25HS_RTAI_Note

lecture note

目录 / Contents

1 Part 1: Introduction 2
1.1 课程 Verticals 2
1.2 核心问题 2
1.3 Robustness Verification 2
1.3.1 问题定义 2
1.3.2 Certification 方法对比 2
1.4 Min-Max 优化框架 2
1.4.1 Standard Training 2
1.4.2 Robust Training 2
1.5 Robustness ≈ Individual Fairness 2
1.6 Differential Privacy 3
2 Part 2: Verification 3
2.1 Verification 问题形式化 3
2.2 Box Relaxation (IBP) 3
2.2.1 Abstract Transformers 3
2.2.2 Affine 层的传播 3
2.2.3 Crossing ReLU 3
2.3 MILP 编码 4
2.3.1 具体例子 4
2.4 DeepPoly Relaxation 4
2.4.1 Affine 层 4
2.4.2 ReLU 层 4
2.4.3 Back-Substitution 4
2.4.4 Single-Neuron vs Multi-Neuron 5
2.4.5 Triangle Relaxation 为什么不 Scale 5
2.5 α-β-CROWN 5
2.5.1 Lagrange Multiplier 方法 5
2.6 Floating-Point Soundness 5
2.7 Branch & Bound 算法 5
2.7.1 算法伪代码 5
2.7.2 Branching 启发式 5
2.7.3 复杂度分析 5
2.8 VNN-COMP 竞赛批判分析 5
2.8.1 需要检查的指标 6
2.8.2 Critical Thinking 6
2.8.3 VNN-COMP 常见问题 6
2.9 Part 2 易错点补充 6
2.10 三种攻击方法对比 6
2.11 Targeted vs Untargeted Attack 6
2.11.1 Targeted Attack 6
2.11.2 Untargeted Attack 6
2.12 FGSM 6
2.12.1 为什么用 Sign 函数? 6
2.13 C&W Attack 6
2.13.1 OPS 函数的契约 6
2.14 PGD 6
2.15 Adversarial Training 7
2.15.1 伪代码 7
2.15.2 Contrastive Learning 视角 7
2.16 TRADES：精度-鲁棒性权衡 7
2.16.1 PGD-AT vs TRADES 7
2.16.2 选择指南 7
2.17 AutoAttack：可靠的攻击评估 7
3 Part 4: Certified Training 7
3.1 PGD Training vs Certified Training 7
3.2 Certified Training Paradox 7
3.2.1 原因分析 7
3.3 SABR: Layer-wise Training 8
3.4 Logic → Loss Translation 8
4 Part 5: Randomized Smoothing & GCG Attack 8
4.1 Randomized Smoothing 8
4.1.1 核心思想 8
4.1.2 Certified Radius 8
4.1.3 两阶段采样 8
4.1.4 Deterministic vs Randomized Smoothing 9
4.1.5 为什么 RS 主要限于 ℓ_2? 9
4.1.6 DP 与 RS 的对偶性（Prof 强调!） 9
4.1.7 共同的 Lipschitz 基础 9
4.1.8 考试常考对比 9
4.1.9 常见失败模式 9
4.1.10 核心挑战 9
4.1.11 Three-Step Algorithm 9
4.1.12 White-box vs Black-box 10
4.1.13 Universal & Transferable Suffix 10
4.2 Mixed Adversarial Training 10
4.2.1 动机 10
4.3 Post-Training Attacks 10
4.3.1 Quantization Attack 10
4.3.2 Fine-Tuning Attack 10
5 Part 6: 考试要点 10
5.1 核心概念速查 10
5.1.1 Sound vs Complete 10
5.1.2 Crossing ReLU 10
5.1.3 Min-Max 结构 10
5.1.4 参数含义 10
5.2 方法对比表 10
5.3 易错点 10
6 Part 7: Privacy & Differential Privacy 10
6.1 Differential Privacy 核心思想 10
6.1.1 ε-DP 黄金公式 11
6.1.2 (ε, δ)-DP 放松 11
6.1.3 邻居关系的三种定义 11
6.2 两大基础机制 11
6.2.1 Laplace 机制 11
6.2.2 Gaussian 机制 11
6.3 DP 三大黄金性质 11
6.3.1 Post-Processing 11
6.3.2 Composition 11
6.3.3 Subsampling 11
6.4 DPSGD 算法 11
6.4.1 为什么需要 gradient 裁剪? 11
6.4.2 隐私预算累积 11
6.4.3 Privacy-Utility Trade-off 12
6.5 PATE：教师集成的 DP 12
6.5.1 噪声添加位置 12
6.5.2 隐私预算 12
6.6 Federated Learning + DP 12
6.6.1 FedSGD + DP 12
6.6.2 FedAvg + DP 区别 12
6.7 Model Stealing Attack 12
6.7.1 方法 12
6.7.2 防御 12
6.8 Model Inversion Attack 12
6.8.1 攻击公式 12
6.8.2 可视化 12
6.9 Membership Inference Attack (MIA) 12
6.9.1 Shadow Model 方法 13
6.9.2 Score-Based 方法（现代方法） 13
6.9.3 LiRA 详解（Likelihood Ratio Attack） 13
6.9.4 实现步骤 13
6.9.5 为什么比 Loss 更好? 13
6.9.6 MIA 实际表现 13
6.10 Dataset Inference 13
6.11 Memorization 13
6.12 DP 与 RS 的对偶性 14
6.13 Gradient Inversion Attack 14
6.13.1 Image Prior 14
6.13.2 Text Prior 14
6.13.3 Tabular Prior 14
6.13.4 FedSGD vs FedAvg 攻击难度 14
6.14 Attribute Inference 14
6.15 隐私攻击层次关系 14
6.16 Agentic AI 安全 15
6.16.1 Instruction Hierarchy 15
6.16.2 Command Sense 15
6.16.3 Dual-LLM Pattern 15
6.17 敏感度的统一视角 15
6.18 MIA Score 函数总览 15
6.19 考试模式识别 15
6.19.1 场景 → 威胁类型映射 15
6.19.2 对比/辨析题要点 15
6.20 Privacy 易错点 15
7 Part 8: Watermarking & Benchmarking 15
7.1 LLM Watermarking 核心思想 15
7.2 Red-Green Watermark (KGW) 16
7.2.1 Generate 函数 16
7.2.2 Detect 函数 16
7.3 ITS Watermark（Distortion-Free） 16
7.3.1 Red-Green 的问题 16
7.3.2 Distortion-Free 核心 16
7.4 SynthID (Google DeepMind) 16
7.4.1 特点 16
7.4.2 Tournament Sampling 17
7.5 水印攻击方法 17
7.5.1 Scrubbing（移除） 17
7.5.2 Spoofing（伪造） 17
7.5.3 Stealing（窃取） 17
7.6 Radioactivity（数据保护） 17
7.7 LLM Benchmarking 17
7.7.1 四大评估范式 17
7.8 Contamination（污染问题） 17
7.8.1 形式化定义 17
7.8.2 检测方法（按 Access Level 分类） 17
7.8.3 Level 1: Oracle Access 17
7.8.4 Level 2: White-box 17
7.8.5 Level 3: Black-box 17
7.8.6 Outcome-based Detection 17
7.9 Dynamic Benchmarks（动态评估） 17
7.9.1 Dynamic Benchmark 类型 18
7.9.2 核心优势 18
7.9.3 Polyrating：De-biasing 方法 18
7.10 Scoring Mechanisms 18
7.10.1 Goodhart's Law 18
7.10.2 Bradley-Terry Model 18
7.10.3 Judge Bias Problem 18
7.11 Reporting Best Practices 18
7.11.1 应该做的 √ 18
7.11.2 常见不诚实手法 ✗ 18
7.12 Watermarking 评估指标 18
7.13 易错点 18

1 Part 1: Introduction

1.1 课程 Verticals

课程围绕三个研究方向展开：

Attacks & Guarantees	Privacy	Provenance
Convex Relaxation	Membership Inference	Watermarking
Certified Training	Differential Privacy	Benchmark Eval
Randomized Smoothing	Federated Learning	Contamination

Table 1: RTAI 三大方向

1.2 核心问题

所有方法都围绕一个核心问题展开：

$$\forall \delta \in \mathcal{B}(x): \quad f(x+\delta) = f(x)?$$

其中 $\mathcal{B}(x) = \left\{ x' \;\middle|\; \|x' - x\|_p \le \varepsilon \right\}$ 是扰动集合。

⚄ 关键张力在于 **存在量词** $\exists$ vs **全称量词** $\forall$：
- Attack：证明 $\exists \delta$（找反例）
- Verification：证明 $\forall \delta$（性质成立）
- Defense：构造 f_θ 使 Verification 成功

1.3 Robustness Verification

1.3.1 问题定义

Robustness Verification

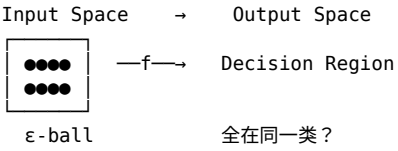
给定网络 f 和输入规格[1] $\Phi(x)$，验证：

$$\forall x' \in \Phi(x) : f(x') = f(x)$$

其中 $\Phi(x)$ 通常是 ℓ_p 球：$\Phi(x) = \left\{ x' \;\middle|\; \|x' - x\|_p \le \varepsilon \right\}$

为什么难？考虑 MNIST：
- 输入 dim：784
- 可能的扰动：2^{784} 种（穷举不可行）
- 即使是 ℓ_∞ 球，内部点数也趋近无穷

解决思路：不枚举每个点，而是**把整个凸形状推过网络**。



1.3.2 Certification 方法对比

两类主要方法提供不同类型的保证：

	Convex Methods	Randomized Smoothing
原理	传播凸集合通过网络	采样 + 统计保证
是否需要特殊训练	需要 Certified Training	推理时即可使用
可验证性质	多种（robustness, fairness 等）	有限
可扩展性	小到中型网络	大 model（包括 LLM）
保证类型	确定性	prob 性

Table 2: 两类 Certification 方法对比

⚄ 选择哪种方法取决于应用场景：
- 需要确定性保证：Convex Methods
- 需要扩展到大 model：Randomized Smoothing
- 需要训练时优化：Certified Training

1.4 Min-Max 优化框架

1.4.1 Standard Training

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(f_\theta(x), y)]$$

只关心单点 x 的准确率。

1.4.2 Robust Training

$$\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{x'\in\Phi(x)} \mathcal{L}(f_\theta(x'), y)\right]$$

