DML

Privacy in DML

- Privacy in distributed machine learning (DML) is a critical concern, especially when dealing with sensitive data across multiple parties. Some key challenges and solutions include:

Privacy in DML

Key Privacy Challenges in DML

1. **Data Leakage**: Models trained on distributed data can still reveal private information through gradients or model updates.
2. **Membership Inference Attacks**: Attackers can infer whether a particular data sample was used in training.
3. **Model Inversion Attacks**: Adversaries can reconstruct input data by analyzing model parameters.
4. **Poisoning Attacks**: Malicious participants can inject harmful data to manipulate model outputs.
5. **Communication Overhead & Security**: Exchanging model updates over a network introduces security risks and increased latency.

Privacy-Preserving Gradient Descent in DML

Gradient descent is a core optimization method in DML, but sharing gradients across nodes can leak sensitive information. Several techniques help preserve privacy while still enabling effective model training.

Challenges in Privacy-Preserving Gradient Descent

1. **Gradient Leakage**: Gradients can be exploited to reconstruct original training data.
2. **Membership Inference**: Attackers can infer whether a particular sample was part of the training set.
3. **Communication Overhead**: Privacy-preserving mechanisms often increase the computational and network costs.

Techniques for Privacy-Preserving Gradient Descent

Differential Privacy (DP)

- **Concept**: Adds noise to gradients before sharing them, preventing individual data points from being inferred.
- **Techniques**:
  - **Gaussian Noise Addition**: Perturbs gradients with normally distributed noise.
  - **Clipping Gradients**: Limits the maximum gradient value to reduce sensitivity.
- **Example**: **DP-SGD (Differentially Private Stochastic Gradient Descent)**, used in Google's TensorFlow Privacy.
- **Trade-offs**: Improves privacy but may degrade model accuracy.

Differentially Private Stochastic Gradient Descent (DP-SGD)

- **DP-SGD** is an extension of standard Stochastic Gradient Descent (SGD) that incorporates **differential privacy (DP)** to protect individual data points in a dataset. It is widely used in privacy-sensitive machine learning applications, such as healthcare and finance.

How DP-SGD Works

DP-SGD introduces **two main modifications** to standard SGD:

1.Gradient Clipping: Limits the sensitivity of individual samples' gradients.

2.Noise Addition: Introduces controlled noise to obscure individual contributions.

θ (theta) represents the **model parameters** (weights and biases) being optimized.

Steps of DP-SGD

Step 1: Mini-Batch Sampling

- Randomly sample a batch of training data at each iteration.

Step 2: Compute Per-Sample Gradients

- Compute the gradient $\nabla \mathcal{L}(x_i, \theta)$ for each individual data point x_i in the batch.

Step 3: Clip Gradients

- Limit the magnitude of each gradient to a predefined **clipping threshold** C
- **Clipping function**:

$$\bar{g}_i = \frac{\nabla \mathcal{L}(x_i, \theta)}{\max(1, \frac{||\nabla \mathcal{L}(x_i, \theta)||}{C})}$$

  - If the gradient norm is **greater than** C, it is scaled down.
  - Prevents individual data points with **large gradients** from having an outsized influence.

- **Case 1: When** $||g_i|| \leq C$
  - The fraction $\frac{||g_i||}{C}$ is **≤ 1**.
  - So, $\max(1, \frac{||g_i||}{C}) = 1$.
  - This means **no scaling is applied** → the gradient remains unchanged.
- **Case 2: When** $||g_i|| > C$
  - The fraction $\frac{||g_i||}{C}$ is **> 1**.
  - So, $\max(1, \frac{||g_i||}{C}) = \frac{||g_i||}{C}$.
  - The gradient is **scaled down** to have norm **exactly** C.

Step 4: Add Noise

- Add **Gaussian noise** to the sum of clipped gradients:

$$\tilde{g} = \frac{1}{B} \sum_{i=1}^{B} \bar{g}_i + \mathcal{N}(0, \sigma^2 C^2 I)$$

  - B = batch size.
  - σ = noise scale (determines privacy level).
  - I = identity matrix.

Step 5: Perform Model Update

- Update model parameters using the noisy gradient:

$$\theta \leftarrow \theta - \eta \tilde{g}$$

  - η = learning rate.

Step 6: Track Privacy Loss (Optional)

- Use the **privacy accountant** (e.g., the **Rényi differential privacy accountant**) to **track cumulative privacy loss** across training iterations.

Differentially Private Stochastic Gradient Descent (DP-SGD)

Feature	Advantage	Trade-off
Privacy Protection	Prevents individual data leakage	Reduces model accuracy
Gradient Clipping	Limits outlier influence	Can slow convergence
Noise Addition	Ensures DP guarantees	Requires careful tuning of σ
Scalability	Works with large datasets	Increases computational cost

Secure Multi-Party Computation (SMPC)

- **Concept**: Splits gradients into encrypted shares among multiple parties.

Techniques:

- **Additive Secret Sharing**: Each party holds a random share of the gradient, and only the sum of all shares reconstructs the original.
- **Oblivious Transfer**: Ensures secure computation without revealing data.
- **Example**: Crypten, PySyft's SMPC framework.
- **Trade-offs**: Strong security, but computationally expensive.

Homomorphic Encryption (HE)

- **Concept**: Encrypts gradients before transmission, allowing computations on encrypted values without decryption.
- **Techniques**:
  - **Fully Homomorphic Encryption (FHE)**: Allows any function to be computed on encrypted data.
  - **Partially Homomorphic Encryption (PHE)**: Supports only limited operations (e.g., addition or multiplication).
- **Example**: Microsoft SEAL, Google's TF Encrypted.
- **Trade-offs**: Computationally expensive and slower than other methods.

Federated Learning (FL) + Privacy Enhancements

- **Concept**: Model training happens locally, and only encrypted or noise-perturbed updates are sent to the central server.
- **Techniques**:
  - **Secure Aggregation**: Ensures the central server only receives an aggregated update, not individual gradients.
  - **Local Differential Privacy (LDP)**: Noise is added before updates leave the client's device.
- **Example**: Google's federated learning framework.
- **Trade-offs**: Reduces direct data exposure, but requires strong aggregation methods.

Blockchain for Secure Gradient Exchange

- **Concept**: Uses distributed ledger technology to ensure secure and auditable gradient updates.
- **Techniques**:
  - **Smart Contracts**: Automate secure gradient exchange.
  - **Zero-Knowledge Proofs (ZKP)**: Allow updates to be verified without revealing actual data.
- **Example**: Decentralized FL using Ethereum-based smart contracts.
- **Trade-offs**: Increases training time due to blockchain overhead.

Comparison

Technique	Security Level	Computational Overhead	Accuracy Impact
Differential Privacy (DP)	Medium	Low	Medium
SMPC	High	High	Low
Homomorphic Encryption (HE)	High	Very High	Low
Federated Learning (FL) + Secure Aggregation	Medium	Medium	Low
Blockchain-based Gradient Exchange	High	High	Low