What is Machine Learning Used For

- Machine Learning is used in many real-world applications:
  - **Translation services** (e.g., Google Translate)
  - **Robotics** (adaptive and capable robots)
  - **Autonomous vehicles** (detect and predict objects, pedestrians, vehicles)
  - **Healthcare** (data analysis, trend detection, treatment recommendation)

Failures in Machine Learning

- **Example 1: Google Translate attack**
  - Malicious users gave nonsense input.
  - The model produced Bible-like text in low-resource languages.
  - This showed the model *memorized training data*, causing privacy issues.
- **Example 2: Tesla Autopilot crashes**
  - Model failures led to fatal accidents.
  - Demonstrates that ML errors can have serious real-world consequences.
- **Example 3: Microsoft Tay Chatbot**
  - Released on Twitter in 2016.
  - Learned offensive behavior from users.
  - Had to be shut down within 16 hours.
- **Example 4: Skin Cancer Diagnosis Bias**
  - Models were less accurate for darker skin tones.
  - Shows *bias* and *fairness issues* in ML systems.

Key Risks to Trustworthy ML

A. Privacy Risks

1. **Reconstruction Attacks**
   - The model memorizes training data.
   - Attackers can reconstruct sensitive information (e.g., a person's face) by probing the model.
2. **Membership Inference Attacks**
   - Attackers try to find out if a specific data point was part of the training set.
   - Example: If a cancer model used someone's data, knowing they were part of the dataset reveals they had cancer.

B. Model Security Risks

1. **Model Stealing (Model Extraction)**
   - Training ML models is expensive (data, labeling, computation).
   - Attackers can query a public model through an API and train their own copy cheaply.
   - The stolen model can also be used for further attacks.
2. **Adversarial Attacks (Robustness Problem)**
   - Small, invisible changes to input (noise) can cause wrong predictions.
   - Example: A stop sign image slightly altered may be misclassified as a "right of way" sign.
   - Can cause real-world accidents in self-driving cars.

C. Lack of Transparency (Interpretability)

- Many ML models are *black boxes*, we don't know how they make decisions.
- This makes it hard to detect and fix bias or understand behavior.
- **Interpretable ML** tries to explain model decisions and understand which inputs affect predictions.

Other Topics to Be Covered in the Course

- **Collaborative Learning**: How multiple parties, train ML models together while preserving privacy.
- **Fairness:** Ensuring models perform equally well across different groups.
- **Security:** Protecting models and data from attacks.
- **Model Governance:** Integrating ML systems into society responsibly while respecting social values.

Summary

- ML brings powerful applications but also new risks.
- Trustworthiness includes:
  - Privacy protection
  - Robustness and security
  - Fairness and transparency
  - Responsible integration into society

1. Introduction

- The lecture covers **privacy in machine learning**.
- Focus areas:
  1. Where privacy leakage can happen in ML
  2. Who the adversaries are
  3. What the threat space looks like
  4. Common privacy attacks:
     - Attribute inversion
     - Model inversion
     - Membership inference
  5. The concept of **Differential Privacy** as a defense

---

2. Why Privacy Matters in ML

- Modern ML models often memorize data.
- When they generate new outputs or predictions, they can unintentionally reveal sensitive training data.

Example: Diffusion Models (Image Generators)

- These models learn to *denoise* images during training.
- They can sometimes regenerate almost identical copies of their training images.
- This can expose private data that came from the internet or social media.
- The better the model quality, the more likely it is to leak private data.
- Researchers showed this by generating many samples and checking which ones matched training data (membership inference).

Takeaway:

Better image generation quality often means *higher privacy risk*.

---

3. Why Models Leak Data

- During training, a model learns patterns from data and stores information about the training examples.
- We do not fully understand what information is stored or where it is stored inside the model.
- Because it “remembers” data to make good predictions, parts of this memory can be extracted.

---

4. Where Privacy Leakage Can Occur

Privacy can leak at several stages of the ML pipeline:

1. **During Training**

    - Attackers observing model updates can extract sensitive data.
    - Even gradient updates can reveal original samples.
2. **From Model Parameters**
    - Access to final model weights can expose information about the training data.
3. **From Model Predictions**
    - Attackers can detect differences between predictions on training vs unseen data.
    - These differences can leak membership or attribute information.