关心整个 $\Phi(x)$ 内的 worst-case。

Min-Max 双层优化

Robust Training 是嵌套优化问题：
- **Inner max**：在 $\Phi(x)$ 内找 worst-case 扰动（攻击者视角）
- **Outer min**：优化 model 参数 θ 抵御最坏情况（防御者视角）

两者形成对抗博弈。

这个框架统一了三节课的内容：

任务	Inner Max	Outer Min	方法
Attack	找 worst-case δ^*	—（θ 固定）	FGSM/PGD/C&W
Adversarial Training	PGD 生成对抗 sample	SGD 更新权重	PGD-AT
Certified Training	Convex Relaxation 推区域	优化 certified loss	IBP/CROWN

Table 3: Min-Max 框架的三种实例化

1.5 Robustness ≈ Individual Fairness

技术等价性

Robustness 和 Individual Fairness 在数学上等价，区别仅在距离度量 d 的定义：

Robustness：$\forall x' : d(x, x') \le \varepsilon \Rightarrow f(x') = f(x)$

[1] Input Specification，定义了允许的扰动范围

Individual Fairness：$\forall x' : d_{\text{sensitive}}(x, x') \le \varepsilon \Rightarrow f(x') = f(x)$

实际意义：同一套 Convex Relaxation 技术可以用于：
- Robustness：d 是像素级 ℓ_p 距离
- Fairness：d 只看敏感属性（如种族、性别）的差异
- Quantization：d 是量化误差范围

1.6 Differential Privacy

$(epsilon, delta)$-Differential Privacy
算法 M 满足 (ε, δ)-DP，若对所有相差一条记录的数据库 D, D'：

$$P[M(D) \in S] \le e^{\varepsilon} \cdot P[M(D') \in S] + \delta$$

直觉：加入/移除一个人，输出分布变化很小 → 无法推断个体是否在数据中。

符号	含义	备注
M	随机化算法	如 DP-SGD 训练过程
D, D'	相差一条记录的数据库	“邻居”dataset
ε	隐私预算	越小越隐私
δ	失败 prob	通常取 $\ll \frac{1}{n}$

Table 4: DP 符号说明

2 Part 2: Verification

2.1 Verification 问题形式化

Formal Verification Problem

$$\forall i \in I : \varphi(i) \Longrightarrow N(i) \vDash C$$

对所有输入 i，若满足前条件 $\varphi(i)$，则网络输出 $N(i)$ 满足后条件 C。

Sound
若方法说“成立”，则性质**确实成立**。

$$\text{Proved} \Rightarrow \text{True}$$

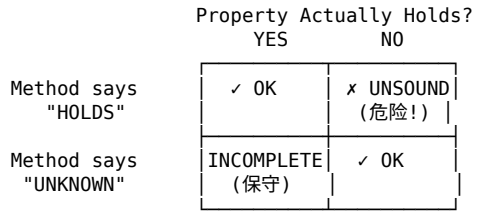
宁可多说“不知道”（保守），也不能误报“安全”。

Complete
若性质**确实成立**，则方法**能够证明**。

$$\text{True} \Rightarrow \text{Provable}$$

不会遗漏可证明的性质。

大多数实用方法是 **Sound but Incomplete**：
- 说“安全”时可信
- 说“不知道”时不代表不安全，可能只是方法能力有限



2.2 Box Relaxation (IBP)

Interval Bound Propagation
用区间 $[l, u]$ 表示神经元可能的取值范围。区间形成 hyper-rectangle（Box）。

$$\text{Box}: \quad [l_1, u_1] \times [l_2, u_2] \times \cdots \times [l_n, u_n]$$

2.2.1 Abstract Transformers

对每种运算定义在区间上的操作：

操作	符号定义
加法	$[a, b] \oplus [c, d] = [a + c, b + d]$
取负	$-[a, b] = [-b, -a]$
标量乘	$\lambda[a, b] = \begin{cases} [\lambda a, \lambda b] \lambda \ge 0 \\ [\lambda b, \lambda a] \lambda < 0 \end{cases}$
ReLU	$\text{ReLU}([l, u]) = [\max(0, l), \max(0, u)]$

Table 5: Box Abstract Transformers

2.2.2 Affine 层的传播

对于 $\boldsymbol{z} = W\boldsymbol{x} + \boldsymbol{b}$，精确计算区间：

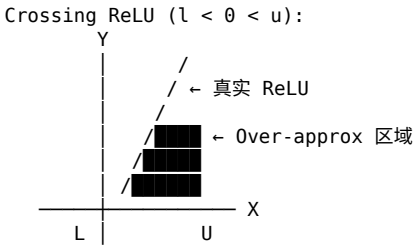
$$[\boldsymbol{l}', \boldsymbol{u}'] = [W^+\boldsymbol{l} + W^-\boldsymbol{u} + \boldsymbol{b}, \quad W^+\boldsymbol{u} + W^-\boldsymbol{l} + \boldsymbol{b}]$$

其中 $W^+ = \max(W, 0)$，$W^- = \min(W, 0)$。

2.2.3 Crossing ReLU

Crossing ReLU
若 ReLU 输入的 bounds 满足 $l < 0 < u$，则称该 ReLU 处于 **crossing** 状态。
非 crossing 情况更简单：

- $l \geq 0$：恒正，$y = x$（直接传递）
- $u \leq 0$：恒负，$y = 0$（输出恒为 0）



Crossing ReLU 引入 **over-approximation**：
- 真实可达集（黄色碎片）
- 近似包络（紫色 Box）
- 紫色 ⊇ 黄色（**Sound**）
- 额外区域称为 “garbage points”

⚠ Over-approximation 误差随深度**累积**：

$$\text{Error}_k = f(\text{Error}_{k-1}, \text{Layer}_k)$$

深度是 bound propagation 的敌人。

2.3 MILP 编码

Mixed Integer Linear Programming 通过引入二进制变量实现 **Sound & Complete** 验证。

Crossing ReLU 的 MILP 编码

对于 crossing ReLU（$l < 0 < u$），引入二进制变量 $a \in \{0, 1\}$：

$$y \geq x$$
$$y \leq x - l(1 - a)$$
$$y \leq u \cdot a$$
$$y \geq 0$$

- 当 $a = 1$：约束简化为 $y = x$（active branch）
- 当 $a = 0$：约束简化为 $y = 0$（inactive branch）

MILP 复杂度

复杂度为 $O(2^k)$，其中 k 是 **Crossing ReLU 数量**，而非总神经元数。

$$\text{Complexity} \propto 2^{|\{\text{Crossing ReLUs}\}|}$$

⚠ 这意味着：
- 更 tight 的 bounds → 更少 crossing → MILP 更快
- Box/DeepPoly 预计算 bounds 可以大幅减少 MILP 分支数
- 论文声称“验证百万神经元”时，要检查 crossing 数量和网络准确率

2.3.1 具体例子

给定：$x_1 \in [0, 0.5]$，$x_2 \in [0.2, 0.7]$

Affine 层：$x_3 = x_1 + x_2$，$x_4 = x_1 - x_2$

Box 传播：

$$x_3 \in [0 + 0.2, 0.5 + 0.7] = [0.2, 1.2] \quad (\text{非 crossing}, l \geq 0)$$
$$x_4 \in [0 - 0.7, 0.5 - 0.2] = [-0.7, 0.3] \quad (\text{Crossing } ! \, l < 0 < u)$$

结论：x_3 无需分支（$y = x$），x_4 需要 MILP 二进制变量。

2.4 DeepPoly Relaxation

DeepPoly 是介于 Box 和 MILP 之间的方法：比 Box 精确，比 MILP 快。

Linear Symbolic Bounds

每个神经元 X_j 维护 four **constraints**：

$$X_j \geq \sum_i a_i^L X_i + b^L \quad (\text{下界线性约束})$$
$$X_j \leq \sum_i a_i^U X_i + b^U \quad (\text{上界线性约束})$$
$$X_j \geq L_j \quad (\text{具体下界})$$
$$X_j \leq U_j \quad (\text{具体上界})$$

⚠ 为什么需要具体 bounds L_j, U_j?
- 判断 ReLU 是否 crossing
- 随时停止 back-substitution（不必回溯到输入层）
- 计算效率：$O(1)$ 的 ReLU transformer

2.4.1 Affine 层

对于 $\boldsymbol{z} = W\boldsymbol{x} + \boldsymbol{b}$，DeepPoly 是 **Exact**（无损）：

$$\boldsymbol{z} \leq W\boldsymbol{x} + \boldsymbol{b} \leq \boldsymbol{z} \quad (\text{upper} = \text{lower})$$

2.4.2 ReLU 层

对于 crossing ReLU $Y = \text{ReLU}(X)$，其中 $X \in [l, u]$ 且 $l < 0 < u$：

Upper bound（固定）：

$$Y \leq \frac{u}{u - l}(X - l) = \lambda X - \lambda l, \quad \text{where } \lambda = \frac{u}{u - l}$$

Lower bound（可选/可优化）：

$$Y \geq \alpha X, \quad \text{where } \alpha \in [0, 1]$$

⚠ 这个 α 就是 α-CROWN 中的 α！它是可优化参数。
- $\alpha = 0$：$Y \geq 0$
- $\alpha = 1$：$Y \geq X$
- 中间值：精度与速度的 trade-off

2.4.3 Back-Substitution

核心算法：递归展开线性约束直到输入层，获得更 tight 的 bounds。

⚙ Back-Substitution

计算神经元 X_j 的具体 bounds $[L_j, U_j]$：
1. 获取 X_j 的线性上界：$X_j \leq \sum_i c_i X_i + d$
2. 对每个 X_i，用其自身的线性约束替换
3. 递归直到到达输入层
4. 用输入 bounds 计算最终数值

⚠ **符号反转陷阱**：计算 upper bound 时，若系数为**负**，需要用变量的 **lower** bound：

$$\text{If} \quad X_j \leq -X_i + \ldots, \quad \text{then substitute} \quad X_i \geq \ldots \ (\text{lower bound})$$

