Mathematical Foundations of Trustworthy AI

Aizierjiang Aiersilan

▶ **To cite this version:**

Aizierjiang Aiersilan. Mathematical Foundations of Trustworthy AI. Doctoral. Studies in Trustworthy AI, Online, Macau SAR China. 2026, pp.142. ⟨hal-05449393v2⟩

HAL Id: hal-05449393

https://hal.science/hal-05449393v2

Submitted on 11 Mar 2026

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons CC BY 4.0 - Attribution - International License

Mathematical Foundations of Trustworthy AI

Theory, Algorithms, and Engineering Principles

Lecture Notes

Aizierjiang Aiersilan

Version 1.1 — March 10, 2026

Abstract

When I was learning about Trustworthy AI, I found there were limited systematic books on its mathematical foundations, so I put together my notes taken during the early phase of my self-learning. In these notes, I provide a rigorous mathematical examination of Trustworthy AI, exploring reliable machine learning through the lens of constrained optimization, statistical learning theory, and differential geometry. I bring together concepts to define the multi-dimensional requirements of AI systems, moving beyond heuristic approaches toward a grounded engineering discipline. The notes formalize critical properties such as adversarial robustness via Lipschitz continuity and dual norms, algorithmic fairness via group constraints and impossibility theorems, and causal inference via structural causal models. Through these, I aim to provide a useful systematic mathematical foundation for better understanding Trustworthy AI.

Keywords: Trustworthy AI, Adversarial Robustness, Algorithmic Fairness, Differential Privacy, Explainability, Causal Inference, Safe Reinforcement Learning

Contents

1 Notation and Conventions

Throughout this document, we use consistent mathematical notation. Let $\mathcal{X} \subseteq \mathbb{R}^d$ denote the input space and $\mathcal{Y}$ the output space. A dataset is $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$. We use $\mathbb{E}$ for expectation, Var for variance, and $P(\cdot)$ for probability.

Probability Notation: $P(X)$ or $\mathcal{P}$ denotes a probability distribution. We write $X \perp Y|Z$ for conditional independence.

Norms: $\|x\|_p$ is the L_p norm: $\|x\|_2 = \sqrt{\sum_i x_i^2}$ (Euclidean), $\|x\|_\infty = \max_i |x_i|$ (max). For matrices, $\|W\|_{op}$ is the spectral norm.

Indicator Function: $\mathbb{I}(\text{event}) = 1$ if true, 0 otherwise.

Operators: $\arg\min_x f(x)$ and $\arg\max_x f(x)$ denote the minimizing and maximizing arguments. ∇f is the gradient.

Causal Notation: Interventions are denoted by the do-operator $\text{do}(X = x)$, setting variable X to value x. Counterfactuals are written as $Y_{x'}$, the value of Y under intervention $\text{do}(X = x')$. Graph manipulations include $G_{\bar{X}}$ (arrows into X removed) and $G_{\underline{X}}$ (arrows out of X removed).

Subscripts and Superscripts: Common conventions include $_t$ for time step or round, $_k$ for client k in federated learning, * for optimal values, $'$ for baseline or perturbed versions, and $\hat{\cdot}$ for empirical estimates.

Federated Learning: Client-specific quantities are denoted with subscripts, e.g., p_k for client weight, F_k for local loss, S_t for selected clients at round t.

Safety and Robustness: Barrier functions are denoted $h(s)$, Lyapunov functions $V(s)$, and uncertainty sets $\mathcal{U}$.

General Conventions: Bold symbols like $\mathbf{w}$ denote vectors or matrices. Calligraphic letters like $\mathcal{X}$ denote sets or spaces. Greek letters are used for parameters (e.g., ϵ for perturbation budgets, δ for failure probabilities).

2 Introduction: The Necessity of Trust

[2, 9, 67]

The rapid and pervasive integration of Artificial Intelligence (AI) into the fabric of modern society has fundamentally altered the landscape of decision-making in critical sectors such as healthcare, epidemiological modeling, medical imaging, and mathematical theorem proving. While the empirical performance of these systems, measured by aggregate metrics like accuracy or F1-score on static test sets, has reached superhuman levels, this superficial success often masks deep-seated vulnerabilities. The deployment of AI in high-stakes environments has precipitated a crisis of trust, driven by high-profile failures where systems have exhibited brittleness in the face of adversarial manipulation, exhibited unintended disparities in data sub-populations, leaked sensitive personal information, or operated as inscrutable black boxes whose reasoning cannot be audited. Trustworthy AI, therefore, is not merely a collection of ethical guidelines or policy desiderata; it is a rigorous technical discipline that demands a foundational reimagining of how we design, train, and verify machine learning systems. It requires moving beyond the paradigm of average-case performance maximization to a regime of worst-case robustness, causal validity, and provable constraints.

We posit that the nebulous concept of "trust" can be decomposed into a set of mathematically precise, verifiable properties. Rather than relying on intuition, we define these properties through the lens of constrained optimization, statistical learning theory, and differential geometry. **Robustness** is formalized as the invariance of the model's output under bounded perturbations of the input, requiring Lipschitz continuity or specific geometric margins. **Fairness** is articulated as statistical independence between model predictions and sensitive attributes, or as the equality of error rates across demographic groups, often invoking impossibility theorems that delineate the trade-offs between different ethical ideals. **Privacy** is rigorously defined via Differential Privacy, which quantifies the stability of the algorithm's output distribution with respect to the presence or absence of any single individual's data. **Explainability** seeks to provide faithful, interpretable surrogates or attribution maps that elucidate the decision boundary, while **Causality** ensures that the model captures the structural equations governing the data generation process, rather than merely exploiting spurious correlations. This document may also serve as a comprehensive mathematical and technical reference for Trustworthy AI.

Table 1: Dimensions of Trustworthy AI and their Mathematical Formalizations

Dimension	**Core Concept**	**Mathematical Formalism**
Robustness	Stability under perturbation	Lipschitz continuity, Dual norms, Min-max optimization
Fairness	Independence from sensitive traits	Statistical Parity ($\hat{Y} \perp A$), Equalized Odds ($\hat{Y} \perp A \mid Y$)
Privacy	Indistinguishability of datasets	Differential Privacy (ϵ, δ-DP), Renyi Divergence
Explainability	Interpretability of decisions	Shapley values, Local linear approximations, Saliency maps
Causality	Structural validity	Structural Causal Models (SCM), Do-calculus, Intervention

Note: In this table, $\hat{Y}$ denotes the model prediction, A represents the sensitive attribute, Y is the true label, $\perp$ indicates statistical independence, and ϵ, δ are privacy budget parameters.

3 Statistical Learning Theory and Generalization

[52, 56, 73]

This section establishes the mathematical foundations of statistical learning theory [49, 62], which provides the theoretical framework for understanding when and why machine learning algorithms work. We develop the core concepts rigorously, with complete proofs and detailed explanations that connect the mathematical formalism to practical implications for Trustworthy AI.

3.1 The Learning Problem Formulation

3.1.1 Mathematical Setup

The fundamental goal of supervised machine learning is to construct a predictive mapping $f : \mathcal{X} \to \mathcal{Y}$ that generalizes well to unseen data. Let $\mathcal{X} \subseteq \mathbb{R}^d$ denote the input manifold and $\mathcal{Y}$ the output space. We assume the existence of a joint probability distribution $\mathcal{P}$ over $\mathcal{X} \times \mathcal{Y}$. The learner is provided with a training dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where each sample is drawn independently and identically distributed (i.i.d.) from $\mathcal{P}$.

Definition 3.1 (Statistical Learning Problem)**.** A statistical learning problem is specified by a tuple $(\mathcal{X}, \mathcal{Y}, \mathcal{P}, \mathcal{H}, \ell)$ where:

- $\mathcal{X} \subseteq \mathbb{R}^d$ is the **input space** (feature space)
- $\mathcal{Y}$ is the **output space** (label space)
- $\mathcal{P}$ is an unknown probability distribution over $\mathcal{X} \times \mathcal{Y}$
- $\mathcal{H} \subseteq \{f : \mathcal{X} \to \mathcal{Y}\}$ is the **hypothesis class**
- $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is the **loss function**

The assumption of i.i.d. sampling is central to classical learning theory, allowing us to apply the Law of Large Numbers and concentration inequalities to bound the difference between empirical observations and true population statistics. However, in the context of Trustworthy AI, this assumption is often violated; adversaries may shift the distribution, or the deployment environment may differ from the training environment (distribution shift).

3.1.2 Common Output Spaces and Loss Functions

For **binary classification**, $\mathcal{Y} = \{-1, +1\}$ or $\{0, 1\}$. Common loss functions include:

- **0-1 Loss**: $\ell_{0-1}(\hat{y}, y) = \mathbb{I}(\hat{y} \neq y)$
- **Hinge Loss**: $\ell_{\text{hinge}}(\hat{y}, y) = \max(0, 1 - y\hat{y})$ (used in SVMs)
- **Logistic Loss**: $\ell_{\log}(\hat{y}, y) = \log(1 + e^{-y\hat{y}})$ (used in logistic regression)
- **Exponential Loss**: $\ell_{\exp}(\hat{y}, y) = e^{-y\hat{y}}$ (used in AdaBoost)

For **multi-class classification**, $\mathcal{Y} = \{1, 2, \ldots, K\}$. The cross-entropy loss is:

$$\ell_{CE}(\hat{p}, y) = -\sum_{k=1}^{K} \mathbb{I}(y = k) \log \hat{p}_k = -\log \hat{p}_y \tag{1}$$

where $\hat{p} \in \Delta^{K-1}$ is a probability distribution over classes, $\hat{p}_k$ is the predicted probability for class k, $y \in \{1, \ldots, K\}$ is the true class label, and $\mathbb{I}(\cdot)$ is the indicator function which is 1 if the argument is true and 0 otherwise.

For **regression**, $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = \mathbb{R}^m$. Common loss functions include:

- **Squared Loss**: $\ell_2(\hat{y}, y) = (y - \hat{y})^2$
- **Absolute Loss**: $\ell_1(\hat{y}, y) = |y - \hat{y}|$
- **Huber Loss**: $\ell_\delta(\hat{y}, y) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & |y - \hat{y}| \leq \delta \\ \delta(|y - \hat{y}| - \frac{\delta}{2}) & \text{otherwise} \end{cases}$ where $\delta > 0$ is a threshold parameter controlling the transition from quadratic to linear behavior.
- **Quantile Loss**: $\ell_\tau(\hat{y}, y) = \max(\tau(y - \hat{y}), (\tau - 1)(y - \hat{y}))$ for $\tau \in (0, 1)$, where τ represents the target quantile (e.g., $\tau = 0.5$ for median regression).

3.1.3 Risk and Empirical Risk

The quality of a predictor f is measured by a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$. The true objective is to minimize the *expected risk* (also called *population risk* or *true risk*), which represents the average loss over the entire data distribution:

Definition 3.2 (Expected Risk)**.** The expected risk of a hypothesis $f \in \mathcal{H}$ is:

$$\mathcal{R}(f) \coloneqq \mathbb{E}_{(x,y)\sim\mathcal{P}}[\ell(f(x), y)] = \int_{\mathcal{X}\times\mathcal{Y}} \ell(f(x), y)\, d\mathcal{P}(x, y) \tag{2}$$

where $\mathbb{E}$ denotes the expectation operator with respect to the joint distribution $\mathcal{P}$, and ℓ is the loss function.

