Lab : Introduction to Stable Diffusion Models

Level: M2 (Master's 2)

Duration: 4 hours

Objective

Objective

- Understand how neural networks can approximate the inverse of a function.
- Explore diffusion models and their applications in image generation.
- Implement a basic diffusion model to generate images.

Part 1: Preliminary Activity- Neural Network for Function Inversion

Concept

Neural networks can approximate complex functions, but can they learn to estimate the inverse of a function? We'll test this by training a model to approximate the inverse of y=sin(x), i.e., x=arcsin(y).

Steps

1. **Generate Dataset:**
    - Sample x values in the range [−1,1].
    - Compute y=sin(x).
    - The training goal is to predict x given y.
2. **Train a Neural Network:**
    - Input: y values.
    - Output: x values (i.e., trying to learn arcsin(y)).
    - Architecture: A simple MLP with:
        - 1 input neuron (for y),
        - 2 hidden layers (e.g., 3 neurons, ReLU activation),
        - 1 output neuron (for x).
3. **Evaluate the Model:**
    - Plot predicted x vs true arcsin(y).
    - Measure Mean Squared Error (MSE).
    - Discuss how well the network learns the inverse.

4. **Key Discussion Points:**
    - What happens for values outside the range [-1,1]?
    - What are the implications of approximating inverses in more complex functions?

Part 2: Diffusion Models on Images

Concept

Diffusion models learn to generate images by gradually denoising a noisy input. We'll implement a simple diffusion model to understand the process.

Steps

1. **Understanding Diffusion:**
    - The model starts with a pure noise image.
    - It gradually removes noise over multiple steps to generate a meaningful image.
    - Uses a U-Net as the denoising model.
2. **Dataset:**
    - Use MNIST or CIFAR-10 for simplicity.
3. **Training Steps:**
    - Add Gaussian noise to images at different levels.
    - Train a U-Net to predict the noise added at each step.
    - Use the trained model to iteratively denoise images.
4. **Implementation:**
    - Use PyTorch or TensorFlow.
    - Define the noise schedule.
    - Train a simple U-Net as the denoiser.
5. **Inference:**
    - Start with a random noise image.
    - Apply the trained model iteratively to generate a meaningful image.
6. **Discussion:**
    - What happens if we change the noise schedule?
    - How do diffusion models compare to GANs?

Final Reflection

- How does function inversion relate to diffusion models?
- How does iterative noise removal help generate realistic images?
- Potential applications of diffusion models (e.g., text-to-image generation like Stable Diffusion).