原因：$-X_i$ 要最大化，需要 X_i 最小化。

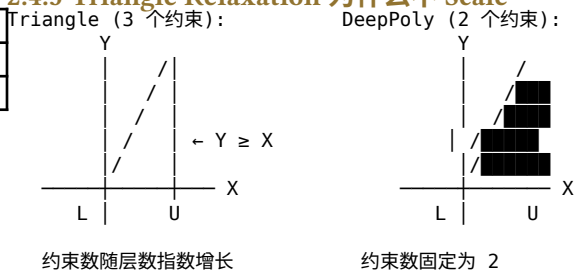
2.4.4 Single-Neuron vs Multi-Neuron

类型	依赖关系	并行性	精度
Single-Neuron	仅依赖前一层	完全并行（GPU 友好）	较低
Multi-Neuron	可用同层神经元约束	串行	较高

Table 6: Relaxation 类型对比

DeepPoly 采用 Single-Neuron，牺牲精度换取并行性。

2.4.5 Triangle Relaxation 为什么不 Scale



约束数随层数指数增长　　约束数固定为 2

2.5 α-β-CROWN

2.5.1 Lagrange Multiplier 方法

问题：朴素 split 会丢失关系信息（DeepPoly 只能维护一对约束）。

对于 split 约束 $X \geq 0$（即 $-X \leq 0$）：

$$\max_X f(X) \quad \text{s.t.} \quad X \geq 0$$

转化为 Lagrangian：

$$\max_X \min_{\beta \geq 0} [f(X) + \beta \cdot X]$$

由 **Weak Duality**：

$$\max_X \min_\beta [...] \leq \min_\beta \max_X [...]$$

右侧更好处理：

- $\max_X$ 可通过 back-substitution 计算
- $\min_\beta$ 可用 gradient 下降优化

$alpha$-$beta$-CROWN 组成

- α：ReLU 下界斜率参数，$\in [0, 1]$，可 gradient 优化
- β：Lagrange 乘子，≥ 0，用于编码 split 约束
- CROWN：DeepPoly 框架

⚠ 关键性质：α 和 β 只影响 **tightness**，不影响 **soundness**。

- 任意 $\alpha \in [0, 1]$ 都是 sound 的 ReLU relaxation
- 任意 $\beta \geq 0$ 都给出 sound 的 upper bound

2.6 Floating-Point Soundness

⚠ 很多"Sound" verifier 在浮点运算下实际是 **Unsound** 的！

- 理论：MILP 在实数 $\mathbb{R}$ 上是 Sound & Complete
- 现实：硬件用 IEEE-754 浮点数，有 rounding error

$$\text{Sound}_{\text{theory}} \neq \text{Sound}_{\text{hardware}}$$

2.7 Branch & Bound 算法

Branch & Bound for NN Verification

核心思想：结合 Relaxation（快但 incomplete）和 Splitting（慢但 complete）

算法流程：

1. 用 DeepPoly/CROWN 计算 bounds（**Bound** 阶段）
2. 若证明成功 → 返回 SAFE
3. 若 bounds 不够紧 → 选择一个 unstable ReLU split（**Branch** 阶段）
4. 递归处理两个子问题（$X \geq 0$ 和 $X < 0$）

2.7.1 算法伪代码

⚙ Branch & Bound

```
def verify(spec, model, bounds):
    # 1. Bound: 尝试用 relaxation 证明
    lb, ub = compute_bounds(model, bounds)  # DeepPoly/
CROWN

    if lb > 0:  # 所有输出 > 0
        return SAFE
    if ub < 0:  # 存在必然违反
        return UNSAFE (with counterexample)

    # 2. Branch: 选择最不稳定的 ReLU
    neuron = select_unstable_relu(bounds)  # 启发式选择

    # 3. 递归
    result_pos = verify(spec, model, bounds ∪ {neuron ≥ 0})
    if result_pos == UNSAFE:
        return UNSAFE

    result_neg = verify(spec, model, bounds ∪ {neuron < 0})
    return result_neg
```

2.7.2 Branching 启发式

启发式	选择标准	直觉
Largest Interval	$\max(u - l)$	区间最大的最不确定
Closest to Zero	$\min(\lvert l\rvert, \lvert u\rvert)$	最接近 0 的最关键
Gradient-based	$\max\lvert\nabla_x \text{ objective}\rvert$	对目标影响最大
Learning-based	神经网络预测	从历史学习

2.7.3 复杂度分析

Branch & Bound 复杂度

最坏情况：$O(2^k)$，其中 k 是 unstable ReLU 数量

实际表现：取决于

- Bounds 的 tightness（越紧需要 branch 越少）
- Branching 启发式的质量
- 问题本身的结构

关键优化：用 α-β-CROWN 在 runtime 优化 bounds，减少 branch 次数

2.8 VNN-COMP 竞赛批判分析

⚠ **读论文时必须警惕的指标陷阱！**

论文声称 "验证了 68,000,000 参数网络" 时，立即检查：

2.8.1 需要检查的指标

1. **Crossing ReLU 有多少?**
   - 若只有 10 个 $\rightarrow 2^{10} = 1024$
   - 复杂度取决于 crossing 数而非总参数
2. **网络准确率是多少?**
   - 过度正则化的网络容易验证
   - 但实际 accuracy 可能很低
3. **用了什么 specification?**
   - 小 $\varepsilon \rightarrow$ 更少 crossing
   - $\varepsilon = 0.001$ 比 $\varepsilon = 0.3$ 验证简单得多

2.8.2 Critical Thinking

⚠ **Sound but Impractical**:
- 论文可能只验证了特殊网络
- 在 standard benchmark 上可能失败

Complete but Slow:
- MILP 理论上 complete
- 但 timeout = 3600s 后也算"证明失败"

Verified $\neq$ Practically Robust

2.8.3 VNN-COMP 常见问题

问题	警示
Network 太小	只在 tiny network 上验证，无法泛化
Epsilon 太小	$\varepsilon = \frac{2}{255}$ 对 ImageNet 几乎没意义
Timeout 太长	3600s 证明一个 sample 不实用
Certified Accuracy 低	验证成功但只有 30% certified

2.9 Part 2 易错点补充

α 在 α-β-CROWN 中的作用：控制 ReLU 下界斜率，$\alpha \in [0, 1]$，可 gradient 优化
β 的物理意义：Lagrange 乘子，编码分支约束 $X \geq 0$
Weak Duality：$\max\min \leq \min\max$（总是成立）
为什么 tighter relaxation 不一定更好?：更紧 = 更难优化 = 训练可能失败
Certified Accuracy 的局限：只衡量"能证明安全"的比例，不是"实际安全"的比例
Branch & Bound 的 bottleneck：不是神经元总数，而是 **unstable ReLU 数量**

2.10 三种攻击方法对比

方法	步数	范数约束	优化目标	典型用途
FGSM	1	ℓ_∞（固定）	快速启发式	快速评估脆弱性
C&W	多步优化	ℓ_2（最小化）	最小扰动	精确攻击
PGD	10-20	ℓ_∞（投影）	最大化 loss	攻击 + Adversarial Training

Table 9: 三种攻击方法对比

⚠ 核心关系：PGD = FGSM × K 迭代 + 投影
C&W 代表另一种哲学：最小化扰动大小，而非固定扰动预算。

2.11 Targeted vs Untargeted Attack

2.11.1 Targeted Attack

目标：使 model 输出**特定**错误类别 $t \neq y$

$$\eta^* = \arg\min_\eta \|\eta\|_p \quad \text{s.t.} \quad f(x+\eta) = t$$

优化方向：**靠近**目标类（gradient 下降）

2.11.2 Untargeted Attack

目标：使 model 输出**任意**错误类别

$$\eta^* = \arg\min_\eta \|\eta\|_p \quad \text{s.t.} \quad f(x+\eta) \neq y$$

优化方向：**远离**正确类（gradient 上升）

2.12 FGSM

Fast Gradient Sign Method

Targeted:

$$x' = x - \varepsilon \cdot \text{sign}(\nabla_x \mathcal{L}(f(x), t))$$

Untargeted:

$$x' = x + \varepsilon \cdot \text{sign}(\nabla_x \mathcal{L}(f(x), y))$$

2.12.1 为什么用 Sign 函数?

1. **归一化效应**：gradient 各 dim 量级差异巨大，sign 统一为 $\{-1, 0, +1\}$
2. **最大步长**：在 ℓ_∞ 约束下，每个像素都走到 box 边界
3. **单步最优**：一阶 Taylor 展开下，这是 ℓ_∞ 约束的最优单步移动

```
gradient空间                Sign 空间
[100, 0.01, -50]  →  [1, 1, -1]
不同尺度          →  统一步长 ε
连续实数向量      →  离散方向集

几何意义：跳到 ℓ∞ 球的顶点
```

2.13 C&W Attack

Carlini & Wagner

原问题（难优化）:

$$\min_\eta \|\eta\|_p \quad \text{s.t.} \quad f(x+\eta) = t$$

松弛后（连续可优化）:

$$\min_\eta \|\eta\|_p^2 + c \cdot \text{OPS}(x+\eta, t)$$

其中 $\text{OPS}(x', t) = \max\left(0, \max_{i \neq t} Z(x')_i - Z(x')_t + \kappa\right)$

2.13.1 OPS 函数的契约

$$\text{OPS}(x', t) \leq 0 \quad \longrightarrow \quad f(x') = t$$

这是**单向蕴含**（Sound but Incomplete）:
- OPS $\leq 0 \rightarrow$ 攻击必然成功
- OPS $> 0 \rightarrow$ 不一定失败

⚠ 参数 κ（margin）控制 confidence:
- $\kappa = 0$：只要分类正确即可
- $\kappa > 0$：目标类 logit 至少比其他类大 κ

2.14 PGD

⚙ Projected Gradient Descent

```
初始化: x₀ = x + random_in_ε_box
for k = 1 to K:
    # 1. FGSM step
    g_k = ∇_x L(f(x_{k-1}), y)
    x'_k = x_{k-1} + α · sign(g_k)

    # 2. Projection (关键!)
```

对于 ℓ_∞ 范数，投影操作非常简单:

$$\Pi_{\mathcal{B}_\varepsilon(x)}(z) = \text{clip}(z, x-\varepsilon, x+\varepsilon)$$

逐 dim 裁剪：超出范围则拉回边界。

⚠ 投影复杂度是关键:
- ℓ_∞ 球：$O(n)$（逐 dim clip）
- ℓ_2 球：$O(n)$（归一化）
- 复杂凸多面体：可能需要 QP solver

这是 Certified Training 的计算瓶颈之一。

```
    x_k = Π_{B_ε(x)}(x'_k)
return x_K
```

2.15 Adversarial Training

Adversarial Training (PGD-AT)

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\max_{\delta\in\mathcal{B}_\varepsilon(0)} \mathcal{L}(f_\theta(x+\delta), y)\right]$$

- **Inner max**：用 PGD 找 worst-case 对抗 sample
- **Outer min**：在对抗 sample 上做标准 SGD

2.15.1 伪代码

```
for (x, y) in train_loader:
    # Inner Max: 找最难的对抗sample
    x_adv = PGD_attack(x, model, epsilon, steps=10)

    # Outer Min: 在对抗sample上训练
    loss = CrossEntropy(model(x_adv), y)
    loss.backward()  # 对 θ 求gradient
    optimizer.step()
```