Definition 3.3 (Bayes Optimal Predictor)**.** The Bayes optimal predictor f^* minimizes the expected risk over all measurable functions:

$$f^* \coloneqq \arg\min_{f:\mathcal{X}\to\mathcal{Y}} \mathcal{R}(f) \tag{3}$$

where the minimization is taken over the space of all measurable functions from $\mathcal{X}$ to $\mathcal{Y}$. The corresponding minimum risk $\mathcal{R}^* \coloneqq \mathcal{R}(f^*)$ is called the **Bayes risk**.

Proposition 3.1 (Bayes Optimal Predictors)**.** *For common loss functions, the Bayes optimal predictor has closed form:*

1. *For squared loss:* $f^*(x) = \mathbb{E}[Y|X = x]$ *(conditional mean)*
2. *For absolute loss:* $f^*(x) = \text{median}(Y|X = x)$ *(conditional median)*
3. *For 0-1 loss:* $f^*(x) = \arg\max_y P(Y = y|X = x)$ *(MAP estimate)*

Proof. We prove each case separately.

Case 1 (Squared Loss): For squared loss:

$$\mathcal{R}(f) = \mathbb{E}[(Y - f(X))^2] = \mathbb{E}[\mathbb{E}[(Y - f(X))^2|X]] \tag{4}$$

$$= \mathbb{E}[\mathbb{E}[Y^2|X] - 2f(X)\,\mathbb{E}[Y|X] + f(X)^2] \tag{5}$$

Taking the functional derivative with respect to $f(x)$ and setting to zero:

$$\frac{\partial}{\partial f(x)} \mathbb{E}[(Y - f(X))^2 | X = x] = -2\,\mathbb{E}[Y|X = x] + 2f(x) = 0 \tag{6}$$

Thus $f^*(x) = \mathbb{E}[Y|X = x]$.

Case 2 (Absolute Loss): For absolute loss, we minimize:

$$\mathcal{R}(f) = \mathbb{E}[|Y - f(X)|] = \mathbb{E}[\mathbb{E}[|Y - f(X)||X]] \tag{7}$$

For a fixed x, we need to minimize $\mathbb{E}[|Y - c||X = x]$ over $c \in \mathbb{R}$. This is a classical result: the minimizer of $\mathbb{E}[|Z - c|]$ for any random variable Z is the median of Z. To see this, note that the derivative of $g(c) = \mathbb{E}[|Y - c||X = x]$ is:

$$g'(c) = \mathbb{E}[\text{sign}(c - Y)|X = x] = P(Y < c|X = x) - P(Y > c|X = x) \tag{8}$$

This equals zero when $P(Y < c|X = x) = P(Y > c|X = x) = 1/2$, i.e., when c is the conditional median.

Case 3 (0-1 Loss): For 0-1 loss in classification:

$$\mathcal{R}(f) = \mathbb{E}[\mathbb{I}(f(X) \neq Y)] = \mathbb{E}[\mathbb{E}[\mathbb{I}(f(X) \neq Y)|X]] \tag{9}$$

$$= \mathbb{E}[P(f(X) \neq Y|X)] = \mathbb{E}[1 - P(Y = f(X)|X)] \tag{10}$$

For fixed x, we maximize $P(Y = c|X = x)$ over $c \in \mathcal{Y}$. This is achieved by the maximum a posteriori (MAP) estimate:

$$f^*(x) = \arg\max_{y \in \mathcal{Y}} P(Y = y|X = x) \tag{11}$$

□

Since $\mathcal{P}$ is unknown, we cannot compute $\mathcal{R}(f)$ directly. Instead, we resort to the *Empirical Risk Minimization* (ERM) principle, which minimizes the empirical average of the loss on the training set:

Definition 3.4 (Empirical Risk)**.** Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, the empirical risk is:

$$\hat{\mathcal{R}}_{\mathcal{D}}(f) \coloneqq \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \tag{12}$$

where n is the number of training samples and (x_i, y_i) is the i-th input-output pair.

Definition 3.5 (Empirical Risk Minimization)**.** The ERM principle selects the hypothesis that minimizes empirical risk:

$$\hat{f}_{ERM} \coloneqq \arg\min_{f \in \mathcal{H}} \hat{\mathcal{R}}_{\mathcal{D}}(f) \tag{13}$$

The central question of learning theory is: Under what conditions does minimizing the empirical risk $\hat{\mathcal{R}}(f)$ lead to a small expected risk $\mathcal{R}(f)$? This convergence is governed by the complexity of the hypothesis class $\mathcal{H}$ from which f is chosen. If $\mathcal{H}$ is too complex, we may find a function that fits the training data perfectly (memorization) but fails to generalize; if it is too simple, we may fail to capture the underlying patterns (underfitting).

3.1.4 Decomposition of Excess Risk

A key insight is to decompose the excess risk of ERM into interpretable components:

Theorem 3.2 (Excess Risk Decomposition)**.** *Let $f^* = \arg\min_f \mathcal{R}(f)$ be the Bayes optimal predictor, $f^*_{\mathcal{H}} = \arg\min_{f\in\mathcal{H}} \mathcal{R}(f)$ be the best-in-class predictor, and $\hat{f} = \arg\min_{f\in\mathcal{H}} \hat{\mathcal{R}}(f)$ be the ERM solution. Then:*

$$\underbrace{\mathcal{R}(\hat{f}) - \mathcal{R}(f^*)}_{\textit{Excess Risk}} = \underbrace{\mathcal{R}(f^*_{\mathcal{H}}) - \mathcal{R}(f^*)}_{\textit{Approximation Error}} + \underbrace{\mathcal{R}(\hat{f}) - \mathcal{R}(f^*_{\mathcal{H}})}_{\textit{Estimation Error}} \tag{14}$$

where $\mathcal{R}(\cdot)$ denotes the expected risk.

The **approximation error** measures the expressiveness of $\mathcal{H}$: how well the best function in our hypothesis class can approximate the true optimal function. This error decreases as $\mathcal{H}$ becomes richer.

The **estimation error** measures the statistical difficulty of finding the best function in $\mathcal{H}$ from finite samples. This error increases with the complexity of $\mathcal{H}$ and decreases with more data.

This decomposition reveals a fundamental trade-off: richer hypothesis classes reduce approximation error but increase estimation error. Trustworthy AI adds additional constraints to $\mathcal{H}$ (fairness, robustness), which may increase approximation error but provide crucial safety guarantees.

3.2 Bias-Variance Decomposition

To understand the sources of error in a learning algorithm, we can decompose the expected risk into interpretable components. Consider the squared error loss $\ell(y, \hat{y}) = (y - \hat{y})^2$ in a regression setting. Let the true relationship be $y = g(x) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is irreducible noise with zero mean and variance σ^2. Let $\hat{f}_D(x)$ be the predictor learned from a specific dataset D. We are interested in the expected error of the algorithm at a point x, averaged over all possible realizations of the training dataset D.

Theorem 3.3 (Bias-Variance Decomposition)**.** *For squared loss and $y = g(x) + \epsilon$ with $\mathbb{E}[\epsilon] = 0$ and $\mathrm{Var}(\epsilon) = \sigma^2$:*

$$\mathbb{E}_D\, \mathbb{E}_\epsilon[(y - \hat{f}_D(x))^2] = \sigma^2 + (g(x) - \bar{f}(x))^2 + \mathbb{E}_D[(\hat{f}_D(x) - \bar{f}(x))^2] \tag{15}$$

where $\bar{f}(x) = \mathbb{E}_D[\hat{f}_D(x)]$ is the average prediction over all possible training sets D, $\mathbb{E}_D$ denotes expectation over the random choice of training set, and $\mathbb{E}_\epsilon$ denotes expectation over the noise in the test point.

Proof. We begin by expanding the expected squared error:

$$\mathbb{E}_D\, \mathbb{E}_\epsilon[(y - \hat{f}_D(x))^2] = \mathbb{E}_D\, \mathbb{E}_\epsilon[(g(x) + \epsilon - \hat{f}_D(x))^2] \tag{16}$$

Expanding the square and using linearity of expectation:

$$= \mathbb{E}_D\, \mathbb{E}_\epsilon[(g(x) - \hat{f}_D(x))^2 + \epsilon^2 + 2\epsilon(g(x) - \hat{f}_D(x))] \tag{17}$$

Since ϵ is independent of D and has mean zero, the cross-term vanishes ($\mathbb{E}_\epsilon[\epsilon] = 0$). Also, $\mathbb{E}_\epsilon[\epsilon^2] = \sigma^2$:

$$= \mathbb{E}_D[(g(x) - \hat{f}_D(x))^2] + \sigma^2 \tag{18}$$

Now we analyze the term $\mathbb{E}_D[(g(x) - \hat{f}_D(x))^2]$. Let $\bar{f}(x) = \mathbb{E}_D[\hat{f}_D(x)]$ be the average prediction. We add and subtract $\bar{f}(x)$:

$$\mathbb{E}_D[(g(x) - \hat{f}_D(x))^2] = \mathbb{E}_D[(g(x) - \bar{f}(x) + \bar{f}(x) - \hat{f}_D(x))^2] \tag{19}$$

Let $a = g(x) - \bar{f}(x)$ (a constant with respect to D) and $b = \bar{f}(x) - \hat{f}_D(x)$. Then:

$$\mathbb{E}_D[(a+b)^2] = a^2 + \mathbb{E}_D[b^2] + 2a\,\mathbb{E}_D[b] \tag{20}$$

$$= (g(x) - \bar{f}(x))^2 + \mathbb{E}_D[(\hat{f}_D(x) - \bar{f}(x))^2] + 2(g(x) - \bar{f}(x))\,\mathbb{E}_D[\bar{f}(x) - \hat{f}_D(x)] \tag{21}$$

The cross term is zero because $\mathbb{E}_D[\hat{f}_D(x)] = \bar{f}(x)$ by definition, so $\mathbb{E}_D[\bar{f}(x) - \hat{f}_D(x)] = 0$.

Thus, we arrive at the classic decomposition:

$$\boxed{\text{Error} = \underbrace{\sigma^2}_{\text{Irreducible}} + \underbrace{(g(x) - \bar{f}(x))^2}_{\text{Bias}^2} + \underbrace{\mathbb{E}_D[(\hat{f}_D(x) - \bar{f}(x))^2]}_{\text{Variance}}} \tag{22}$$

□

Remark 3.1 (Interpretation of Terms)**.**

- The **Bias** term measures the systematic error introduced by the simplifying assumptions of the model class (e.g., approximating a non-linear function with a linear model). It represents the error incurred even with infinite data, due to fundamental limitations of the hypothesis space.
- The **Variance** term measures the sensitivity of the model to the specific realization of the training data, quantifying how much predictions fluctuate across different datasets of the same size.
- The **Irreducible Error** (σ^2) represents noise inherent in the data-generating process that no model can eliminate.

In the context of Trustworthy AI, we often introduce constraints (such as fairness or robustness) that restrict the hypothesis space $\mathcal{H}$. This inevitably increases the bias of the model (as we may exclude the "true" unconstrained minimizer) but can potentially reduce variance or ensure safety properties that are critical for deployment. Understanding this trade-off is essential for designing trustworthy systems where perfect accuracy is sacrificed for guaranteed safety or fairness.