---

5. Who Are the Adversaries

To understand privacy threats, we must know what the attacker can access:

- **Knowledge**:
    - Model type, data distribution, hyperparameters, architecture, etc.
    - More knowledge allows stronger attacks.
- **Access Level**:
    - Can the attacker view or manipulate training?
    - Can they see or edit model parameters?
    - Can they only query the model through an interface?

---

6. Threat Scenarios

A. White-box

- The model is released publicly (e.g., on GitHub or Hugging Face).
- Adversaries can view model weights directly.
- Enables stronger and more direct attacks.

B. Black-box

- Model and data remain private.
- Attackers can only interact via inputs and outputs (e.g., an API).
- More realistic in practice (used in real-world systems like recommendations or ad targeting).

---

7. Privacy Attacks

A. Attribute Inversion

- Goal: infer a hidden attribute (like smoking status) of a specific person from a trained model.
- Example:
    - A healthcare model predicts cancer risk using both public and private data.
    - An attacker tries different values for the private field (“smoker: yes/no”) and observes the model’s confidence.
    - If the model is much more confident when “smoker=yes,” it reveals that the person likely *is* a smoker.
- Based on the fact that models are more confident on data they have seen during training.

B. Model Inversion

- Goal: reconstruct representative samples of each class from the model.
- Example:
    - In a face recognition model, each class represents a person.
    - An attacker reconstructs what each person’s face looks like by maximizing the model’s confidence for that class.
- Technique:
    - Start with random noise as input.
    - Adjust the input gradually using backpropagation to increase the model’s confidence for a target class.

- Result: an image that looks like a training example (a "prototype").
- **Risk:** serious privacy concern when classes represent individuals (like faces).

C. Membership Inference Attack

- Goal: determine if a specific data point was used to train the model.
- Example:
  - If a model was trained on cancer patients, knowing that someone's data was included reveals they had cancer.

Formal Setup:

- A "challenger" trains a model and picks a random data point (either from the training set or not).
- The "adversary" must guess whether the data point was in the training data.
- If the adversary performs better than random guessing (50%), it means the model leaks private information.

---

8. Membership Inference Using Shadow Models

Introduced by **Shokri et al. (2017)**

Idea:

- Train multiple "shadow models" that imitate the target model.
- Use them to learn how members and non-members differ in model confidence.

Steps:

1. The attacker collects or generates a *shadow dataset* similar to the target model's training data.
2. They train multiple shadow models on subsets of this data (some include certain samples, some do not).
3. They record model confidences for both members and non-members.
4. Using this information, they train a **binary classifier** that predicts whether a data point was a training member or not.
5. Then, the attacker queries the *real* target model and inputs the obtained confidence into the binary classifier to decide "member" or "non-member."

Drawback:

- Very computationally expensive (requires many models and data).

1. Data Collection and Training

- To perform membership inference, data must be collected and multiple models must be trained.
- These models help build a binary classifier that predicts whether a data point was part of the training set (a member) or not (a non-member).
- But this is a complex process, so researchers proposed simpler methods.

Threshold-Based Membership Inference Attack

- **Idea:** Models are usually more confident when predicting on data they have seen during training.
- Members → higher confidence
- Non-members → lower confidence
- We can set a **threshold** value for model confidence:
  - Above threshold → member
  - Below threshold → non-member

Choosing the Threshold

- Adjusting the threshold changes accuracy:
  - High threshold → fewer false positives but might miss members
  - Low threshold → catch all members but increase false positives
- Researchers use this tradeoff to study attack effectiveness.

ROC Curve (Receiver Operating Characteristic)

- Plots **True Positive Rate (TPR)** vs **False Positive Rate (FPR)** for different thresholds.
- Shows the balance between correctly identifying members and avoiding misclassifying non-members.
- **AUC (Area Under Curve)** measures average success, but privacy cannot be averaged.
  - Even leaking one person's data is a privacy breach.
- The key region of interest: **High TPR at low FPR** (catch real members without many false alarms).
- Existing attacks (like shadow model and threshold-based) fail to achieve this balance well.