2.15.2 Contrastive Learning 视角

Adversarial Training 可理解为对抗性对比学习：

- **Anchor**：原始输入 x
- **Positive Bag**：$\mathcal{B}_\varepsilon(x)$ 内所有点（语义应保持不变）
- **Hard Negative**：PGD 在 $\mathcal{B}_\varepsilon(x)$ 内找到的最大 loss 点

传统对比学习采样**有限个**负 sample；Adversarial Training 对**整个** ε-**球**都鲁棒。

2.16 TRADES：精度-鲁棒性权衡

TRADES Loss

$$\mathcal{L}_{\text{TRADES}} = \underbrace{\mathcal{L}(f(x), y)}_{\text{Natural Accuracy}} + \lambda \underbrace{\max_{x'\in\mathcal{B}_\varepsilon} \text{KL}(f(x) \,\|\, f(x'))}_{\text{Robustness}}$$

核心思想：自然准确率和鲁棒性**分开**优化，用 λ 权衡。

2.16.1 PGD-AT vs TRADES

	PGD-AT	TRADES
Loss	$\max \mathcal{L}(f(x'), y)$	$\mathcal{L}(f(x), y) + \lambda$ KL
目标	最小化最坏情况	平衡精度与鲁棒性
Clean Acc	较低	较高
Robust Acc	较高	中等

2.16.2 选择指南

- **需要最高鲁棒性**：使用 PGD-AT
- **需要平衡**：使用 TRADES + 调节 λ
- λ 越大 $\rightarrow$ 越关注鲁棒性
- 典型值：$\lambda \in [1, 6]$

实践中 TRADES 通常在 clean-robust 权衡曲线上表现更好。

2.17 AutoAttack：可靠的攻击评估

AutoAttack Ensemble

组成[2]：

1. **APGD-CE**：Auto-PGD with CE loss
2. **APGD-DLR**：Auto-PGD with Difference of Logits Ratio loss
3. **FAB**：Fast Adaptive Boundary attack
4. **Square Attack**：Black-box query-based attack

为什么需要 AutoAttack?

- 单一攻击可能被"过拟合防御"绕过
- 论文可能选择性报告弱攻击结果
- AutoAttack 提供**标准化评估**

使用规则：

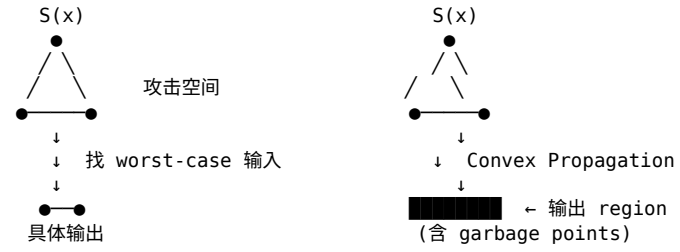
- 报告 Robust Accuracy 时必须用 AutoAttack
- 自定义攻击结果只能作为**补充**

3 Part 4: Certified Training

3.1 PGD Training vs Certified Training

	PGD Training	Certified Training
优化空间	输入空间 $S(x)$	输出空间 $\gamma(f^\sharp(S(x)))$
Inner max	$\max_{x'\in S(x)} \mathcal{L}(f(x'), y)$	$\max_{z\in\gamma(f^\sharp(S(x)))} \mathcal{L}(z, y)$
使用的点	具体对抗 sample	符号区域（含 garbage points）
保证类型	Heuristic（可能 miss attacks）	Sound（可证明保证）
计算方式	具体前向传播	符号前向传播

Table 11: 训练范式对比



[2]设计思想：组合多种攻击以避免假阳性（误以为 model 鲁棒）

3.2 Certified Training Paradox

实验发现：**更 tight 的 relaxation 反而导致更差的训练结果!**

Relaxation	Tightness	Certified Accuracy
Box (IBP)	Low	86%
Zonotope	Medium	73%
DeepPoly	High	70%

Table 12: Tightness vs Training Performance（反直觉!）

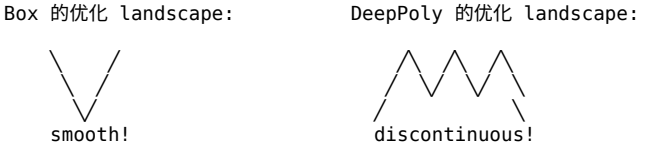
3.2.1 原因分析

Sensitivity（敏感性）：

- DeepPoly 有 discrete switching（选 α 时）
- 权重小变化 $\rightarrow$ bounds 剧变
- gradient 不稳定

Discontinuity（不连续性）：

- 复杂 relaxation 引入更多不连续点
- 优化 landscape 更难 navigate



反直觉但重要：**Tightness ≠ Optimizability**
Box 虽然松（精度低），但 gradient 平滑，反而更好优化。

↓
在这里找 worst-case

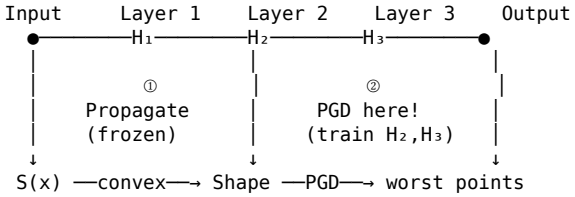
3.3 SABR: Layer-wise Training

核心思想：在中间层做 PGD，而非在输出空间优化。

⚙ SABR Method

对于每层 k：

1. 用 convex relaxation 将输入 spec 传播到第 k 层
2. 冻结第 k 层之前的参数
3. 在 intermediate shape 上做 PGD（只训练后面的层）
4. 更新第 k 层及之后的权重



⚠ 投影问题：PGD 需要投影到 $S(x)$，但中间层 shape 可能不是 ℓ_∞ 球！

- ℓ_∞ ball 投影：简单 clip
- DeepPoly shape 投影：需要解 QP

解决方案：用 Zonotope（可高效投影）。

3.4 Logic → Loss Translation

逻辑约束到 loss 函数

任意逻辑公式 φ 可翻译为 loss L_φ，满足：

$$L_\varphi(x) = 0 \quad \Leftrightarrow \quad x \vDash \varphi$$

⚠ 这提供了处理任意 safety specs 的统一框架：

- Adversarial attack：$\exists \delta : \|\delta\|_\infty \le \varepsilon \wedge \arg\max f(x+\delta) \neq y$
- Robustness verification：$\forall \delta : \|\delta\|_\infty \le \varepsilon \Rightarrow f(x+\delta) = y$
- 训练：$\min_\theta \max_{z \in S(x)} L_{\neg\varphi}(z)$

公式 φ	loss L_φ
$t_1 = t_2$	$(t_1 - t_2)^2$
$t_1 \le t_2$	$\max(0, t_1 - t_2)^2$
$\varphi_1 \wedge \varphi_2$	$L_{\varphi_1} + L_{\varphi_2}$
$\varphi_1 \vee \varphi_2$	$L_{\varphi_1} \cdot L_{\varphi_2}$

Table 13: Logic → Loss 翻译表

4 Part 5: Randomized Smoothing & GCG Attack

4.1 Randomized Smoothing

4.1.1 核心思想

Smoothed Classifier

给定 base classifier F（黑盒），构造 smoothed classifier G：

$$G(x) = \arg\max_c \mathbb{P}_{\varepsilon \sim \mathcal{N}(0,\sigma^2 I)}[F(x+\varepsilon) = c]$$

直觉：对每个输入，采样大量 Gaussian noise 扰动，用 majority vote 决定输出。

关键区分：

- **定理保证是 deterministic**（数学证明）
- **实践估计是 probabilistic**（采样 Monte Carlo）

不要混淆这两者！

Base Classifier F（可能脆弱）
↓
□ Gaussian noise包裹
↓
Smoothed Classifier G（构造出鲁棒性）

4.1.2 Certified Radius

认证半径公式

设 $\underline{p_A}$ 为最高类 prob 的下界，且 $\underline{p_A} > 0.5$，则：

$$R = \sigma \cdot \Phi^{-1}(\underline{p_A})$$

其中 Φ^{-1} 是标准正态 CDF 的逆函数（probit function）。

⚠ 增大 σ 不一定增大 R！

- 直接效应：σ 项增大
- 间接效应：p_A 降低（噪声大，分类散乱）

存在最优 σ^*，需 empirical tuning。

4.1.3 两阶段采样

⚙ Certification Pipeline

Stage 1 (Exploration, $n_0 \approx 100$):

```
votes = [F(x + noise) for _ in range(n0)]
c_A = most_common(votes)  # 猜测 top class
```

Stage 2 (Certification, $n \approx 10^5$):

```
votes = [F(x + noise) for _ in range(n)]
p_A_hat = count(votes == c_A) / n
p_A_lower = binomial_CI_lower(p_A_hat, n, α)
if p_A_lower > 0.5:
    R = σ * Φ⁻¹(p_A_lower)
else:
    return "Abstain"
```

4.1.4 Deterministic vs Randomized Smoothing

	Deterministic (CROWN)	Randomized Smoothing
保证类型	100% 确定	$(1-\alpha)$ confidence
model 假设	需知道权重、激活函数	黑盒即可
可扩展性	小网络（精度爆炸）	任意大小（包括 LLM）
范数	$\ell_\infty, \ell_1, \ell_2$ 皆可	主要 ℓ_2（对应 Gaussian）

Table 14: Certification 方法对比

4.1.5 为什么 RS 主要限于 ℓ_2?

Gaussian 噪声与 ell_2 的数学联系
Gaussian 分布具有**旋转不变性**[3]:

$$X \sim \mathcal{N}(0, \sigma^2 I) \Rightarrow \|X\|_2 \text{ 与方向无关}$$

这导致 Neyman-Pearson 最优检测器在 ℓ_2 球上均匀，从而得到 ℓ_2 certified radius。
其他范数的困难:

- ℓ_∞：需要 discrete/uniform noise，但 prob 界更弱
- ℓ_1：需要 Laplace noise，但 certified radius 公式更复杂

⚠ 不要与 DP 中的 Laplace vs Gaussian 混淆!