3.3 Uniform Convergence and VC Dimension

To guarantee generalization, we require that the empirical risk converges to the true risk uniformly over the hypothesis class $\mathcal{H}$. If the convergence is not uniform, the learning algorithm might select a "bad" hypothesis that happens to perform well on the training set due to random chance.

Definition 3.6 (Uniform Convergence)**.** A hypothesis class $\mathcal{H}$ has the uniform convergence property if for every $\epsilon > 0$ and $\delta > 0$, there exists a sample size $n_0(\epsilon, \delta)$ such that for all $n \geq n_0(\epsilon, \delta)$:

$$P\left(\sup_{f \in \mathcal{H}} |\mathcal{R}(f) - \hat{\mathcal{R}}(f)| > \epsilon\right) < \delta \tag{23}$$

where P denotes the probability over the random draw of the training set of size n.

The Vapnik-Chervonenkis (VC) dimension provides a combinatorial measure of the complexity of $\mathcal{H}$ for binary classification. It is defined as the size of the largest set of points that can be "shattered" by $\mathcal{H}$, meaning that the hypothesis class can realize all possible labelings of those points.

Example 3.1 (VC Dimension of Linear Classifiers in $\mathbb{R}^2$)**.** Consider $\mathcal{H}$ as the set of all linear classifiers (lines) in $\mathbb{R}^2$. Given three non-collinear points, say $(0,0)$, $(1,0)$, and $(0,1)$, we can find a line that separates any of the $2^3 = 8$ possible labelings (e.g., to label only $(0,0)$ as $+1$, we draw a line close to the origin separating it from the other two). However, for four points in general position, there always exists a labeling that no line can separate, specifically, the "XOR" pattern where diagonally opposite points share the same label. Therefore, $d_{VC}(\text{linear classifiers in } \mathbb{R}^2) = 3$. In general, $d_{VC}(\text{linear classifiers in } \mathbb{R}^d) = d + 1$. For instance, in a medical imaging application where each image is represented by $d = 100$ features (e.g., pixel intensities or extracted measurements), a linear classifier has VC dimension 101, meaning the VC bound requires roughly $n = O(101 \cdot \log n/\epsilon^2)$ training images to guarantee generalization with error at most ϵ.

Theorem 3.4 (Vapnik-Chervonenkis Bound)**.** *For a binary classification problem with 0-1 loss, if $\mathcal{H}$ has finite VC dimension d_{VC}, then with probability at least $1 - \delta$, for any $f \in \mathcal{H}$:*

$$\mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + \sqrt{\frac{8d_{VC}\ln(2n) + 8\ln(4/\delta)}{n}} \tag{24}$$

where n is the number of samples, ln *is the natural logarithm, and the probability is taken over the random draw of the training set.*

Proof Sketch. The key insight is to bound the growth function $\Pi_{\mathcal{H}}(n)$, which counts the maximum number of distinct labelings that $\mathcal{H}$ can produce on any dataset of size n. Sauer's Lemma establishes that if $d_{VC}(\mathcal{H}) = d$, then:

$$\Pi_{\mathcal{H}}(n) \leq \sum_{i=0}^{d} \binom{n}{i} \leq \left(\frac{en}{d}\right)^d \tag{25}$$

This polynomial growth (compared to the exponential 2^n for unrestricted classes) enables uniform convergence. Using a union bound over the $\Pi_{\mathcal{H}}(n)$ possible dichotomies and applying Hoeffding's inequality yields the stated bound. The full proof requires symmetrization arguments and is found in [70]. □

This bound is fundamental because it is distribution-free; it holds for any data distribution $\mathcal{P}$. However, this universality comes at a cost: the bound is often loose for practical applications. For modern neural networks, the VC dimension is roughly proportional to the number of parameters, which can be in the billions. The VC bound would suggest that we need billions of samples to generalize, yet these networks generalize well with far fewer samples. This discrepancy motivates the study of data-dependent complexity measures.

3.4 Rademacher Complexity

Rademacher Complexity measures the ability of the hypothesis class to fit random noise. Unlike VC dimension, it depends on the specific distribution of the data.

Definition 3.7 (Empirical Rademacher Complexity)**.** Let $\sigma = (\sigma_1, \ldots, \sigma_n)$ be independent Rademacher variables where $P(\sigma_i = 1) = P(\sigma_i = -1) = 0.5$. The empirical Rademacher

complexity of $\mathcal{H}$ on a dataset $S = \{x_1, \ldots, x_n\}$ is:

$$\hat{\mathfrak{R}}_S(\mathcal{H}) \coloneqq \mathbb{E}_\sigma \left[\sup_{f \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i f(x_i) \right] \tag{26}$$

where $\mathbb{E}_\sigma$ denotes the expectation over the random variables σ.

Definition 3.8 (Rademacher Complexity)**.** The Rademacher complexity of $\mathcal{H}$ for sample size n is the expected value of the empirical Rademacher complexity over random samples of size n:

$$\mathfrak{R}_n(\mathcal{H}) \coloneqq \mathbb{E}_{S \sim \mathcal{P}^n}[\hat{\mathfrak{R}}_S(\mathcal{H})] \tag{27}$$

where $\mathcal{P}^n$ denotes the product distribution of n i.i.d. samples from $\mathcal{P}$.

Intuitively, if a hypothesis class is complex enough to correlate with random noise labels σ_i, it has high capacity and is prone to overfitting.

Example 3.2 (Rademacher Complexity in Handwritten Digit Recognition)**.** Consider classifying handwritten digits (e.g., MNIST, $d = 784$ pixel features). Suppose $\mathcal{H}$ consists of linear classifiers $f(x) = \text{sign}(w^T x)$ with $\|w\|_2 \leq W$ and inputs satisfy $\|x\|_2 \leq R$. The Rademacher complexity is $\hat{\mathfrak{R}}_S(\mathcal{H}) \leq WR/\sqrt{n}$. With $W = 1$, $R \approx 20$ (typical for normalized pixel vectors), and $n = 60000$ training images, this gives $\hat{\mathfrak{R}}_S \approx 20/\sqrt{60000} \approx 0.08$. The bound $\mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + 2 \times 0.08 + O(1/\sqrt{n})$ tells us that the true risk exceeds the training risk by at most ≈ 0.16, a meaningful guarantee that does not depend on the dimension $d = 784$, explaining why high-dimensional classifiers can still generalize.

Theorem 3.5 (Rademacher Generalization Bound)**.** *With probability at least $1 - \delta$, for all $f \in \mathcal{H}$:*

$$\mathcal{R}(f) \leq \hat{\mathcal{R}}(f) + 2\,\mathfrak{R}_n(\mathcal{H}) + \sqrt{\frac{\ln(1/\delta)}{2n}} \tag{28}$$

where $\mathcal{R}(f)$ is the expected risk, $\hat{\mathcal{R}}(f)$ is the empirical risk, and $\mathfrak{R}_n(\mathcal{H})$ is the Rademacher complexity.

Proof Sketch. The proof proceeds via symmetrization. Let $S = \{(x_i, y_i)\}_{i=1}^n$ be the training set and $S' = \{(x_i', y_i')\}_{i=1}^n$ be an independent "ghost" sample from the same distribution. By symmetry:

$$\mathbb{E}_S[\mathcal{R}(f) - \hat{\mathcal{R}}_S(f)] = \mathbb{E}_{S,S'}[\hat{\mathcal{R}}_{S'}(f) - \hat{\mathcal{R}}_S(f)] \tag{29}$$

Introducing Rademacher variables σ_i to randomly swap between S and S':

$$\mathbb{E}_{S,S'}[\hat{\mathcal{R}}_{S'}(f) - \hat{\mathcal{R}}_S(f)] = \mathbb{E}_{S,S',\sigma} \left[\frac{1}{n} \sum_{i=1}^{n} \sigma_i (\ell(f(x_i'), y_i') - \ell(f(x_i), y_i)) \right] \tag{30}$$

Since the loss is bounded, this can be related to the Rademacher complexity. Applying concentration inequalities (McDiarmid) gives the high-probability bound. See [10] for the complete proof. □

This bound is often tighter than VC bounds for neural networks because it captures the geometric structure of the data and the function class. For example, for linear models with bounded L_2 norm, the Rademacher complexity decays with $1/\sqrt{n}$ independent of the dimension d, explaining why kernel methods and neural networks can generalize in high-dimensional spaces.

3.5 The Double Descent Phenomenon

Classical statistical learning theory suggests a U-shaped curve for test error as a function of model complexity: error decreases as bias decreases, then increases as variance increases. However, modern deep learning exhibits a "Double Descent" phenomenon. In the **Under-parameterized Regime**, the model is not complex enough to fit the data, resulting in high training and test error (bias dominates). In the **Critical Regime**, the model capacity is just enough to fit the training data (interpolation threshold). Here, the model is extremely sensitive to noise, and variance explodes, leading to poor generalization. Finally, in the **Over-parameterized Regime**, as capacity increases further, there are infinitely many solutions that fit the training data. The optimization algorithm (e.g., SGD) acts as an implicit regularizer, selecting the solution with the minimum norm (e.g., smoothest function), causing test error to decrease again, often below the minimum of the classical U-curve.

This phenomenon challenges traditional notions of model selection and emphasizes the role of implicit regularization. In Trustworthy AI, understanding this regime is crucial because over-parameterized models, while accurate, may be more susceptible to adversarial attacks or privacy leakage due to their capacity to memorize individual data points. A key lesson is that scale and capacity are not inherently bad for generalization; the training algorithm and implicit biases matter.

Example 3.3 (Double Descent in Polynomial Regression)**.** Consider fitting a polynomial $\hat{f}(x) = \sum_{j=0}^{p} \beta_j x^j$ of degree p to $n = 20$ noisy data points from $y = \sin(2\pi x) + \epsilon$. For $p < 19$ (under-parameterized), the model cannot interpolate all points, and test error follows the classical U-shaped curve, minimized around $p \approx 5$. At $p = 19$ (interpolation threshold), the polynomial passes through all 20 points but oscillates wildly between them, producing enormous test error. For $p \gg 19$ (over-parameterized), there are infinitely many interpolating polynomials, and the minimum-norm solution (selected by numerical solvers) is smooth and generalizes well; the test error decreases again. This directly parallels neural networks: a network with width w reaches the interpolation threshold at a critical width, and wider networks generalize better due to implicit regularization by gradient descent.

3.6 PAC Learning Framework

The Probably Approximately Correct (PAC) learning framework, introduced by Valiant [69], provides a rigorous foundation for computational learning theory.

Definition 3.9 (PAC Learnability)**.** A concept class $\mathcal{C}$ is PAC-learnable if there exists an algorithm $\mathcal{A}$ and a polynomial function $p(\cdot, \cdot, \cdot, \cdot)$ such that for every $\epsilon, \delta > 0$, every distribution $\mathcal{D}$ on $\mathcal{X}$, and every target concept $c \in \mathcal{C}$, given access to at least $m \geq p(1/\epsilon, 1/\delta, d, \text{size}(c))$ samples drawn i.i.d. from $\mathcal{D}$, $\mathcal{A}$ outputs a hypothesis h satisfying:

$$P_{S \sim \mathcal{D}^m}[\mathcal{R}(h) \leq \epsilon] \geq 1 - \delta \tag{31}$$

where $\mathcal{R}(h) = \mathbb{E}_{x \sim \mathcal{D}}[\mathbb{I}(h(x) \neq c(x))]$ is the true error (generalization error), S is the training set of size m, and d is the dimensionality of the input space.

