterns in large datasets.

**History:**

* **2000s:** DL gained prominence with the development of more powerful computational resources and algorithms, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

**Techniques:**

* **Convolutional Neural Networks (CNNs):** Specialized for processing grid-like data such as images.
* **Recurrent Neural Networks (RNNs):** Designed for sequential data such as time series or text.
* **Generative Adversarial Networks (GANs):** Consist of two networks (generator and discriminator) that compete to create realistic data.
* **Transfer learning:** Reusing a pre-trained model on a new, related task.

**Applications:**

* **Computer vision:** Tasks like image recognition, object detection, and facial recognition.
* **Natural Language Processing (NLP):** Tasks like machine translation, text summarization, and sentiment analysis.
* **Speech recognition:** Converting spoken language into text.
* **Autonomous vehicles:** Enabling self-driving cars to understand and navigate their environment.
* **Medical imaging:** Analyzing medical scans to detect diseases.

**Data Science**

**Definition:** Data Science is the interdisciplinary field that involves extracting insights and knowledge from data using techniques from statistics, computer science, and domain-specific knowledge.

**History:**

* **2000s:** Emerged as a distinct field due to the explosion of data availability and advances in computational power.

**Techniques:**

* **Data collection and cleaning:** Gathering and preprocessing data to ensure quality.
* **Data visualization and exploration:** Using charts and graphs to understand data patterns.
* **Statistical analysis and modeling:** Applying statistical methods to draw inferences and build models.
* **Machine Learning (ML) and Deep Learning (DL):** Using these techniques to make predictions and find patterns in data.
* **Data storytelling and communication:** Presenting data insights in an understandable and actionable way.

**Applications:**

* **Business intelligence (BI):** Analyzing business data to inform decision-making.
* **Healthcare:** Conducting epidemiological studies and improving public health outcomes.
* **Finance:** Performing risk analysis and optimizing investment portfolios.
* **Marketing:** Segmenting customers and personalizing marketing campaigns.
* **Environmental monitoring:** Analyzing climate data and supporting conservation efforts.

**Summary**

* **AI** is the overarching field that includes various techniques for creating intelligent systems.
* **ML** is a type of AI focused on learning from data.
* **DL** is a specialized subset of ML that uses deep neural networks.
* **Data Science** is the practice of extracting insights from data using a mix of AI, ML, and statistical techniques.

**Version Control with Git and GitHub**

**Version Control:**

Version control is a system that tracks changes to files over time, allowing multiple people to collaborate, manage different versions, and revert to previous states if necessary. It is essential for software development, documentation, and any work involving multiple contributors or iterative improvements.

**Git:**

**Definition:** Git is a distributed version control system that allows multiple people to work on a project simultaneously without interfering with each other's work. It tracks changes to files and coordinates work on those files among multiple people.

**Core Concepts:**

1. **Repository (Repo):** A repository is a directory or storage space where your project files and their revision history are stored. You can have a local repository on your computer and a remote repository on a server like GitHub.
2. **Commit:** A commit is a snapshot of your repository at a specific point in time. Each commit has a unique ID and includes a message describing the changes made.
3. **Branch:** A branch is a parallel version of your repository. By default, Git has a master branch (or main branch in newer versions). You can create other branches to work on different features or fixes independently.
4. **Merge:** Merging is the process of combining changes from different branches. This is typically done to integrate new features or fixes into the main branch.
5. **Clone:** Cloning is making a copy of an existing repository. The clone includes the entire history of the repository.
6. **Pull:** Pulling is fetching changes from a remote repository and merging them into your local repository.
7. **Push:** Pushing is sending your local commits to a remote repository.

**Basic Workflow:**

1. **Initialize a repository:**

git init

1. **Clone a repository:**

git clone <repository\_url>

1. **Check the status of your repository:**

git status

1. **Add changes to the staging area:**

git add <filename>

# or add all changes

git add .

1. **Commit changes:**

git commit -m "Commit message describing changes"

1. **Push changes to a remote repository:**

git push origin <branch\_name>

1. **Pull changes from a remote repository:**

git pull origin <branch\_name>

1. **Create a new branch:**

git checkout -b <new\_branch\_name>

1. **Merge a branch into the current branch:**

git merge <branch\_name>

**GitHub**

**Definition:** GitHub is a web-based platform that uses Git for version control and provides a collaborative environment for developers. It offers additional features such as issue tracking, project management, and social coding.

**Key Features:**

1. **Remote Repositories:** GitHub hosts remote repositories, making it easy to collaborate with others.
2. **Pull Requests:** A pull request (PR) is a method for submitting contributions to a project. It allows team members to review and discuss the changes before they are merged into the main branch.
3. **Issues:** Issues are used to track tasks, enhancements, and bugs. They provide a way to manage and prioritize work.
4. **Forks:** A fork is a copy of a repository that allows you to freely experiment with changes without affecting the original project.
5. **Actions:** GitHub Actions enable you to automate workflows such as testing, building, and deploying your code.