- DP 中：Laplace 对应 ℓ_1 **敏感度**，Gaussian 对应 ℓ_2 **敏感度**
- RS 中：Gaussian 对应 ℓ_2 **certified radius**

4.1.6 DP 与 RS 的对偶性（Prof 强调!）

同一枚硬币的两面
DP 和 RS 使用**相同的数学工具**（噪声机制、指数界），但**优化方向相反**:

	Differential Privacy	Randomized Smoothing
目标	使分布**不可区分**	使预测**可区分**
数学	$P[M(D)] \approx P[M(D')]$	$P[G(x)=c] \gg P[G(x)\neq c]$
噪声作用	混淆真实数据	平滑决策边界
假设检验	希望 Power **低**	希望置信度 **高**

4.1.7 共同的 Lipschitz 基础

两者的证明都依赖 Lipschitz 常数 L:

- DP：L 控制敏感度 $\rightarrow$ 决定噪声量
- RS：L 控制 p_A 随 x 变化 $\rightarrow$ 决定认证半径

$$\text{DP Noise} \propto \frac{L}{\varepsilon}, \quad \text{RS Radius} \propto \frac{\sigma}{L}$$

4.1.8 考试常考对比

⚠ **常见陷阱**:

- 误以为 RS 是 probabilistic method（定理是确定性的!）
- 混淆 DP 和 RS 的噪声含义
- 忘记 ℓ_2 限制的数学原因

4.1.9 常见失败模式

Case	问题	解决方案
猜错 top class	n_0 太小	增大 n_0（100-1000）
$p_A < 0.5$	Base model 在噪声下表现差	Gaussian Adversarial Training
Lower bound 太松	真实 $p_A = 0.52$，估计 $\underline{p_A} = 0.45$	增大 n（10k $\rightarrow$ 100k）

4.1.10 核心挑战

LLM 输入是 discrete tokens，不能直接用 PGD（需连续空间）。

GCG 优化目标
找 suffix 使 model 生成有害内容:

$$\min_{\text{suffix}} \mathcal{L}_{\text{CE}}(y_{\text{target}} = \text{Sure} \mid \text{prompt}, \text{suffix})$$

4.1.11 Three-Step Algorithm

⚙ GCG Algorithm

Step 1: One-hot gradient 计算（关键 trick）

- 把 token 变成 one-hot vector $e \in \mathbb{R}^{|V|}$
- 计算 $\nabla_e \mathcal{L}$（连续空间）

Step 2: Top-K 筛选

- 选 gradient 最负的 K 个 tokens 作为候选
- 从 50k 词表筛到 256 个

Step 3: Greedy Search

```
for position i in suffix:
    for token in top_k_candidates:
        suffix[i] = token
        loss = evaluate(prompt + suffix)
        keep best
```

[3] 数学上：$\mathcal{N}(0, \sigma^2 I)$ 在正交变换下不变

⚠ GCG 不是在连续空间更新，而是用 gradient 作为启发式筛选候选，再回到离散空间做 greedy search。

4.1.12 White-box vs Black-box

	White-box GCG	Black-box
gradient	可用	不可用
初始化	随机即可	需强初始化（FIM Inversion）
速度	快（Top-K 筛选）	慢（盲目搜索）

Black-box 策略：用 Fill-in-the-Middle model 做 inversion attack，先写恶意代码，反推 prefix 作为强初始化。

4.1.13 Universal & Transferable Suffix

$$\min_{\text{suffix}} \sum_{i=1}^{M} \mathcal{L}(\text{Sure} \mid \text{prompt}_i, \text{suffix})$$

在多个 prompts 上同时优化 → universal suffix → 可 transfer 到其他 model（甚至 GPT-4）。

4.2 Mixed Adversarial Training

4.2.1 动机

Attack Type	Speed	Strength	Realistic?
Continuous	快（gradient 下降）	中等	否（token+0.1 无意义）
Discrete (GCG)	慢（greedy search）	强	是（真实攻击）

Mixed-AT Loss

$$\mathcal{L}_{\text{total}} = \underbrace{\mathcal{L}_{\text{clean}}(x, y)}_{\text{保持效用}} + \underbrace{\mathcal{L}_{\text{robust}}(x_{\text{adv}}, y_{\text{safe}})}_{\text{鲁棒性}} + \underbrace{\mathcal{L}_{\text{refuse}}(x_{\text{adv}}, y_{\text{refuse}})}_{\text{拒绝恶意}}$$

策略：

1. Discrete attack（GCG）生成强对抗 sample 作为 anchor
2. Continuous attack 生成大量变种扩充多样性
3. 结合两者训练 → ASR 从 50% 降到 $< 10\%$

4.3 Post-Training Attacks

4.3.1 Quantization Attack

利用量化前后行为差异植入后门：

机制：

1. 训练 malicious model
2. 计算 Box Constraint：$[w_{\text{low}}, w_{\text{high}}]$ 使量化值不变
3. 在 box 内用 clean data fine-tune，使 FP32 表现正常

结果：

- FP32：benign（通过检测）
- INT8：malicious（量化后激活）

Defense 盲点：检测在 FP32，部署用 INT8。

4.3.2 Fine-Tuning Attack

利用 Meta-Learning 在用户微调后激活后门：

$$\mathcal{L} = \underbrace{\mathcal{L}_{\text{clean}}(\theta)}_{\text{现在安全}} + \lambda \underbrace{\mathcal{L}_{\text{attack}}(\theta - \nabla \mathcal{L}_{\text{user}}(\theta))}_{\text{未来恶意}}$$

⚠ 需要二阶导数（Hessian），计算成本高：

$$\frac{\partial \mathcal{L}(\theta')}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \theta'} \cdot (I - \eta \nabla^2 \mathcal{L}_{\text{user}})$$

5 Part 6: 考试要点

5.1 核心概念速查

5.1.1 Sound vs Complete

- **Sound**：证明成立 → 确实成立（不会误报安全）
- **Complete**：确实成立 → 能够证明（不会漏报可证性质）
- 大多数实用方法：Sound but Incomplete

5.1.2 Crossing ReLU

- 定义：输入 bounds $l < 0 < u$
- MILP 复杂度：$O(2^k)$，k = Crossing ReLU 数量（非总神经元数）
- 减少方法：更 tight 的 bounds，Certified Training

5.1.3 Min-Max 结构

$$\min_{\theta} \max_{\delta \in \Delta} \mathcal{L}(f_\theta(x + \delta), y)$$

- Attack：固定 θ，找 δ
- Defense：同时优化两者
- Certification：用 convex relaxation 替代 inner max

5.1.4 参数含义

- α：ReLU 下界斜率（$\in [0, 1]$），可优化
- β：Lagrange 乘子（≥ 0），编码 split 约束
- 两者只影响 tightness，不影响 soundness

5.2 方法对比表

方法	Sound	Complete	复杂度	GPU
Box/IBP	√	✗	$O(n)$	√
DeepPoly	√	✗	$O(n^3 L^2)$	√
MILP	√	√	$O(2^k)$	✗
RS	统计	–	$O(n_{\text{samples}})$	√

5.3 易错点

Crossing ReLU 数量：复杂度取决于 crossing neurons，不是总神经元数

Back-substitution 符号：负系数需要用 opposite bound

浮点 soundness：$\text{Sound}_{\text{theory}} \neq \text{Sound}_{\text{hardware}}$

Training paradox：更 tight ≠ 更好优化

RS 保证类型：定理是 deterministic，估计是 probabilistic

增大 σ：不一定增大 R（p_A 会下降）

GCG vs PGD：GCG 用 gradient 筛选，不是用 gradient 更新

n_0 vs n：Classification (n_0) vs Estimation (n)，信息复杂度不同

6 Part 7: Privacy & Differential Privacy

6.1 Differential Privacy 核心思想

DP 的对抗博弈视角

Hypothesis Testing 视角：

攻击者（Membership Inference）：
- H_0：数据点 x 不在 training set D 中
- H_1：数据点 x 在 training set D 中
- 目标：从 $M(D)$ 区分两种情况

防御者（DP）：
- 使 $P[M(D) \in S] \approx P[M(D') \in S]$
- 攻击者的检验功效 ≈ 随机猜测

```
┌──────────────────────────┐
│ H₀: x ∉ Train (Out)      │
│ H₁: x ∈ Train (In)       │
├──────────────────────────┤
│ 攻击者观察: M(D) 或 M(D')│
│ 做出判断 → Win if b̂ = b  │
└──────────────────────────┘
```

DP目标：使判断≈随机猜测

6.1.1 ε-DP 黄金公式

$epsilon$-Differential Privacy

$$\forall S, \forall (D, D') \text{ neighbors}: \quad P[M(D) \in S] \le e^{\varepsilon} \cdot P[M(D') \in S]$$

当 ε 很小时：$e^{\varepsilon} \approx 1 + \varepsilon$

双边界（利用邻居对称性）：

$$(1-\varepsilon)P[M(D') \in S] \le P[M(D) \in S] \le (1+\varepsilon)P[M(D') \in S]$$

6.1.2 (ε, δ)-DP 放松

$(epsilon, delta)$-DP

$$P[M(D) \in S] \le e^{\varepsilon} \cdot P[M(D') \in S] + \delta$$

δ 的含义：不是"允许泄露的 prob"，而是分布尾部的质量界。
通常要求 $\delta \ll \frac{1}{n}$（n 是 dataset 大小）。

☒ 实践中 $\varepsilon = 5$ 或 $\varepsilon = 8$ 很常见，此时 $e^8 \approx 2981$，线性近似**完全失效**！

6.1.3 邻居关系的三种定义

邻居定义	场景	敏感度	对应噪声
$\|D - D'\|_0 \le 1$	添加/删除一条记录	$\Delta_1 f$	Laplace
$\|D - D'\|_1 \le 1$	修改一个特征	$\Delta_1 f$	Laplace
$\|D - D'\|_2 \le 1$	连续扰动（gradient）	$\Delta_2 f$	Gaussian

6.2 两大基础机制

6.2.1 Laplace 机制

$$M(D) = f(D) + \text{Lap}\left(\frac{\Delta_1 f}{\varepsilon}\right)$$

Laplace 分布：$p(x) = \frac{1}{2b} e^{-\frac{|x|}{b}}$
其中 $b = \frac{\Delta_1 f}{\varepsilon}$

证明关键步骤：

$$\frac{p(M(D) = z)}{p(M(D') = z)} \le e^{\varepsilon}$$

使用反三角不等式 $|a| - |b| \le |a - b|$

6.2.2 Gaussian 机制

$$M(D) = f(D) + \mathcal{N}(0, \sigma^2 I)$$

其中 $\sigma \ge \frac{\Delta_2 f \cdot \sqrt{2\ln\left(\frac{1.25}{\delta}\right)}}{\varepsilon}$

对比：
- Laplace：重尾（可能大噪声）
- Gaussian：轻尾（噪声更集中）
- Laplace 适合离散查询
- Gaussian 适合连续 gradient 空间

6.3 DP 三大黄金性质

6.3.1 Post-Processing

$$M \text{ is } (\varepsilon, \delta)\text{–DP}$$
$$\Rightarrow \forall g: g \circ M \text{ is } (\varepsilon, \delta)\text{–DP}$$