The key elements are that the algorithm succeeds **Probably** (with probability $1 - \delta$), produces a hypothesis that is **Approximately** correct (with error at most ϵ), and is **Correct** with respect to the true underlying concept.

Example 3.4 (PAC Learning Axis-Aligned Rectangles for Anomaly Detection)**.** In industrial quality control, suppose defective products have measurements (x_1, x_2) (e.g., temperature and

pressure during manufacturing) that fall within an unknown rectangular region $[a_1, b_1] \times [a_2, b_2]$. The hypothesis class $\mathcal{H}$ of axis-aligned rectangles in $\mathbb{R}^2$ has VC dimension 4. By the PAC learning theorem, to learn the defect region with error $\epsilon = 0.05$ (mislabeling at most 5% of products) and confidence $1 - \delta = 0.99$, we need $m = O\left(\frac{4+\log(1/0.01)}{0.05}\right) \approx O(172)$ labeled product samples. The ERM algorithm simply outputs the tightest rectangle containing all defective samples, a computationally efficient procedure that provably converges to the true defect region.

Theorem 3.6 (Sample Complexity of PAC Learning)**.** *If $\mathcal{H}$ has VC dimension d, then for any $\epsilon, \delta > 0$, a sample of size:*

$$m = O\left(\frac{d + \log(1/\delta)}{\epsilon}\right) \tag{32}$$

is sufficient for PAC learning. Furthermore, any PAC learning algorithm requires:

$$m = \Omega\left(\frac{d + \log(1/\delta)}{\epsilon}\right) \tag{33}$$

samples.

3.6.1 Agnostic PAC Learning

In the realizable setting, we assume $c \in \mathcal{C}$. The agnostic setting removes this assumption, competing with the best hypothesis in $\mathcal{H}$:

$$P[\mathcal{R}(h) \leq \inf_{h' \in \mathcal{H}} \mathcal{R}(h') + \epsilon] \geq 1 - \delta \tag{34}$$

This is more realistic as we cannot guarantee the true function belongs to our hypothesis class.

3.7 Margin-Based Generalization Bounds

For classification problems, the margin of a classifier provides a more refined measure of confidence than just correctness.

Definition 3.10 (Margin)**.** For a real-valued classifier $f : \mathcal{X} \to \mathbb{R}$ and label $y \in \{-1, +1\}$, the margin on example (x, y) is:

$$\gamma(x, y) \coloneqq y \cdot f(x) \tag{35}$$

A positive margin indicates correct classification; larger margins indicate higher confidence.

Theorem 3.7 (Margin Bound)**.** *Let $\mathcal{H}$ be a class of functions mapping to $[-1, 1]$. For any $\gamma > 0$, with probability at least $1 - \delta$:*

$$\mathcal{R}_{0-1}(f) \leq \hat{\mathcal{R}}_\gamma(f) + \frac{2}{\gamma}\mathfrak{R}_m(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2m}} \tag{36}$$

where $\hat{\mathcal{R}}_\gamma(f) = \frac{1}{m}\sum_{i=1}^{m} \mathbb{I}(y_i f(x_i) < \gamma)$ is the empirical margin loss, $\mathcal{R}_{0-1}(f)$ is the expected 0-1 loss, and m is the sample size.

This bound explains why maximizing margins (as in SVMs) leads to better generalization. For neural networks, the implicit bias of gradient descent toward large margin solutions partially explains their generalization ability.

3.8 Algorithmic Stability

An alternative approach to generalization bounds relies on the stability of the learning algorithm rather than the complexity of the hypothesis class.

Definition 3.11 (Uniform Stability)**.** An algorithm $\mathcal{A}$ is β-uniformly stable if for all datasets S, S' differing in exactly one example:

$$\sup_{(x,y)} |\ell(\mathcal{A}(S), (x, y)) - \ell(\mathcal{A}(S'), (x, y))| \leq \beta \tag{37}$$

Theorem 3.8 (Stability Bound)**.** *If $\mathcal{A}$ is β-uniformly stable and $\ell \leq M$, then with probability at least $1 - \delta$:*

$$\mathcal{R}(\mathcal{A}(S)) \leq \hat{\mathcal{R}}(\mathcal{A}(S)) + 2\beta + (4m\beta + M)\sqrt{\frac{\ln(1/\delta)}{2m}} \tag{38}$$

where m is the sample size, M is the upper bound on the loss function, and β is the stability parameter.

SGD with appropriate learning rates is known to be stable:

Theorem 3.9 (SGD Stability)**.** *For convex, L-Lipschitz loss functions, SGD with step size $\eta_t = 1/(Lt)$ run for T iterations is $O(1/m)$-stable.*

This provides generalization guarantees for deep learning without relying on explicit capacity measures.

Example 3.5 (Stability in Weather Prediction Models)**.** Consider training a neural network for temperature forecasting using $m = 10{,}000$ daily weather records. If the SGD training procedure achieves uniform stability $\beta = 10^{-4}$ (consistent with the $O(1/m)$ rate from the SGD Stability theorem above), then replacing any single training record changes the loss on any test example by at most 10^{-4}. The stability bound guarantees that the expected generalization gap is at most $2\beta = 2 \times 10^{-4}$. If the model achieves 1.5°C mean absolute error on training data, the expected true error is at most ≈ 1.5002°C, a strong guarantee that the model genuinely learns weather patterns rather than memorizing individual days, and that no single training record can disproportionately influence predictions.

Exercises

1. **Bias-Variance Trade-off**: Consider a regression problem $y = \sin(2\pi x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.1)$.
   (a) Generate a dataset of size $N = 10$.
   (b) Fit polynomials of degree $d = 0, 1, \ldots, 9$.
   (c) Plot the training error and test error as a function of d.
   (d) Decompose the error into bias and variance by repeating the experiment 100 times.
2. **VC Dimension**:
   (a) Prove that the VC dimension of a linear classifier in $\mathbb{R}^d$ is $d + 1$.
   (b) Prove that the VC dimension of the class of axis-aligned rectangles in $\mathbb{R}^2$ is 4.
   (c) Construct a set of 3 points in $\mathbb{R}^2$ that cannot be shattered by axis-aligned rectangles.
3. **Rademacher Complexity**:

(a) Show that $\hat{\mathfrak{R}}_S(\mathcal{H}) = \mathbb{E}_\sigma[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum \sigma_i h(x_i)]$ satisfies the bounded difference inequality.

(b) Derive the Rademacher complexity for linear hypotheses with bounded L_2 norm.

4. **PAC Learning**:

(a) Prove that any finite hypothesis class $\mathcal{H}$ is PAC learnable with sample complexity $m = O(\frac{\log|\mathcal{H}|+\log(1/\delta)}{\epsilon})$.

(b) Implement the Consistent algorithm for learning axis-aligned rectangles.

(c) Verify empirically that the sample complexity matches the theoretical bound.

(d) Extend to the agnostic case and compare sample requirements.

5. **Margin Theory**:

(a) Train an SVM on a binary classification task with varying regularization.

(b) Compute the margin achieved by each model.

(c) Plot test accuracy vs. margin size and verify the theoretical relationship.

(d) Compare hard-margin and soft-margin SVM generalization.

6. **Algorithmic Stability**:

(a) Implement SGD with different learning rate schedules.

(b) Empirically measure stability by comparing models trained on datasets differing by one sample.

(c) Verify that smaller learning rates lead to more stable algorithms.

(d) Compare stability of SGD vs. batch gradient descent.

7. **Double Descent**:

(a) Train models of increasing complexity on a fixed dataset (e.g., neural networks with varying widths).

(b) Plot training and test error as a function of model complexity.

(c) Identify the interpolation threshold where training error hits zero.

(d) Verify the double descent phenomenon: test error decreases again past the interpolation threshold.

8. **Generalization in Deep Learning**:

(a) Train a deep network with random labels ([78]).

(b) Compare training dynamics with true labels.

(c) Measure the effective capacity using a noise memorization metric.

(d) Discuss why traditional capacity bounds fail for deep networks.

9. **Concentration Inequalities**:

(a) Prove Hoeffding's inequality for bounded random variables.

(b) Use Hoeffding to derive a uniform convergence bound for finite hypothesis classes.

(c) Compare the tightness of Hoeffding vs. Bernstein bounds on empirical data.

(d) Apply McDiarmid's inequality to analyze the stability of cross-validation.

4 Adversarial Robustness

[14, 54, 63, 74, 84]

4.1 The Phenomenon of Adversarial Examples

The discovery that state-of-the-art deep neural networks are vulnerable to "adversarial examples"—inputs formed by applying small, often imperceptible perturbations to clean data that cause the model to make high-confidence errors—has shattered the illusion that these models learn human-like concepts. In the context of image classification, an adversary can add a specific noise pattern to an image of a panda, such that the modified image looks identical to the human eye but is classified by the network as a gibbon with 99.9% confidence. This vulnerability is not specific to computer vision; it pervades natural language processing, speech recognition, and reinforcement learning.

The existence of adversarial examples poses a severe security risk for safety-critical applications [66]. In personalized medicine, an AI-based diagnostic tool analyzing patient records might misclassify a severe condition as benign due to small, naturally occurring artifact noise in the clinical data, leading to catastrophic outcomes. In health insurance claims processing, subtle data perturbations could falsely justify denied coverage. Therefore, understanding the mathematical nature of these vulnerabilities and developing robust defenses is a prerequisite for the deployment of AI in the real world.

4.2 High-Dimensional Geometry and Concentration

Early hypotheses suggested that adversarial examples were a result of "overfitting" or the extreme non-linearity of deep networks. However, [27] proposed the "Linearity Hypothesis", arguing that the vulnerability arises from the model being *too linear* in high-dimensional spaces. This is a remarkable insight: neural networks are vulnerable not because they are too complex, but because they are too simple in the geometric structure they learn.

Theorem 4.1 (Linearity Hypothesis)**.** *Consider a linear model $f(x) = w^T x$. Let η be a perturbation bounded in L_∞ norm: $\|\eta\|_\infty \leq \epsilon$. The change in the model's output is $w^T \eta$. To maximize this change under the L_∞ constraint, the adversary should choose each component of η to align with the sign of the corresponding weight:*

$$\eta = \epsilon \cdot \text{sign}(w), \quad \Delta = w^T \eta = \epsilon \sum_{i=1}^{d} |w_i| = \epsilon \|w\|_1 \tag{39}$$

where $\text{sign}(w)$ *is the element-wise sign function, d is the dimensionality of the input, and $\|w\|_1$ is the L_1 norm of the weight vector. If the weights are independent with typical magnitude m, then $\|w\|_1 \approx d \cdot m$. Thus, the output change scales linearly with the dimension d. In high-dimensional spaces (e.g., a 224×224 RGB image has $d \approx 150{,}000$), even if each weight is small ($m = O(1)$), the total effect can be huge. A perturbation of $\epsilon = 1/255$ per pixel (imperceptible) causes $\Delta \approx 600$, easily crossing any decision boundary.*

Extension to Neural Networks: While real neural networks are nonlinear, if the local approximation around a data point is nearly linear (as is often the case when weights are not trained to be highly nonlinear), the same principle applies. The model's decision boundary can be expressed locally as a hyperplane, and adversarial perturbations along the normal direction to this hyperplane cause large changes in output.