Likelihood Ratio Attack (LRA)

- A newer, stronger method.
- Based on **hypothesis testing** between:
  - Models trained **with** a data point
  - Models trained **without** that data point

Idea:

- Compare how likely the model's output is under both distributions (with vs without data).
- Use the **Likelihood Ratio Test (Neyman-Pearson Lemma)** to decide if a data point was in training.

Problem:

- The true probability distribution of model parameters is intractable (too large).
- So, it's approximated using the **loss values** instead of model parameters.

Likelihood Ratio Attack

Require: model f, example (x, y), data distribution $\mathbb{D}$

$\text{confs}_{\text{in}} = \{\}$
$\text{confs}_{\text{out}} = \{\}$
for N times **do**
  $D_{\text{attack}} \leftarrow^{\$} \mathbb{D}$ ▷ *Sample a shadow dataset*
  $f_{\text{in}} \leftarrow \mathcal{T}(D_{\text{attack}} \cup \{(x, y)\})$ ▷ *train IN model*
  $\text{confs}_{\text{in}} \leftarrow \text{confs}_{\text{in}} \cup \{\phi(f_{\text{in}}(x)_y)\}$
  $f_{\text{out}} \leftarrow \mathcal{T}(D_{\text{attack}} \backslash \{(x, y)\})$ ▷ *train OUT model*
  $\text{confs}_{\text{out}} \leftarrow \text{confs}_{\text{out}} \cup \{\phi(f_{\text{out}}(x)_y)\}$
end for
$\mu_{\text{in}} \leftarrow \texttt{mean}(\text{confs}_{\text{in}})$
$\mu_{\text{out}} \leftarrow \texttt{mean}(\text{confs}_{\text{out}})$
$\sigma^2_{\text{in}} \leftarrow \texttt{var}(\text{confs}_{\text{in}})$
$\sigma^2_{\text{out}} \leftarrow \texttt{var}(\text{confs}_{\text{out}})$
$\text{conf}_{\text{obs}} = \phi(f(x)_y)$ ▷ *query target model*
return $\Lambda = \dfrac{p(\text{conf}_{\text{obs}} \mid \mathcal{N}(\mu_{\text{in}}, \sigma^2_{\text{in}}))}{p(\text{conf}_{\text{obs}} \mid \mathcal{N}(\mu_{\text{out}}, \sigma^2_{\text{out}}))}$

Membership Inference Attacks (MIA)

Goal: Find if someone's data was used in model training.

Threshold-based Attack

- Uses model confidence on correct label.
- Members = higher confidence.
- Choose threshold to separate members vs non-members.
- Shown with ROC curve.

Limitations:

- Hard to pick good threshold.
- Many false positives.

Likelihood Ratio Attack (LRA):

- Trains models with and without a data point.
- Compares loss distributions.
- Uses Gaussian assumption (mean, std dev).
- Uses logit scaling to make data look normal.
- Needs many shadow models (expensive).

Defenses:

1. Add noise to confidence values
2. Output only labels
3. Reduce overfitting (regularization, dropout, etc.)
4. Differential Privacy → gives mathematical privacy guarantee (next topic)

Differential Privacy Summary:

- Protects individual data by adding noise.
- Goal: Results look the same with or without one person's data.
- Trade-off: More noise = more privacy, less accuracy.

Key Terms:

- **ε (Epsilon):** Privacy strength (smaller = safer).
- **δ (Delta):** Chance of privacy break (smaller = safer).
- **Sensitivity:** How much one data point affects output.
- **Laplace Mechanism:** For one-time stats (pure ε-DP).
- **Gaussian Mechanism:** For repeated stats (ε, δ-DP).

Properties:

- Parallel composition: Separate data = no extra privacy loss.
- Sequential composition: Same data many times = add ε.
- Post-processing: Further steps can't reduce privacy.

Use Case Tip:

- Use **Laplace** for one result.
- Use **Gaussian** for machine learning (many updates).

Comparison:

Feature	Laplace Mechanism	Gaussian Mechanism
Privacy Type	Pure (ε)	Approximate (ε, δ)
Works Best For	One-time statistics	Repeated use (like model training)
Noise Shape	Sharp peak	Smooth bell curve
Utility	Better for single use	Better for many operations
Norm Used	L1	L2 (can also use L1)

https://github.com/AdeebaRafi/Backdoor-Attack-on-Neural-Network-using-MNIST-Dataset