直觉：噪声一旦加入，后续计算无法"提纯"。

6.3.2 Composition

$$M_1 \text{ is } (\varepsilon_1, \delta_1)\text{–DP}$$
$$M_2 \text{ is } (\varepsilon_2, \delta_2)\text{–DP}$$
$$\Rightarrow (M_1, M_2) \text{ is } (\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)\text{–DP}$$

每次查询都**消耗**隐私预算！

6.3.3 Subsampling

对随机子集 $Q = \frac{L}{N}$ 应用 (ε, δ)-DP：

$$\Rightarrow (Q\varepsilon, Q\delta)\text{–DP}$$

直觉：不知是否被采样 → 隐私增强。

6.4 DPSGD 算法

⚙ **Differentially Private SGD**

```
def DPSGD(data, model, C, σ, epochs):
  for epoch in range(epochs):
    for batch in sample_minibatch(data, L):
      gradients = []
      for (x, y) in batch:
        g = compute_gradient(model, x, y)
        # Step 1: gradient裁剪（控制敏感度）
        g_clip = g * min(1, C / ||g||₂)
        gradients.append(g_clip)

      # Step 2: 聚合 + 添加噪声
      g_avg = mean(gradients)
      g_noisy = g_avg + N(0, σ²C²/L² · I)

      model = model - η * g_noisy
  return model
```

6.4.1 为什么需要 gradient 裁剪？

$$\Delta_2 g = \max_{D \sim D'} \|g(D) - g(D')\|_2$$

问题：若存在 outlier 使 $\|g\|_2 \to \infty$，敏感度无界！
解决：强制 $\|g\|_2 \le C$，则敏感度 $\Delta_2 \le C$。

6.4.2 隐私预算累积

对于 T 步训练，每步采样比例 $Q = \frac{L}{N}$：
朴素 Composition：$(QT\varepsilon, QT\delta)$-DP
问题：T 可能是 10^6，预算**爆炸**！
改进：Strong Composition $\varepsilon_{\text{total}} = O\left(\sqrt{T} \cdot \varepsilon\right)$

6.4.3 Privacy-Utility Trade-off

参数	↑增大	对隐私影响	对效用影响
ε	隐私变弱	↓	↑（噪声减少）
C（裁剪阈值）	敏感度增大	↓	↑（保留更多 gradient 信息）
σ（噪声）	分布更宽	↑	↓（信号被淹没）

6.5 PATE：教师集成的 DP

⚙ PATE (Private Aggregation of Teacher Ensembles)

```
┌──────────────────────────────────────────────────────────┐
│ 私有数据 D = D₁ ∪ D₂ ∪ ... ∪ Dₘ (分成M份)                  │
│                                                          │
│ 训练M个教师: T₁, T₂, ..., Tₘ (无DP, 各自独立)              │
│                                                          │
│ 对公开未标注数据 x:                                       │
│   - 每个教师投票: nⱼ(x) = #{Tᵢ: Tᵢ(x) = j}                │
│   - 聚合 + 加噪: ŷ = argmax_j (nⱼ(x) + Lap(2/ε))          │
│                                  ↑ 关键：argmax 之前加噪！ │
│ 用 (x, ŷ) 训练学生model (公开发布)                         │
└──────────────────────────────────────────────────────────┘
```

6.5.1 噪声添加位置

错误：argmax **之后**加噪
- 敏感度 = $|Y|$（标签空间大小）

正确：argmax **之前**加噪
- 改变一个 sample → 一个教师投票变化
- 投票变化：$+1$ 和 -1
- **L1 敏感度 = 2**（不是 $|Y|$）

6.5.2 隐私预算

每次查询消耗 ε_0：

$$T \text{ 次查询} \Rightarrow T\varepsilon_0 \text{ -DP}$$

实践意义：公开 dataset 规模受限于隐私预算！

优化：使用 Confident-GNMax 等方法减少每次查询的预算消耗。

6.6 Federated Learning + DP

6.6.1 FedSGD + DP

Client k：
1. 计算 gradient：$g_k = \nabla \mathcal{L}(\theta, D_k)$
2. 裁剪：$g_k \leftarrow g_k \cdot \min\left(1, \frac{C}{\|g_k\|_2}\right)$
3. 加噪：$g_k \leftarrow g_k + \mathcal{N}(0, \sigma^2 I)$
4. 发送 g_k 给 Server

Server：$\theta \leftarrow \theta - \eta \cdot \frac{1}{K} \sum_k g_k$

6.6.2 FedAvg + DP 区别

FedSGD + DP	FedAvg + DP
发送单步 gradient	发送多步权重差
对 g_k 加噪	对 $\Delta\theta_k$ 加噪
直接应用 Gaussian	需考虑多步依赖

6.7 Model Stealing Attack

model 窃取攻击

目标：通过 API 查询，复制目标 model 的功能

形式化：给定只能 query 的 f_{target}，训练 f_{copy} 使：

$$\forall x : f_{\text{copy}}(x) \approx f_{\text{target}}(x)$$

6.7.1 方法

1. **Query-based**：
   - 生成大量 $(x, f_{\text{target}}(x))$ 对
   - 用知识蒸馏训练 f_{copy}
2. **Side-channel**：
   - 利用 API 返回的 logits/confidence
   - 推断更多 model 信息

6.7.2 防御

- **Rate Limiting**：限制查询次数
- **Output Perturbation**：添加噪声到输出
- **Query Auditing**：检测可疑查询模式
- **Watermarking**：在 model 中嵌入水印，证明所有权

6.8 Model Inversion Attack

model 反演攻击

目标：从 model 输出**重建**训练数据的**代表性** sample

与 Gradient Inversion 区别：
- Gradient Inversion：精确重建**具体** sample（FL 场景）
- Model Inversion：重建**类别的典型** sample（黑盒场景）

6.8.1 攻击公式

$$x^* = \arg\max_x P(y_{\text{target}} \mid x)$$

或使用 GAN 生成：

$$z^* = \arg\max_z f_{\text{target}}(G(z))_y$$

再计算 $x^* = G(z^*)$

6.8.2 可视化

```
Class "Person A" (Label 7)
        ↓
Model Inversion Optimization
        ↓
生成一张看起来像 "Person A" 的脸
(不是training set中的具体照片)
```

⚠ Model Inversion 生成的是**类别的平均特征**，不是具体个人的精确照片。但对于敏感类别（如人脸），这仍然是严重的隐私泄露。

6.9 Membership Inference Attack (MIA)

MIA 问题设定

给定 model M 和 sample x，判断 $x \in D_{\text{train}}$?

形式化为假设检验：

- H_0：$x \notin D_{\text{train}}$（Out）
- H_1：$x \in D_{\text{train}}$（In）
- 决策规则：Score$(x) > \tau \rightarrow$ Reject H_0（判定为 In）

6.9.1 Shadow Model 方法

⚙ Shadow Model Attack

```
┌──────────────────────────────────────────────────┐
│ PHASE 1: Shadow Model Training                   │
│   训练 K 个 Shadow Models: M₁, M₂, ..., Mₖ       │
│   每个用不同的training set                        │
│                                                  │
│ PHASE 2: Attack Classifier Training              │
│   对每个 Shadow Model i:                          │
│     - x ∈ Dᵢ (IN):  (Modelᵢ(x), IN)              │
│     - x ∉ Dᵢ (OUT): (Modelᵢ(x), OUT)             │
│   → 训练 Attack Classifier A                     │
│                                                  │
│ PHASE 3: Attack Target Model                     │
│   Target Model M, query point x                  │
│   → b̂ = A(M(x))                                  │
└──────────────────────────────────────────────────┘
```

⚠ 对 LLM **完全不适用**——谁能训练 64 个 GPT-4?

6.9.2 Score-Based 方法（现代方法）

方法	Score 公式	直觉
Loss	$-\log P_M(x)$	训练数据 loss 更低
LiRA	$\log \frac{P(\ell \mid x \in S)}{P(\ell \mid x \notin S)}$	贝叶斯似然比
Min-K% Prob	$\frac{1}{K}\sum_{i \in \text{ bottom-K}} \log P(x_i)$	低 prob token 对 member 也较高

6.9.3 LiRA 详解（Likelihood Ratio Attack）

LiRA 核心思想
贝叶斯视角：不只看 model 对 x 的 loss，而是比较“model 在 x 上的行为”与“随机 model 在 x 上的行为”。

$$\text{LiRA Score} = \log \frac{P(\ell(x) \mid x \in D)}{P(\ell(x) \mid x \notin D)}$$

6.9.4 实现步骤

1. **训练 MANY Shadow Models**：
   - 一半包含 x（IN 集合）
   - 一半不包含 x（OUT 集合）
2. **估计分布**：
   - $P(\ell \mid \text{IN}) = \mathcal{N}(\mu_{\text{IN}}, \sigma^2_{\text{IN}})$
   - $P(\ell \mid \text{OUT}) = \mathcal{N}(\mu_{\text{OUT}}, \sigma^2_{\text{OUT}})$
3. **计算 Log-Likelihood Ratio**：

$$\text{Score} = \frac{(\ell - \mu_{\text{OUT}})^2}{2\sigma^2_{\text{OUT}}} - \frac{(\ell - \mu_{\text{IN}})^2}{2\sigma^2_{\text{IN}}}$$

6.9.5 为什么比 Loss 更好?