More formally, adversarial examples are related to the **Concentration of Measure** phenomenon. In high-dimensional spaces, probability mass is not distributed uniformly; it concentrates in unexpected ways. For a unit sphere S^{d-1}, most of the measure is concentrated near the equator defined by any fixed hyperplane. This means that if a classifier's decision boundary passes through the typical data region, a randomly chosen point is likely very close to the boundary (in Euclidean distance) even though the input space is enormous. The Isoperimetric Inequality formalizes this: if a set A covers at least half the measure of the sphere, then expanding A by a small distance ϵ covers almost the entire sphere. Conversely, for any classifier with non-trivial error rate $p > 0$, the set of misclassified points has measure at least p, and is thus at distance at most $O(\sqrt{\log(1/p)/d})$ from the correctly classified points. The curse of dimensionality means this distance is vanishingly small as d grows.

4.3 Taxonomy of Adversarial Attacks

Adversarial attacks can be categorized based on the attacker's knowledge and goals.

Table 2: Taxonomy of Adversarial Attacks

Dimension	Description
Knowledge	
White-box	Attacker has full access to model weights, architecture, and gradients.
Black-box	Attacker only has query access (probabilities or hard labels).
Gray-box	Attacker has partial knowledge (e.g., architecture but not weights).
Goal	
Targeted	Misclassify input x as a specific target class y_{target}.
Untargeted	Misclassify input x as any class $y \neq y_{true}$.
Constraint	
L_∞	Maximum change to any single pixel is bounded (imperceptibility).
L_2	Euclidean distance is bounded.
L_0	Number of modified pixels is bounded (sparse attack).
Physical	Perturbations must be realizable in the physical world (e.g., medical imaging artifacts).

4.4 White-Box Attacks

In the white-box setting, the attacker has complete knowledge of the model architecture, parameters θ, and can compute gradients $\nabla_x \ell(f_\theta(x), y)$. This represents the strongest threat model and provides upper bounds on attack effectiveness.

4.4.1 Theoretical Foundation: First-Order Adversaries

The core insight behind gradient-based attacks is that neural network loss landscapes are locally linear in the input space. For a differentiable loss function ℓ, the first-order Taylor expansion around input x gives:

$$\ell(f_\theta(x + \delta), y) \approx \ell(f_\theta(x), y) + \langle \nabla_x \ell(f_\theta(x), y), \delta \rangle + O(\|\delta\|^2) \tag{40}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product, ∇_x is the gradient with respect to the input x, and $O(\|\delta\|^2)$ represents higher-order terms that are negligible for small δ.

To maximize this linear approximation subject to $\|\delta\|_p \leq \epsilon$, we solve:

$$\delta^* = \arg\max_{\|\delta\|_p \leq \epsilon} \langle g, \delta \rangle \quad \text{where } g = \nabla_x \ell(f_\theta(x), y) \tag{41}$$

where $\|\cdot\|_p$ is the L_p norm.

The solution depends on the choice of L_p norm constraint:

Proposition 4.2 (Optimal First-Order Perturbation)**.** *For gradient $g \in \mathbb{R}^d$ and constraint $\|\delta\|_p \leq \epsilon$:*

1. L_∞ ***constraint:*** $\delta^* = \epsilon \cdot \text{sign}(g)$*, achieving* $\langle g, \delta^* \rangle = \epsilon \|g\|_1$
2. L_2 ***constraint:*** $\delta^* = \epsilon \cdot \frac{g}{\|g\|_2}$*, achieving* $\langle g, \delta^* \rangle = \epsilon \|g\|_2$
3. L_1 ***constraint:*** $\delta^* = \epsilon \cdot e_{i^*} \cdot \text{sign}(g_{i^*})$ *where* $i^* = \arg\max_i |g_i|$*, achieving* $\langle g, \delta^* \rangle = \epsilon \|g\|_\infty$

Proof. We use Hölder's inequality: $|\langle g, \delta \rangle| \leq \|g\|_q \|\delta\|_p$ where $\frac{1}{p} + \frac{1}{q} = 1$. Equality holds when δ is aligned with g in the dual norm sense.

Case 1 (L_∞ constraint, $p = \infty$, $q = 1$): We have $\langle g, \delta \rangle \leq \|g\|_1 \|\delta\|_\infty \leq \epsilon \|g\|_1$. To achieve equality, set $\delta_i = \epsilon \cdot \text{sign}(g_i)$. Then:

$$\langle g, \delta \rangle = \sum_{i=1}^{d} g_i \cdot \epsilon \cdot \text{sign}(g_i) = \epsilon \sum_{i=1}^{d} |g_i| = \epsilon \|g\|_1 \tag{42}$$

Case 2 (L_2 constraint, $p = q = 2$): By Cauchy-Schwarz, $\langle g, \delta \rangle \leq \|g\|_2 \|\delta\|_2 \leq \epsilon \|g\|_2$. Equality holds when $\delta = \epsilon \frac{g}{\|g\|_2}$:

$$\langle g, \delta \rangle = \left\langle g, \epsilon \frac{g}{\|g\|_2} \right\rangle = \frac{\epsilon}{\|g\|_2} \|g\|_2^2 = \epsilon \|g\|_2 \tag{43}$$

Case 3 (L_1 constraint, $p = 1$, $q = \infty$): We have $\langle g, \delta \rangle \leq \|g\|_\infty \|\delta\|_1 \leq \epsilon \|g\|_\infty$. To achieve equality, place all mass on the coordinate with largest $|g_i|$: let $i^* = \arg\max_i |g_i|$, set $\delta_{i^*} = \epsilon \cdot \text{sign}(g_{i^*})$ and $\delta_i = 0$ for $i \neq i^*$. Then $\|\delta\|_1 = \epsilon$ and:

$$\langle g, \delta \rangle = g_{i^*} \cdot \epsilon \cdot \text{sign}(g_{i^*}) = \epsilon |g_{i^*}| = \epsilon \|g\|_\infty \tag{44}$$

□

4.4.2 Fast Gradient Sign Method (FGSM)

FGSM [27] is the canonical one-step attack for L_∞ perturbations:

$$x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x \ell(f_\theta(x), y)) \tag{45}$$

where x_{adv} is the adversarial example, x is the original input, ϵ is the perturbation magnitude, and $\text{sign}(\cdot)$ is the element-wise sign function.

Remark 4.1 (Geometric Interpretation)**.** FGSM moves in the direction that maximizes the linear approximation of the loss. The sign function projects onto the vertices of the L_∞ ball, making each pixel change by exactly $\pm\epsilon$. This is optimal for the linearized problem but may be suboptimal for the true (nonlinear) loss.

Proposition 4.3 (FGSM Loss Increase)**.** *For an L-smooth loss function (i.e., with L-Lipschitz continuous gradients), FGSM achieves:*

$$\ell(f_\theta(x_{adv}), y) - \ell(f_\theta(x), y) \geq \epsilon \|\nabla_x \ell\|_1 - \frac{L\epsilon^2 d}{2} \tag{46}$$

where d is the input dimension.

Proof. By Taylor expansion around x:

$$\ell(f_\theta(x_{adv}), y) = \ell(f_\theta(x), y) + \langle \nabla_x \ell, x_{adv} - x \rangle + O(\|x_{adv} - x\|^2) \tag{47}$$

For FGSM, $x_{adv} - x = \epsilon \cdot \text{sign}(\nabla_x \ell)$, so:

$$\langle \nabla_x \ell, \epsilon \cdot \text{sign}(\nabla_x \ell) \rangle = \epsilon \sum_{i=1}^{d} |(\nabla_x \ell)_i| = \epsilon \|\nabla_x \ell\|_1 \tag{48}$$

For the second-order term, by L-smoothness (i.e., the gradient of ℓ is L-Lipschitz in L_2 norm):

$$|\ell(f_\theta(x_{adv}), y) - \ell(f_\theta(x), y) - \langle \nabla_x \ell, x_{adv} - x \rangle| \leq \frac{L}{2} \|x_{adv} - x\|_2^2 = \frac{L\epsilon^2 d}{2} \tag{49}$$

Since for FGSM $\|x_{adv} - x\|_2^2 = d\epsilon^2$ (each of the d components changes by $\pm\epsilon$), we get the stated bound. □

The computational efficiency of FGSM (single forward-backward pass) makes it suitable for generating large-scale adversarial training data.

4.4.3 Projected Gradient Descent (PGD)

PGD [44] is the multi-step extension of FGSM that iteratively refines the adversarial perturbation:

$$x^{t+1} = \Pi_{B_\epsilon(x)} \left(x^t + \alpha \cdot \text{sign}(\nabla_x \ell(f_\theta(x^t), y)) \right) \tag{50}$$

where $\Pi_{B_\epsilon(x)}$ projects onto the ϵ-ball: $\Pi_{B_\epsilon(x)}(z) = x + \text{clip}(z - x, -\epsilon, \epsilon)$, x^t is the adversarial example at iteration t, and α is the step size.

Algorithm 1 Projected Gradient Descent (PGD) Attack

Input: $x, y, \theta, \epsilon, \alpha, T$
$x^0 \leftarrow x + \mathcal{U}(-\epsilon, \epsilon)$ {Random Start}
for $t = 1$ to T **do**
  $g \leftarrow \nabla_x \ell(f_\theta(x^{t-1}), y)$
  $x^t \leftarrow \Pi_{B_\epsilon(x)}(x^{t-1} + \alpha \cdot \text{sign}(g))$
end for
return x^T

Theorem 4.4 (PGD Convergence)**.** *For an L-smooth loss function and step size $\alpha \leq \frac{1}{L}$, PGD converges to a stationary point of the constrained maximization problem. Specifically, the squared norm of the gradient mapping decays as:*

$$\min_{t \leq T} \left\| \frac{x^{t+1} - x^t}{\alpha} \right\|_2^2 \leq \frac{2L(\ell(x^*) - \ell(x^0))}{T} \tag{51}$$

where x^* is a global maximum.

Remark 4.2 (Random Restarts)**.** PGD with random restarts runs multiple independent attacks from different initializations and returns the best. This helps escape local maxima and provides stronger attacks. Empirically, 10-20 restarts often suffice for most images.

Proposition 4.5 (Attack Strength Hierarchy)**.** *For identical perturbation budgets ϵ:*

$$\textit{Success}(\textit{FGSM}) \leq \textit{Success}(\textit{PGD}) \leq \textit{Success}(\textit{PGD-Restarts}) \leq \textit{Success}(\textit{C\&W}) \tag{52}$$

The gap between FGSM and PGD is larger for robust models trained with adversarial training.

4.4.4 Carlini & Wagner (C&W) Attack

The C&W attack [13] finds minimum-norm perturbations by reformulating the attack as an unconstrained optimization problem.