**Example Workflow on GitHub:**

1. **Fork a repository:**
   * Navigate to the repository you want to contribute to and click the "Fork" button. This creates a copy of the repository in your GitHub account.
2. **Clone your forked repository:**

git clone <your\_forked\_repository\_url>

1. **Create a new branch:**

cd <repository\_directory>

git checkout -b feature/new-feature

1. **Make changes and commit:**

git add .

git commit -m "Add new feature"

1. **Push changes to GitHub:**

git push origin feature/new-feature

1. **Create a Pull Request:**
   * Go to your forked repository on GitHub and click the "Compare & pull request" button.
   * Fill out the PR form with a descriptive title and comments about your changes, then submit the PR.
2. **Review and Merge:**
   * Repository maintainers will review your PR. Once approved, your changes can be merged into the main project.

**Example Commands:**

* **Creating a New Repository on GitHub:**
  + Go to GitHub and create a new repository.
  + Follow the instructions to push an existing repository or initialize a new one.
* **Connecting a Local Repository to a New GitHub Repository:**

git remote add origin <repository\_url>

git push -u origin master

* **Handling Merge Conflicts:**
  + When merging branches, you might encounter conflicts if changes overlap. Git will mark the conflict in the affected files.
  + Edit the files to resolve conflicts and then commit the changes:

git add <filename>

git commit -m "Resolve merge conflict"

**Git Branching Hands-On Learning**

Branching is one of Git’s most powerful features, allowing you to work on separate lines of development simultaneously. This hands-on guide will help you understand and practice branching with Git.

**Setting Up Your Environment**

1. **Install Git:** If you haven't already, install Git on your machine.
   * [Download Git](https://git-scm.com/downloads)
2. **Configure Git:** Set up your Git username and email.

git config --global user.name "Your Name"

git config --global user.email "you@example.com"

1. **Create a Local Repository:** Create a new directory for your project and initialize it as a Git repository.

mkdir my\_project

cd my\_project

git init

**Basic Branching Workflow**

1. **Create a New Branch:** To create a new branch and switch to it:

git checkout -b feature-branch

1. **Check Branches:** To see all branches and the current branch:

git branch

1. **Make Changes and Commit:** Make some changes to your files, then add and commit them.

echo "Some changes" > file.txt

git add file.txt

git commit -m "Add changes to file.txt in feature-branch"

1. **Switch Back to Main Branch:**

git checkout main

1. **Merge Feature Branch into Main Branch:** Merge the changes from the feature branch into the main branch.

git merge feature-branch

1. **Delete the Feature Branch:** Once merged, you can delete the feature branch.

git branch -d feature-branch

**Handling Conflicts**

1. **Create Two Branches:**

git checkout -b feature-branch-1

echo "Feature 1 changes" > file1.txt

git add file1.txt

git commit -m "Add file1.txt in feature-branch-1"

git checkout main

git checkout -b feature-branch-2

echo "Feature 2 changes" > file1.txt

git add file1.txt

git commit -m "Add file1.txt in feature-branch-2"

1. **Merge Feature Branch 2 into Main:**

git checkout main

git merge feature-branch-2

1. **Merge Feature Branch 1 into Main (Conflict):**

git merge feature-branch-1

You’ll see a conflict message. Open the conflicted file (file1.txt) and resolve the conflict. The conflict markers will look like this:

<<<<<<< HEAD

Feature 2 changes

=======

Feature 1 changes

>>>>>>> feature-branch-1

1. **Resolve Conflict and Commit:** After resolving the conflict, remove the conflict markers and commit the changes.

git add file1.txt

git commit -m "Resolve merge conflict between feature-branch-1 and feature-branch-2"

**Advanced Branching**

1. **Rebasing:** Rebase the feature branch onto the main branch. This reapplies your commits on top of another base tip.

git checkout feature-branch

git rebase main

1. **Cherry-Picking:** Apply a specific commit from one branch to another.

git checkout main

git cherry-pick <commit-hash>

1. **Stashing:** Save your changes temporarily and apply them later.

git stash

# Do some other work or switch branches

git stash apply

**Collaborative Branching with GitHub**

1. **Push Branch to GitHub:**

git push origin feature-branch

1. **Create a Pull Request (PR):** Go to GitHub and create a pull request from your feature branch to the main branch.
2. **Review and Merge PR:** Once the PR is reviewed, merge it into the main branch on GitHub.
3. **Pull Changes Locally:** Update your local main branch with the merged changes.

git checkout main

git pull origin main

**Practice Exercise**

1. **Create a New Repository:**

mkdir git-practice

cd git-practice

git init

1. **Create a README File and Commit:**

echo "# Git Practice" > README.md

git add README.md

git commit -m "Initial commit with README"

1. **Create and Switch to a New Branch:**

git checkout -b develop

1. **Make Changes and Commit:**

echo "Develop branch changes" > develop.txt

git add develop.txt

git commit -m "Add develop.txt in develop branch"

1. **Create Another Branch from Main:**

git checkout main

git checkout -b feature

1. **Make Changes and Commit:**

echo "Feature branch changes" > feature.txt

git add feature.txt

git commit -m "Add feature.txt in feature branch"

1. **Merge Develop into Main:**

git checkout main

git merge develop

1. **Merge Feature into Main:**

git merge feature