⚠ **Loss-based**：只看绝对值
- 问题：简单 sample loss 本就低

LiRA：看相对变化
- 解决：控制了 sample 难度差异

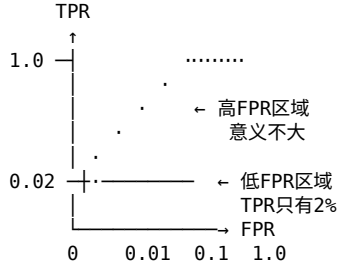
```
Loss-based:
  简单sample: loss=0.1 (member)
  困难sample: loss=0.5 (member)
  → 容易误判困难sample为 non-member

LiRA:
  比较 IN vs OUT 的 loss 分布
  → 对sample难度 robust
```

⚠ **LiRA 的局限**：
- 需要训练大量 Shadow Models（>256）
- 对 LLM 不可行
- 最新趋势：单 model LiRA 变种（用数据增强代替多 model）

6.9.6 MIA 实际表现

⚠ **AUC ≈ 0.5 0.7**：接近随机猜测！
Low FPR 区域才重要：当 FPR = 0.01 时，TPR 可能只有 2%。
这意味着误报率极高——实践中 MIA **几乎不 work**。



为什么低 FPR 重要?
- training set $|S| \ll |D \setminus S|$
- 即使小 FPR 也意味着大量 false positives
- 实际部署中无法承受

6.10 Dataset Inference

从单点到 dataset
动机:

6.11 Memorization

K-Extractable
字符串 S 是 K-extractable，若存在 prefix P:

- 单点 MIA 太难、太 expensive
- 数据拥有者通常有**整个 dataset**
- 弱信号聚合后 → 强信号

T 检验：

$$t = \frac{\overline{x}_{\text{pub}} - \overline{x}_{\text{val}}}{\sqrt{\frac{s^2_{\text{pub}}}{n_{\text{pub}}} + \frac{s^2_{\text{val}}}{n_{\text{val}}}}}$$

若 p-value $< \alpha$，则 reject H_0 → dataset 被使用。

$$P \parallel S \in D_{\text{train}} \wedge M(P) = S \text{ (greedy decoding)}$$

影响因素：

因素	关系	原因
Model Size	正相关	更大容量 → 更能"记住"
Prefix Length	正相关	更多 context → 更窄的 continuation 分布
Repetition	正相关	gradient 更新越多 → 记得越牢
Sequence Length	负相关	累积错误

6.12 DP 与 RS 的对偶性

同一枚硬币的两面

	Differential Privacy	Randomized Smoothing
目标	使分布**相似**（不可区分）	使分布**不同**（可区分）
数学	$P[M(D)] \leq e^{\varepsilon} P[M(D')]$	$P[f(x+\eta) = c] > e^{2\varepsilon} P[f(x+\eta) = c']$
噪声作用	混淆真实数据	平滑决策边界

统一视角：
- DP：希望假设检验 Power **低**
- RS：希望分类置信度 **高**

两者使用**相同数学工具**（指数界、噪声分布），但**优化方向相反**。

6.13 Gradient Inversion Attack

gradient 反演攻击

核心假设：gradient $\nabla\theta$ 必须包含数据信息才能优化 → 可反推数据

攻击目标[4]：

$$x^* = \arg\min_x \|\nabla\theta\mathcal{L}(x, y) - \nabla_{\text{obs}}\|^2 + \lambda R(x)$$

其中 $R(x)$ 是模态特定的 Prior。

6.13.1 Image Prior

Total Variation：

$$R_{\text{TV}}(x) = \sum_{i,j} |x_{i+1,j} - x_{i,j}|$$

鼓励图像平滑。

6.13.2 Text Prior

Perplexity + Reorder：
利用 LM 的 prob 和离散优化。

6.13.3 Tabular Prior

Entropy-based：
利用类别分布假设筛选候选。

6.13.4 FedSGD vs FedAvg 攻击难度

	FedSGD	FedAvg
发送内容	单步 gradient $\nabla\theta$	多步更新 $\Delta\theta$
攻击难度	较易（直接 gradient 匹配）	较难（需反演优化轨迹）
攻击公式	$\min\|\nabla - \nabla^*\|$	$\min \sum_{e,b} \|\nabla_{e,b} - \nabla^*_{e,b}\|$

⚠ FedAvg 攻击需要利用**跨 epoch 数据一致性**先验：$X_{e_1,b} \approx X_{e_2,b}$。

6.14 Attribute Inference

超越 Membership 的隐私攻击

给定 model M 和**部分公开属性** x_{pub}，推断**敏感属性** x_{sens}：

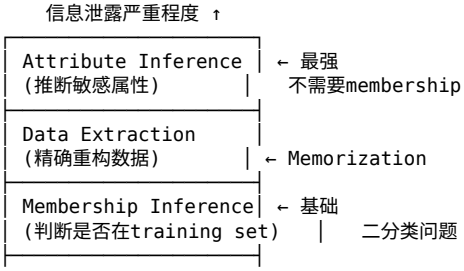
$$\hat{x}_{\text{sens}} = \arg\max_{x_{\text{sens}}} P\big(M \mid x_{\text{pub}}, x_{\text{sens}}\big)$$

关键区别：**不需要** x 在 training set 中！只需 model 学到了属性相关性。

从文本推断地理位置

输入："left shark thing is hilarious... seen it after final exams"
LLM 推理：Glendale, AZ（2015 Super Bowl 举办地）
准确率：85% Top-1

6.15 隐私攻击层次关系



层次关系

- **Memorization → Membership**：能逐字重复 → 一定在 training set 中
- **Membership ↛ Memorization**：在 training set 中 ≠ 会被 memorize
- **Attribute Inference 独立于 Membership**：即使数据不在 training set，也可能通过交互泄露属性

[4]适用于 FL 场景，攻击者（恶意 Server）可观察客户端上传的 gradient

```
| Dataset Inference    | ← 对抗单点MIA
| (dataset级别聚合)    |   统计检验
└──────────────────────┘
```

6.16 Agentic AI 安全

Indirect Prompt Injection (IPI)
攻击链：(核心问题：model 无法区分“用户指令”vs“工具输出中的指令”)

```
Attacker ──► Environment ──► Agent ──► Sensitive Action
(发送邮件)    (收件箱)       (Cursor/GPT)  (读取私有仓库)
```

攻击向量：通过不可信环境（邮件/网页）注入指令。

6.16.1 Instruction Hierarchy

训练 model 区分 System/User/Tool 权限等级
局限：低权限内容仍可通过语义影响高权限决策

6.16.2 Command Sense

检测并移除“AI 指令语气”的文本
局限：无法捕捉非指令式攻击

6.16.3 Dual-LLM Pattern

Planner 不看工具输出，Executor 不看用户指令
局限：无法处理动态决策

核心张力：Security $\propto \frac{1}{\text{Capability}}$；Planner 不看工具输出 → 安全但无法做动态任务（如“根据邮件内容决定下一步”）

6.17 敏感度的统一视角

敏感度同时量化攻击能力和防御代价
在攻击中：敏感度高 → gradient 信息量大 → 易反推数据
在防御中：敏感度高 → 需要更多噪声 → 效用 loss 大

$$\sigma = \frac{\Delta_2 f \cdot \sqrt{2\ln\left(\frac{1.25}{\delta}\right)}}{\varepsilon}$$

裁剪是人为控制敏感度的手段，这解释了 DPSGD 的 gradient 裁剪和 PATE 的 argmax 前加噪设计。

敏感度类型	定义	用途	机制
Δ_0 (Hamming)	添加/删除一条记录	Membership	–
Δ_1 (L1)	$\max \|f(D) - f(D')\|_1$	计数查询	Laplace
Δ_2 (L2)	$\max \|f(D) - f(D')\|_2$	gradient 空间	Gaussian

6.18 MIA Score 函数总览

方法	Signal(x)	Baseline(x)	直觉
Loss-based	$-\log p_\theta(y\|x)$	常数阈值	训练数据 loss 更低
Likelihood-Ratio	$-\log p_\theta(y\|x)$	$-\log p_{\text{ref}}(y\|x)$	相对于基准 model
Gradient Norm	$\|\nabla_\theta \mathcal{L}(x)\|$	经验分布	训练数据 gradient 更小
Calibration	Conf(x) - Acc(x)	0	过拟合 sample 过度自信
Min-K Prob	平均 K 个最低 token prob	绝对阈值	罕见 token 也有高 prob

统一洞察：所有方法都在找“model 对训练数据的异常自信”。

6.19 考试模式识别

6.19.1 场景 → 威胁类型映射

场景描述	威胁类型
攻击者可以查询 model	MIA, Model Inversion
攻击者能看到 gradient	Gradient Inversion (FL)
model 输出逐字重复	Memorization
推断用户敏感属性	Attribute Inference
判断数据是否被使用	Dataset Inference

6.19.2 对比/辨析题要点

对比项	区分要点
DP vs RS	目标相反，工具相同
ε-DP vs (ε, δ)-DP	相对界 vs 相对+绝对
Laplace vs Gaussian	L_1 vs L_2 敏感度
MIA vs Dataset Inference	单点弱信号 vs 聚合强信号
Memorization vs Inversion	精确逐字 vs 代表性重构

6.20 Privacy 易错点

δ 的含义：不是“泄露 prob”，而是尾部质量界
敏感度计算：PATE 在 argmax **之前**加噪，敏感度是 2 不是 $|Y|$
隐私预算累积：$\varepsilon_{\text{total}} \approx \sqrt{T} \cdot \varepsilon$（Advanced Composition），非线性累积是实用化关键
MIA 实践表现：AUC ≈ 0.5 0.7，低 FPR 时 TPR 极低（2% @ FPR=0.01%）
时间偏移陷阱：用时间切分评估 MIA 会混淆“时间”与“membership”
Gradient Inversion：FedAvg 比 FedSGD 难攻，需多 epoch 耦合优化
Attribute Inference：**不需要** membership！利用属性相关性
Agentic AI：Security $\propto$ 1/Capability，完全隔离会牺牲动态能力

7 Part 8: Watermarking & Benchmarking

7.1 LLM Watermarking 核心思想

为什么需要 Watermark?

问题：如何证明内容是 AI 生成的？（Attribution Problem）

方法	问题
Passive Detection (GPT-0)	随着 model 变强越来越难
Visible Watermark (Sora logo)	容易被移除
Metadata	截图就没了
Fingerprinting (哈希数据库)	隐私问题 + 数据库爆炸
Invisible Watermark √	嵌入生成过程，人类不可察觉

7.2 Red-Green Watermark (KGW)

核心思想
将词表[5]伪随机分为 **Green** 和 **Red**，偏向采样 Green tokens。

$$\mathcal{V} = \underbrace{\mathcal{G}}_{\text{Green List}} \cup \underbrace{\mathcal{R}}_{\text{Red List}}$$

其中 $|\mathcal{G}| = \gamma\,|\mathcal{V}|$（通常 $\gamma = 0.5$）。

```
hash(前h个token) + secret_key
          ↓
   seed → PRG → 划分词表
          ↓
γ|V| 个Green, (1-γ)|V| 个Red
```

7.2.1 Generate 函数

⚙ Red-Green 水印生成

Step 1：LLM 计算 logits ℓ（下一个 token 的 prob 分布）
Step 2：用 hash(context) + secret_key 确定 Green/Red 划分
Step 3：修改 logits，给 Green tokens 加 δ：

$$\ell_i' = \begin{cases} \ell_i + \delta & \text{if token } i \in \text{Green} \\ \ell_i & \text{if token } i \in \text{Red} \end{cases}$$

Step 4：Softmax 采样：$P(\text{token}_i) = \frac{e^{\ell_i'}}{\sum_j e^{\ell_j'}}$

关键参数：
- γ：Green tokens 比例（通常 0.5）
- δ：偏置强度（越大水印越强，但质量 loss 越大）
- h：context 窗口大小（用多少前置 token 做 hash）