Definition 4.1 (C&W Objective)**.** The attack minimizes:

$$\min_{\delta} \|\delta\|_2^2 + c \cdot f(x+\delta) \tag{53}$$

where $c > 0$ is a constant balancing the perturbation size and the misclassification success, and f is a surrogate objective for misclassification:

$$f(x') = \max(\max_{i \neq t} Z(x')_i - Z(x')_t, -\kappa) \tag{54}$$

Here $Z(x')$ are the logits (pre-softmax outputs), t is the target class, and $\kappa \geq 0$ is a confidence parameter.

Remark 4.3 (Change of Variables)**.** To ensure $x+\delta \in [0,1]^d$ (valid pixel range), C&W introduces:

$$\delta = \frac{1}{2}(\tanh(w) + 1) - x \tag{55}$$

and optimizes over unconstrained $w \in \mathbb{R}^d$. This guarantees box constraints are always satisfied.

Proposition 4.6 (C&W Optimality)**.** *For the L_2 attack, C&W finds perturbations that are on average 10× smaller than PGD perturbations for the same attack success rate. However, it requires ∼10,000 optimization steps per image.*

4.4.5 AutoAttack: Reliable Evaluation

AutoAttack [19] combines multiple attacks for reliable robustness evaluation:

1. **APGD-CE**: Auto-PGD with cross-entropy loss and adaptive step sizes
2. **APGD-DLR**: Auto-PGD with Difference of Logits Ratio loss
3. **FAB**: Fast Adaptive Boundary attack
4. **Square Attack**: Score-based black-box attack

Theorem 4.7 (AutoAttack Completeness)**.** *AutoAttack provides a lower bound on adversarial accuracy that is tight up to 0.1% for most defenses tested on standard benchmarks. Any defense claiming higher robustness should be evaluated against AutoAttack.*

4.5 Black-Box Attacks

In the black-box setting, the attacker cannot access model gradients or architecture. This is the more realistic threat model for deployed systems.

4.5.1 Transfer Attacks

Transfer attacks exploit the phenomenon that adversarial examples generated for one model often fool other models.

Definition 4.2 (Adversarial Transferability)**.** An adversarial example x_{adv} generated against source model f_s *transfers* to target model f_t if:

$$f_s(x_{adv}) \neq y \implies f_t(x_{adv}) \neq y \tag{56}$$

The transfer rate is $P(f_t(x_{adv}) \neq y \mid f_s(x_{adv}) \neq y)$, where P denotes probability.

Theorem 4.8 (Transferability Conditions)**.** *Adversarial examples transfer more reliably when:*

1. *Source and target models have similar decision boundaries*
2. *The perturbation exploits input features rather than model-specific artifacts*
3. *Ensemble methods are used:* $x_{adv} = \arg\max_{\|\delta\| \leq \epsilon} \sum_{i=1}^{K} w_i \ell(f_i(x + \delta), y)$

4.5.2 Query-Based Attacks

Query-based attacks estimate gradients through model queries.

Definition 4.3 (Zeroth-Order Gradient Estimation)**.** The gradient can be approximated using finite differences:

$$\hat{g}_i = \frac{f(x + \mu e_i) - f(x - \mu e_i)}{2\mu} \tag{57}$$

where e_i is the i-th standard basis vector and $\mu > 0$ is the step size.

This requires $O(d)$ queries per gradient estimate where d is input dimension—prohibitively expensive for images.

Proposition 4.9 (Random Direction Gradient Estimation)**.** *A more efficient estimator uses random directions* $u \sim \mathcal{N}(0, I)$*:*

$$\hat{g} = \frac{d}{\mu} \cdot (f(x + \mu u) - f(x)) \cdot u \tag{58}$$

This provides an unbiased estimate: $\mathbb{E}[\hat{g}] = \nabla f(x)$ *with* $O(1)$ *queries per estimate.*

4.5.3 Score-Based Attacks: Square Attack

Square Attack [7] uses random search without gradient estimation:

Proposition 4.10 (Query Efficiency)**.** *Square Attack achieves comparable success rates to PGD with only ∼1000-5000 queries, making it practical for attacking real-world APIs.*

Algorithm 2 NES (Natural Evolution Strategies) Attack

Input: $x, y, f, \epsilon, \alpha, T, n$
$\delta^0 \leftarrow 0$
for $t = 1$ to T **do**
  Sample $u_1, \ldots, u_n \sim \mathcal{N}(0, I)$
  $\hat{g} \leftarrow \frac{1}{n\sigma} \sum_{i=1}^{n} (f(x + \delta^{t-1} + \sigma u_i) - f(x + \delta^{t-1} - \sigma u_i)) \cdot u_i$
  $\delta^t \leftarrow \Pi_{B_\epsilon}(\delta^{t-1} + \alpha \cdot \text{sign}(\hat{g}))$
end for
return $x + \delta^T$

Algorithm 3 Square Attack

Input: x, y, f, ϵ, T
Initialize δ with random $\pm\epsilon$ values
for $t = 1$ to T **do**
  Sample random square region S of size $p \times p$
  Sample new values $v \in \{-\epsilon, +\epsilon\}^{p \times p}$
  $\delta' \leftarrow \delta$; set $\delta'_S \leftarrow v$
  if $\ell(f(x + \delta'), y) > \ell(f(x + \delta), y)$ **then**
    $\delta \leftarrow \delta'$ {Accept update}
  end if
end for
return $x + \delta$

4.6 Adversarial Training and Robust Optimization

The most effective defense against adversarial attacks is Adversarial Training. We formalize robustness as a min-max saddle point problem. The goal is to find model parameters θ that minimize the loss on the worst-case perturbations within the allowed set $\mathcal{S}$.

$$\theta^* = \arg\min_\theta \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{\delta \in \mathcal{S}} \ell(f_\theta(x + \delta), y) \right] \tag{59}$$

where $\mathcal{S} = \{\delta : \|\delta\|_p \leq \epsilon\}$ is the set of allowed perturbations, $\mathcal{D}$ is the data distribution, and θ represents the model parameters. This formulation views training as a zero-sum game between the trainer (minimizer) and the attacker (maximizer).

4.6.1 Game-Theoretic Interpretation

Definition 4.4 (Zero-Sum Game)**.** Adversarial training corresponds to finding a Nash equilibrium of the zero-sum game:

$$(\theta^*, \delta^*) = \text{Nash} \left(\min_\theta \max_\delta \ell(f_\theta(x + \delta), y) \right) \tag{60}$$

At equilibrium, neither player can improve by unilateral deviation.

Theorem 4.11 (Minimax Theorem)**.** *For convex-concave objectives where $\ell(\cdot, \delta)$ is convex in θ and $\ell(\theta, \cdot)$ is concave in δ:*

$$\min_\theta \max_\delta \ell(f_\theta(x + \delta), y) = \max_\delta \min_\theta \ell(f_\theta(x + \delta), y) \tag{61}$$

The order of optimization does not matter.

Remark 4.4 (Non-Convexity in Practice)**.** Neural network losses are non-convex in θ, so the minimax theorem does not directly apply. Adversarial training uses alternating optimization: fix θ, maximize over δ (PGD attack); fix δ^*, minimize over θ (gradient descent).

4.6.2 Danskin's Theorem and Training Dynamics

To optimize the outer minimization via Gradient Descent, we need to compute the gradient of the value function $V(\theta) = \max_{\delta \in \mathcal{S}} \ell(f_\theta(x+\delta), y)$. The challenge is that the max operator is not differentiable everywhere. Danskin's theorem provides a solution by showing that the gradient of the max is the gradient at the maximizing point.

Theorem 4.12 (Danskin's Theorem)**.** *Let $\Phi : \Theta \times \mathcal{S} \to \mathbb{R}$ be a continuous function where $\mathcal{S}$ is a compact set. Define the value function:*

$$V(\theta) = \max_{\delta \in \mathcal{S}} \Phi(\theta, \delta) \tag{62}$$

Let $\delta^*(\theta) = \arg\max_{\delta \in \mathcal{S}} \Phi(\theta, \delta)$. If:

1. *Φ is continuously differentiable in θ for each δ*
2. *$\delta^*(\theta)$ is unique*

Then $V(\theta)$ is differentiable and:

$$\nabla_\theta V(\theta) = \nabla_\theta \Phi(\theta, \delta^*(\theta)) \tag{63}$$

Proof Sketch. By the envelope theorem from optimization theory. Consider a small perturbation $\theta' = \theta + \epsilon h$:

$$V(\theta') - V(\theta) = \max_\delta \Phi(\theta', \delta) - \max_\delta \Phi(\theta, \delta) \tag{64}$$

$$= \Phi(\theta', \delta^*(\theta')) - \Phi(\theta, \delta^*(\theta)) \tag{65}$$

By continuity of δ^* and differentiability of Φ, taking the limit as $\epsilon \to 0$ gives the result. The key insight is that at the optimum, the derivative with respect to δ is zero (by first-order optimality conditions), so only the direct effect through θ remains. □

Remark 4.5 (Application to Adversarial Training)**.** This theorem provides the theoretical justification for adversarial training: to update the model weights, we simply find the optimal adversarial example δ^* (using PGD) and then compute the gradient of the loss at that perturbed point. By Danskin's theorem, this is the gradient of the robust objective $\max_{\|\delta\| \leq \epsilon} \ell(f_\theta(x+\delta), y)$ with respect to θ.

Example 4.1 (Danskin's Theorem in Adversarial Training of an Image Classifier)**.** Consider training a convolutional network to classify chest X-rays as normal or abnormal, with robustness to ℓ_∞ perturbations of size $\epsilon = 0.01$ (modeling scanner noise variations). At each training step for an X-ray image x: (1) we run $K = 10$ steps of PGD to find the worst-case perturbation $\delta^* = \arg\max_{\|\delta\|_\infty \leq 0.01} \ell(f_\theta(x+\delta), y)$; (2) by Danskin's theorem, we compute the gradient $\nabla_\theta \ell(f_\theta(x+\delta^*), y)$, which equals $\nabla_\theta V(\theta)$, the gradient of the worst-case loss. Without this theorem, computing $\nabla_\theta[\max_\delta \ell]$ would require differentiating through the entire PGD optimization (expensive second-order computation). Danskin's theorem shows the simple approach of "find the worst input, then backpropagate" is mathematically correct.

4.6.3 TRADES: Trading Off Robustness and Accuracy

TRADES [79] decomposes the robust error into a natural error term and a boundary error term.

Definition 4.5 (TRADES Objective)**.** The TRADES algorithm minimizes a surrogate loss that balances natural accuracy and local Lipschitz continuity (robustness):

$$\mathcal{L}_{TRADES} = \ell(f_\theta(x), y) + \beta \cdot \max_{\|\delta\| \leq \epsilon} D_{KL}(f_\theta(x) \| f_\theta(x+\delta)) \tag{66}$$

where $\beta > 0$ controls the trade-off, D_{KL} is the Kullback-Leibler divergence, and ϵ is the perturbation radius. The first term encourages the model to be accurate on clean data, while the second term (regularizer) pushes the decision boundary away from the data points by penalizing changes in the output distribution within the ϵ-ball.

This formulation is motivated by a theoretical bound on the robust error, where the KL divergence serves as a differentiable proxy for the condition that the prediction should not change within the perturbation radius. Higher β yields more robust but potentially less accurate models on clean data.

Example 4.2 (TRADES in Medical Image Segmentation)**.** Consider a diagnostic AI system classifying benign vs. malignant tumors from biopsy slides with $\epsilon = 8/255$ (an imperceptible pixel-level perturbation). With $\beta = 1$ (low robustness weight), the model achieves 96% clean accuracy and 40% robust accuracy under PGD attack. Increasing to $\beta = 6$ (standard TRADES setting) yields 84% clean accuracy but 56% robust accuracy. At $\beta = 20$, clean accuracy drops to 75% while robust accuracy plateaus at 58%. The first loss term $\ell(f_\theta(x), y)$ trains the model to correctly classify clean slides, while the second term $\beta \cdot \max_{\|\delta\| \leq \epsilon} D_{KL}(f_\theta(x) \| f_\theta(x+\delta))$ ensures the model's output distribution remains stable under perturbations, preventing, for example, a malignant biopsy from being reclassified as benign due to small scanning artifacts or lighting variations.

4.7 Certified Robustness via Randomized Smoothing

While PGD provides empirical robustness, it offers no guarantees. A stronger adversary might still find a loophole. Certified robustness aims to provide a mathematical proof that for a given input x, no perturbation within radius R can change the label. Randomized smoothing is a scalable technique that converts any base classifier f into a smoothed classifier g that is certifiably robust in L_2 norm.

Definition 4.6 (Smoothed Classifier)**.** Let $f : \mathbb{R}^d \to \mathcal{Y}$ be a base classifier (which need not be robust). The smoothed classifier g is defined as the majority vote of f under Gaussian noise:

$$g(x) = \arg\max_{c \in \mathcal{Y}} P_{\epsilon \sim \mathcal{N}(0, \sigma^2 I)}(f(x+\epsilon) = c) \tag{67}$$

where $\mathcal{N}(0, \sigma^2 I)$ is a multivariate Gaussian distribution with mean 0 and covariance $\sigma^2 I$, and P denotes probability.

Theorem 4.13 (Certification [18])**.** *Suppose we estimate that class c_A has probability $p_A = P(f(x+\epsilon) = c_A)$ and the runner-up class has probability $p_B = \max_{c \neq c_A} P(f(x+\epsilon) = c)$. If $p_A > p_B$, then the smoothed classifier $g(x)$ is provably robust within an L_2 radius R [18]:*

$$R = \frac{\sigma}{2}(\Phi^{-1}(p_A) - \Phi^{-1}(p_B)) \tag{68}$$

where Φ^{-1} is the inverse CDF of the standard normal distribution.

Example 4.3 (Certified Robustness of a Smoothed Classifier for Satellite Imagery)**.** Suppose we apply randomized smoothing with $\sigma = 0.5$ to a land-cover classifier for satellite images. For a test image of a forest region, we estimate (via 10,000 Monte Carlo samples) that $p_A = P(f(x+\epsilon) =$ "forest") $= 0.85$ and the runner-up probability is $p_B = 0.10$. Then:

$$R = \frac{0.5}{2}\left(\Phi^{-1}(0.85) - \Phi^{-1}(0.10)\right) = 0.25 \times (1.036 - (-1.282)) = 0.25 \times 2.318 = 0.580$$

This certifies that *no* ℓ_2-perturbation of norm ≤ 0.580 can change the classification from "forest" to any other class. In normalized pixel space, this radius corresponds to meaningful robustness against atmospheric distortions, sensor noise, or adversarial tampering. If instead $p_A = 0.55$ (less confident), the certified radius drops to $R = 0.25 \times (\Phi^{-1}(0.55) - \Phi^{-1}(0.40)) \approx 0.25 \times 0.378 = 0.095$, illustrating that higher confidence translates directly to larger certified regions.

Proof Sketch. The key insight is that for any perturbation $\|\delta\|_2 \leq R$, we need to show $P(f(x + \delta + \epsilon) = c_A) > P(f(x + \delta + \epsilon) = c_B)$ for all classes $c_B \neq c_A$, where $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$.

Consider the worst-case perturbation δ with $\|\delta\|_2 = R$. By the Neyman-Pearson lemma, the optimal decision boundary for distinguishing between $\mathcal{N}(0, \sigma^2 I)$ and $\mathcal{N}(\delta, \sigma^2 I)$ is a hyperplane perpendicular to δ. The perturbation δ shifts the Gaussian, and we need to compute how much probability mass crosses from class A to class B.

For a fixed direction (without loss of generality, take $\delta = R \cdot e_1$), let $Z \sim \mathcal{N}(0, \sigma^2)$ be the projection onto the first coordinate. At the original point x:

$$P(f(x+\epsilon) = c_A) = P(Z > t_A) = \Phi\left(\frac{-t_A}{\sigma}\right) \tag{69}$$

$$P(f(x+\epsilon) = c_B) = P(Z > t_B) = \Phi\left(\frac{-t_B}{\sigma}\right) \tag{70}$$

for some thresholds t_A, t_B. After perturbation by δ, the Gaussian shifts by R:

$$P(f(x+\delta+\epsilon) = c_A) = \Phi\left(\frac{R - t_A}{\sigma}\right) \tag{71}$$

$$P(f(x+\delta+\epsilon) = c_B) = \Phi\left(\frac{R - t_B}{\sigma}\right) \tag{72}$$

Since $p_A > p_B$, we have $t_A < t_B$, and by monotonicity of Φ we get $\Phi\left(\frac{R-t_A}{\sigma}\right) > \Phi\left(\frac{R-t_B}{\sigma}\right)$ for *any* R. However, this 1D model oversimplifies how perturbation redistributes probability mass across classes: in the full analysis, the worst-case redistribution (not merely a mean shift in a single threshold model) is governed by the Neyman-Pearson lemma.

Setting $\Phi(-t_A/\sigma) = p_A$ and $\Phi(-t_B/\sigma) = p_B$ gives $t_A = -\sigma\Phi^{-1}(p_A)$ and $t_B = -\sigma\Phi^{-1}(p_B)$. The gap $t_B - t_A = \sigma(\Phi^{-1}(p_A) - \Phi^{-1}(p_B))$ characterizes the separation between the two class thresholds and provides the correct scaling.

The rigorous proof applies the Neyman-Pearson lemma to the likelihood ratio of the full d-dimensional Gaussians $\mathcal{N}(x, \sigma^2 I)$ and $\mathcal{N}(x + \delta, \sigma^2 I)$, establishing that the worst-case probability transfer occurs along the perturbation direction. This yields the certified radius $R = \frac{\sigma}{2}(\Phi^{-1}(p_A) - \Phi^{-1}(p_B))$, where the factor of $\frac{1}{2}$ arises from the optimal hypothesis test. See [18] for the complete proof. □

Exercises

1. **FGSM, PGD, and Variants**:

(a) Implement the Fast Gradient Sign Method (FGSM) and PGD on the MNIST dataset.

(b) Plot the robust accuracy as a function of ϵ for both attacks.

(c) Why is PGD considered a stronger attack than FGSM?

(d) Compare the running time of FGSM vs. PGD for the same perturbation budget.

(e) Implement targeted FGSM that forces misclassification to a specific class.

(f) Implement I-FGSM (iterative FGSM) with step size α and iterations T.

(g) Compare attack success rates and perturbation magnitudes across all variants.

2. **Lipschitz Constant**:

(a) Compute the Lipschitz constant of a single linear layer $f(x) = Wx$.

(b) Compute the Lipschitz constant of the ReLU activation function.

(c) Show that the Lipschitz constant of a deep network is upper bounded by the product of the spectral norms of its weight matrices.

(d) Implement spectral normalization and verify it constrains the Lipschitz constant.

3. **Randomized Smoothing**:

(a) Prove that if $f(x)$ is the base classifier, the smoothed classifier $g(x)$ is Lipschitz continuous.

(b) Calculate the certified radius for a point x if $p_A = 0.99$ and $\sigma = 0.25$.

(c) Implement randomized smoothing certification and plot the certified accuracy vs. radius trade-off.

4. **Adversarial Training**:

(a) Implement adversarial training with PGD for CIFAR-10.

(b) Compare the clean accuracy and robust accuracy before and after adversarial training.

(c) Investigate the accuracy-robustness trade-off: plot clean accuracy vs. robust accuracy for different ϵ values during training.

(d) Examine what features the adversarially trained model learns by visualizing filters.

5. **Transferability Analysis**:

(a) Train three different architectures (ResNet, VGG, DenseNet) on CIFAR-10.

(b) Generate adversarial examples for each model.

(c) Test each set of adversarial examples on all three models.

(d) Construct a transferability matrix and analyze which model pairs have highest transfer rates.

6. **Black-Box Attacks**:

(a) Implement the Square Attack algorithm.

(b) Compare its query efficiency with zeroth-order gradient estimation methods.

(c) How does the attack success rate change with the number of queries?

7. **Certified Defense Evaluation**:

(a) Train a model with certified training (IBP or CROWN-IBP).

(b) Compare its certified accuracy with empirical PGD accuracy.

(c) Why might there be a gap between certified and empirical robustness?

8. **C&W Attack**:
   (a) Implement the Carlini & Wagner L_2 attack with variable w and binary search over c.
   (b) Compare with PGD in terms of perturbation size and attack success rate.
   (c) Implement the L_∞ variant and compare effectiveness.
   (d) Test against a defended model and analyze the difference in required perturbation.
9. **AutoAttack Benchmark**:
   (a) Evaluate a robust model using the AutoAttack benchmark.
   (b) Compare accuracy under APGD-CE, APGD-DLR, FAB, and Square attacks.
   (c) Analyze which attack is most effective for your model architecture.
   (d) Implement a simple "robust" defense and test if AutoAttack breaks it.
10. **Physical Adversarial Attacks**:
   (a) Implement the Expectation over Transformation (EOT) attack.
   (b) Generate an adversarial patch that remains effective under affine transformations.
   (c) Test robustness to printing and photographing the adversarial example.
   (d) Analyze the trade-off between physical robustness and perturbation magnitude.
11. **Adversarial Training Variants**:
   (a) Implement TRADES: trade-off between accuracy and robustness.
   (b) Compare clean/robust accuracy of AT, TRADES, and MART.
   (c) Implement Friendly Adversarial Training (FAT) with early stopping.
   (d) Analyze the effect of perturbation budget scheduling during training.

4.8 Natural Distribution Shifts and Domain Adaptation

While adversarial perturbations represent worst-case distribution shifts, natural distribution shifts arise from legitimate changes in the data-generating process. These are often more relevant in practice.

4.8.1 Types of Natural Distribution Shift

Distribution shifts arise when the deployment data distribution $P_T(X, Y)$ differs from the training distribution $P_S(X, Y)$. The four primary types: **Covariate Shift** ($P_T(X) \neq P_S(X)$, $P_T(Y|X) = P_S(Y|X)$), **Label Shift** ($P_T(Y) \neq P_S(Y)$, $P_T(X|Y) = P_S(X|Y)$), **Concept Drift** ($P_T(Y|X) \neq P_S(Y|X)$), and **Domain Shift** (both marginal and conditional distributions differ) are formally defined with detailed explanations and concrete examples in Section 11.1.