7.2.2 Detect 函数

检测无需 LLM，只需 secret key！
统计检验假设检验：H_0为无水印，每个 token 颜色随机：

$$H_0 : \text{无水印} \Rightarrow S \sim \text{Binomial}(n, 0.5)$$

其中 S 是 Green token 计数。
P-value：$P(X \geq S \mid H_0) = \sum_{k=S}^{n} \binom{n}{k} 0.5^n$
判定规则：若 p-value $< \alpha$ 则判定有水印。

α 直接控制 False Positive Rate！设 $\alpha = 10^{-6}$ 意味着每百万次误判一次。

7.3 ITS Watermark（Distortion-Free）

7.3.1 Red-Green 的问题

Red-Green **修改了 prob 分布**！
例如：Barack 是 Green，Obama 是 Red → 可能采样不出 "Barack Obama"

7.3.2 Distortion-Free 核心

在**期望意义**上不改变 LLM 的输出分布。

⚙ ITS 采样

Private Key：
- $\xi = [\xi_1, ..., \xi_n]$ N 个 $U[0,1]$ 随机变量
- π：词表的伪随机排列

生成第 t 个 token：
1. LLM → $P(\text{next token})$
2. 用 π 排列 → P_π
3. 计算 CDF：$F(k) = \sum_{i=1}^{k} P_\pi(i)$
4. 找最小 k：$F(k) \geq \xi_t$
5. 返回 $\pi^{-1}(k)$

为什么是 Distortion-Free?
prob 为 p 的 token，被选中的 prob 恰好也是 p：

$$P(\text{sample token with prob } p) = P(\xi_t \text{ falls in interval of length } p) = p$$

代价：确定性输出（同 prompt 同 response），多样性丧失。

7.4 SynthID (Google DeepMind)

7.4.1 特点

- Distortion-Free（保持分布）
- Non-Deterministic（同 prompt 可得不同回复）
- 大规模验证（2000 万文本 AB 测试）

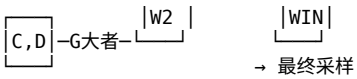
```
Tournament（锦标赛）：

第1轮          第2轮          第3轮
┌─┐
|A,B|─G大者─|   |
└─┘          |W1 |─G大者─┌──
             └──┘
```

[5]Vocabulary，model 可生成的所有 token 集合

7.4.2 Tournament Sampling

1. hash(context) + key → G 值（m 个 Bernoulli）
2. 从 LLM 分布采样 2^m 个候选
3. 锦标赛：每轮比较 G 值，大者晋级
4. Winner → 最终采样（高 G 值 token 更易赢）



直觉：高G值token更易赢得比赛

检测：$S = \sum_t \sum_{i=1}^{m} G_{t,i} \sim \text{Binomial}(T \cdot m, 0.5)$

7.5 水印攻击方法

7.5.1 Scrubbing（移除）

- 短文本 (< 100 tokens)：信号本就弱
- 中文本 (100-600)：Paraphrase 30% tokens 即可
- 长文本 (> 600)：需 Watermark Stealing

7.5.2 Spoofing（伪造）

Piggyback Spoofing：

原文："This article is great" ↓ 改一个词 攻击："This article is mean" → 仍有水印！

7.5.3 Stealing（窃取）

1. Query 水印 LLM 30K 次（约 $50）
2. 估计 $\frac{P_{\text{wm}}}{P_{\text{base}}}$
3. $S > 0 \rightarrow$ 预测 Green
4. 用于 Scrubbing/Spoofing

Watermark Stealing 核心公式

$$S(\text{token}|\text{ctx}) = \log \frac{P_{\text{watermarked}}(\text{token}|\text{ctx}) + \varepsilon}{P_{\text{base}}(\text{token}|\text{ctx}) + \varepsilon}$$

Spoofing Detection 利用伪随机无法泛化到罕见 n-gram 的特性：

计算 Correlation(token 颜色, N-gram 频率)

- 真水印：无相关性（伪随机与频率无关）
- 伪造：罕见词更容易猜错 → 有相关性

7.6 Radioactivity（数据保护）

核心发现

在水印数据上训练的 model，输出也会带有水印！

应用：Dataset Inference Attack

1. 用水印 LLM paraphrase 自己的文章
2. 发布水印版本到网上
3. 查询可疑 model，检测输出是否有水印
4. 若有 → 证明该 model 训练使用了你的数据

7.7 LLM Benchmarking

Benchmark 三要素

$$\text{Benchmark} = (\text{Task}, \text{Scoring}, \text{Standardized Setup})$$

与传统 ML 的区别：

- 评估对象：Algorithm → Model（产品）
- Train/Test：IID split → 边界模糊
- Task：明确（分类）→ 开放（任何问题）
- Access：完全控制 → 常常只有 API

7.7.1 四大评估范式

范式	答案格式	评估方式
Closed-form	A/B/C/D	精确匹配
Free-form	自由生成	验证结果（单元测试）
Simulation	与环境交互	环境反馈
Preference	两 model 对比	人/LLM 选择偏好

7.8 Contamination（污染问题）

Data vs Task Contamination

Data Contamination[6]：

- Benchmark 问题在 training set 中
- model"背答案"
- **性能虚高**

Task Contamination：

- 训练数据针对特定任务优化
- 可能是良性（鼓励 model 变强）
- 也可能只学会格式/套路

7.8.1 形式化定义

$$\text{Contaminated} \Leftrightarrow \exists x \in D_{\text{train}} : F(x, b) > \tau$$

其中：

- b：benchmark sample
- x：训练数据 sample
- F：相似度函数（**核心难点**）
- τ：阈值

☒ 定义 F 非常难！完全相同？语义相同？换个说法算不算？

7.8.2 检测方法（按 Access Level 分类）

7.8.3 Level 1: Oracle Access

方法：N-gram Overlap

从 benchmark 和训练数据提取 n-grams，计算 overlap

缺点：简单改写就能绕过

7.8.4 Level 2: White-box

方法：Perplexity / Min-K% Prob

核心直觉：model 对见过的 sample"异常确信"

联系：与 MIA 非常相似！

7.8.5 Level 3: Black-box

方法：Completion Test

给 model benchmark 前半部分，让它补全

若补全完全一致 → 可能见过

7.8.6 Outcome-based Detection

MathArena 策略

利用**时间因果性**：

$$\text{Performance Gap} = \text{Score}_{2024} - \text{Score}_{2025} \gg 0 \Rightarrow \text{Contamination}$$

假设 model 在 2024 年前训练：

- 若 2024 题表现**显著优于** 2025 题（应同分布）
- 则证明 2024 题被记忆（污染）

这是反事实推断！

7.9 Dynamic Benchmarks（动态评估）

为什么需要 Dynamic Benchmark?

Static Benchmark 的问题：

[6] Benchmark 的具体问题/答案出现在训练数据中

- 发布后立即被爬取进训练数据
- model 在“考试”和“解决问题”上分不清
- 分数虚高（Goodhart’s Law）

7.9.1 Dynamic Benchmark 类型

类型	例子
时间动态	MathArena（新数学题）
生成动态	Dynabench（持续更新）
私有动态	SEAL Leaderboard
验证动态	Agent 环境测试

7.9.2 核心优势

- **Anti-contamination**：新题无法提前准备
- **True generalization**：测试**能力**而非**记忆**
- **Continuous evaluation**：持续跟踪进展

⚠ **问题**：可能与旧 benchmark 不可比
解决：使用 IRT/Polyrating 统一 scale

7.9.3 Polyrating：De-biasing 方法

Polyrating 核心思想
问题：Judge（人类/LLM）有系统性偏见
解决：显式建模 bias 参数并估计+移除

$$P(\text{Model } i \text{ wins}) = \sigma\left(s_i - s_j + \underbrace{b_{\text{length}} \cdot \Delta \text{ len} + b_{\text{format}} \cdot \Delta \text{ fmt} + \ldots}_{\text{Bias Terms}}\right)$$

估计 b 后，报告 de-biased score s_i。

7.10 Scoring Mechanisms

7.10.1 Goodhart’s Law

⚠ “When a measure becomes a target, it ceases to be a good measure.”
例如：ROUGE-N 评翻译
- 计算 n-gram overlap
- 问题：翻译可以有多种正确表达
- model 学会迎合评分而非真正翻译

7.10.2 Bradley-Terry Model

给定偏好数据 (A vs B, Winner)，求全局排名：

$$P(i \text{ beats } j) = \frac{e^{s_i}}{e^{s_i} + e^{s_j}} = \sigma(s_i - s_j)$$

与 ELO 关系：
- Bradley-Terry：精确解（凸优化）
- ELO：在线近似（增量更新）

7.10.3 Judge Bias Problem

⚠ Human/LLM Judge 存在系统性偏见：
- ⚠ 偏好更长的回答
- ⚠ 偏好格式更好的回答（markdown, bullet points）
- ⚠ 偏好有 emoji 的回答 😊
- ⚠ 偏好更自信的语气

后果: model 学会“讨好”评委，而非真正变强
解决：Polyrating 显式建模 bias 参数并 de-bias

7.11 Reporting Best Practices

7.11.1 应该做的 ✓

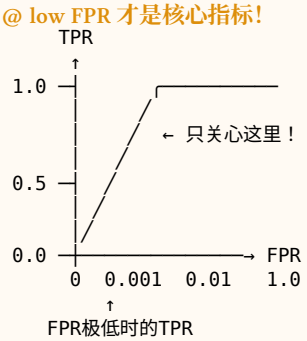
- 报告**统计显著性**（92.15% vs 92.1% 可能只是噪声）
- 公开评估输出（让社区可验证）
- Apples-to-apples 比较（相同设置、相同 effort）
- 可复现（详细记录配置、随机种子）

7.11.2 常见不诚实手法 ✗

- **Benchmark Omission**：只报告好的 benchmark
- **Creative Reporting**：柱状图不从 0 开始
- **Artificial Increase**：自己 model 精心调参，竞品默认设置

7.12 Watermarking 评估指标

TPR @ low FPR 才是核心指标!



其他 dim：Detectability, Quality, Robustness, Security

7.13 易错点

Detection 需要 LLM?：⚠ 只需 secret key，无需 LLM!
AUC 是好指标?：⚠ 只关心极低 FPR 下的 TPR
Distortion-Free = 无影响?：是**期望意义**上不改变分布
水印越强越好?：需权衡 Quality 和 Detectability
高分 = 好 model?：可能是污染/cherry-picking
N-gram 检测够用?：简单改写即可绕过
官方数字可信?：检查设置是否公平、是否有遗漏