4.8.2 Importance Weighting for Covariate Shift

Under covariate shift, the optimal predictor remains the same, but the empirical risk estimator is biased. We can correct this via importance weighting:

$$\hat{R}_T(f) = \frac{1}{n}\sum_{i=1}^{n} \frac{P_T(x_i)}{P_S(x_i)} \ell(f(x_i), y_i) \tag{73}$$

where n is the number of samples, $P_T(x_i)$ and $P_S(x_i)$ are the target and source densities evaluated at x_i, and ℓ is the loss function.

The density ratio $w(x) = P_T(x)/P_S(x)$ can be estimated without estimating the densities directly using kernel mean matching or discriminative methods.

Theorem 4.14 (Consistency Under Covariate Shift)**.** *If $P_T(Y|X) = P_S(Y|X)$ and the importance weights are correctly specified, then the importance-weighted empirical risk is an unbiased estimator of the target risk:*

$$\mathbb{E}_{P_S}\left[w(X)\ell(f(X), Y)\right] = \mathbb{E}_{P_T}\left[\ell(f(X), Y)\right] \tag{74}$$

Proof. By definition of the importance weight $w(x) = P_T(x)/P_S(x)$ and the tower property of expectation:

$$\mathbb{E}_{P_S}\left[w(X)\ell(f(X), Y)\right] = \mathbb{E}_{P_S}\left[\frac{P_T(X)}{P_S(X)}\ell(f(X), Y)\right] \tag{75}$$

$$= \int\int \frac{P_T(x)}{P_S(x)} \ell(f(x), y) P_S(y|x) P_S(x)\, dy\, dx \tag{76}$$

$$= \int\int \ell(f(x), y) P_S(y|x) P_T(x)\, dy\, dx \tag{77}$$

Under the covariate shift assumption $P_S(Y|X) = P_T(Y|X)$:

$$= \int\int \ell(f(x), y) P_T(y|x) P_T(x)\, dy\, dx = \mathbb{E}_{P_T}\left[\ell(f(X), Y)\right] \tag{78}$$

□

4.8.3 Domain Adversarial Training

Domain Adversarial Neural Networks (DANN) learn representations that are predictive of Y but invariant to the domain D.

The architecture consists of:

- Feature extractor: $G_f : \mathcal{X} \to \mathcal{Z}$
- Label predictor: $G_y : \mathcal{Z} \to \mathcal{Y}$
- Domain classifier: $G_d : \mathcal{Z} \to \{S, T\}$

The objective combines supervised loss and adversarial domain loss:

$$\min_{G_f, G_y} \max_{G_d} \mathbb{E}_{(x,y)\sim P_S}\left[\ell(G_y(G_f(x)), y)\right] - \lambda\, \mathbb{E}_{x\sim P_S \cup P_T}\left[\ell_d(G_d(G_f(x)), d)\right] \tag{79}$$

where $\lambda > 0$ is a hyperparameter balancing the classification and domain confusion objectives, ℓ is the task loss, and ℓ_d is the domain classification loss.

The Gradient Reversal Layer (GRL) enables end-to-end training by negating gradients from the domain classifier.

4.8.4 Invariant Risk Minimization (IRM)

IRM seeks representations such that the optimal classifier on top is invariant across environments.

Definition 4.7 (Invariant Representation)**.** A representation Φ elicits an invariant predictor w if w is simultaneously optimal for all environments:

$$w \in \arg\min_{\bar{w}} R^e(\bar{w} \circ \Phi) \quad \forall e \in \mathcal{E}_{train} \tag{80}$$

The IRM objective relaxes this to a penalty:

$$\min_{\Phi, w} \sum_{e \in \mathcal{E}_{train}} R^e(w \circ \Phi) + \lambda \|\nabla_{w|w=1.0} R^e(w \circ \Phi)\|^2 \tag{81}$$

where $\mathcal{E}_{train}$ is the set of training environments, R^e is the risk in environment e, Φ is the representation learner, w is the linear classifier, and λ is the regularization strength.

The gradient penalty encourages $w = 1$ to be optimal in all environments, which happens when the representation captures invariant causal features.

4.8.5 Distributionally Robust Optimization (DRO)

DRO optimizes for the worst-case distribution within an uncertainty set:

$$\min_{\theta} \sup_{Q \in \mathcal{U}} \mathbb{E}_Q[\ell(f_\theta(X), Y)] \tag{82}$$

where $\mathcal{U}$ is the uncertainty set of distributions, Q is a distribution within that set, and θ are the model parameters.

Common uncertainty sets include:

- **f-divergence ball**: $\mathcal{U} = \{Q : D_f(Q\|P) \leq \rho\}$

  Detailed Explanation: The f-divergence family includes KL-divergence, χ^2-divergence, and total variation distance. This uncertainty set contains all distributions Q whose f-divergence from the empirical distribution P is at most ρ. The choice of f determines the geometry of the uncertainty set: KL-divergence corresponds to an exponential family, while χ^2-divergence has computationally attractive quadratic structure. The radius ρ controls the robustness level—larger ρ means protecting against more severe distribution shifts but potentially sacrificing nominal performance.

  Concrete Example: For image classification with χ^2-divergence and $\rho = 0.1$, if the empirical distribution assigns probability p_i to each training example, the uncertainty set includes reweighted distributions where example weights q_i satisfy $\sum_i (q_i - p_i)^2 / p_i \leq 0.1$. This allows the adversary to upweight or downweight training examples (modeling class imbalance or selection bias) but constrains the total reweighting. Training with DRO over this set makes the model robust to test distributions with similar imbalances.

- **Wasserstein ball**: $\mathcal{U} = \{Q : W_p(Q, P) \leq \rho\}$

  Detailed Explanation: The Wasserstein distance measures the minimal "cost" of transporting probability mass from distribution P to Q, where cost is determined by the underlying metric on the input space $\mathcal{X}$. For order $p = 1$ or $p = 2$, $W_p(Q, P)$ quantifies how much the data points must be moved (in expectation) to transform P into Q. The Wasserstein ball with radius ρ contains all distributions reachable by moving each data point by at most ρ

in the input space metric. This is particularly natural for image data where small pixel perturbations correspond to meaningful Wasserstein distance.

Concrete Example: For MNIST digit classification with ℓ_2 metric and $\rho = 0.3$ (normalized pixel space), the Wasserstein ball around a training image of "7" includes all images obtainable by moving each pixel value by small amounts such that the total ℓ_2 displacement is at most 0.3. This naturally models adversarial perturbations, brightness/contrast changes, and small geometric distortions. Wasserstein DRO with this uncertainty set automatically learns features robust to these perturbations without explicit adversarial training, and it has theoretical connections to adversarial training with ℓ_2 attacks.

- **Group DRO**: $\mathcal{U} = \text{conv}(\{P_1, \ldots, P_k\})$ over predefined groups

  Detailed Explanation: Group DRO partitions the data into k predefined groups (e.g., demographic subgroups, domains, or difficulty strata) with empirical distributions $P_1, \ldots, P_k$. The uncertainty set is the convex hull of these group distributions, meaning the adversary can choose any weighted mixture $Q = \sum_{i=1}^k \lambda_i P_i$ with $\lambda_i \geq 0, \sum_i \lambda_i = 1$. Unlike f-divergence or Wasserstein balls which are geometric, Group DRO is combinatorial—it explicitly optimizes for the worst-case group. The minimax optimization becomes $\min_\theta \max_{\lambda \in \Delta_k} \sum_{i=1}^k \lambda_i \mathbb{E}_{P_i}[\ell_\theta]$, which has a particularly clean solution: always upweight the group with the highest current loss.

  Concrete Example: In medical diagnosis with patient groups defined by age (pediatric, adult, elderly) and metabolic status (type A, type B), yielding 6 groups total, Group DRO ensures the model performs well on the worst-performing subgroup. If the model initially achieves 95% accuracy on adults with type A status but only 85% on elderly with type B status, Group DRO will upweight the latter group during training, forcing the model to improve performance on this specific subgroup. This is crucial for healthcare applications where disparate performance across subgroups violates safety requirements. Unlike vanilla ERM which optimizes average performance, Group DRO provides robustness guarantees for every predefined group.

Theorem 4.15 (DRO Duality)**.** *For χ^2-divergence uncertainty set, the robust optimization has the dual form:*

$$\min_\theta \sup_{Q: D_{\chi^2}(Q\|P) \leq \rho} \mathbb{E}_Q[\ell] = \min_\theta \left\{ \mathbb{E}_P[\ell] + \sqrt{\rho \cdot Var_P[\ell]} \right\} \tag{83}$$

This shows DRO penalizes the variance of the loss, preferring solutions that perform uniformly well.

5 Mathematical Foundations of Algorithmic Fairness

[20, 25, 60]

5.1 The Philosophical and Mathematical Underpinnings

Algorithmic fairness is concerned with ensuring that machine learning models do not reproduce or exacerbate existing social inequalities. Unlike accuracy, which is a single scalar metric, fairness is a multi-dimensional concept with mutually incompatible definitions rooted in different philosophical traditions of justice. This complexity stems from the fundamental question: What does it mean for an algorithm to be "fair"?

The philosophical foundations of fairness in AI draw on multiple traditions of justice theory. *Rawlsian justice* emphasizes that inequalities should benefit the least advantaged members of society, suggesting that AI systems should be evaluated in terms of their impacts on specific subgroups rather than merely on aggregate outcomes. *Luck egalitarianism* distinguishes between inequalities arising from circumstances beyond individual control (morally arbitrary) and those arising from genuine choice (potentially acceptable). *Anti-subordination theories* focus on disrupting patterns of group-based disparities rather than merely achieving statistical parity.

These philosophical traditions translate into distinct mathematical formulations of fairness, each capturing different intuitions about what fair treatment means. The diversity of these definitions is not a bug but a feature—it reflects the genuine complexity of fairness as a normative concept. However, this diversity creates practical challenges, as impossibility theorems demonstrate that many intuitive fairness properties cannot be simultaneously satisfied.

5.2 Group Fairness: Statistical Parity and Beyond

Group fairness requires that the system's behavior be similar across different demographic groups. Consider a binary classification setting with prediction $\hat{Y} \in \{0, 1\}$, true outcome $Y \in \{0, 1\}$, and protected attribute $A \in \{0, 1\}$ (extended naturally to multi-group settings).

Definition 5.1 (Demographic Parity)**.** A classifier satisfies demographic parity (also called statistical parity) if:

$$P(\hat{Y} = 1|A = 0) = P(\hat{Y} = 1|A = 1) \tag{84}$$

where $\hat{Y}$ is the predicted label and A is the sensitive attribute.

Demographic parity requires that positive prediction rates be equal across groups regardless of other characteristics. This definition ensures that members of different groups receive positive outcomes at equal rates but does not account for whether group members are equally qualified.

Example 5.1 (Demographic Parity in Clinical Trial Admission)**.** Consider a clinical trial admission algorithm where $\hat{Y} = 1$ denotes "accepted to trial" and $A \in \{0, 1\}$ represents two groups of patients (e.g., patients from rural vs. urban clinics). Suppose 500 rural applicants and 500 urban applicants are evaluated. Demographic parity requires that if 30% of urban applicants are accepted to the trial, then 30% of rural applicants must also be accepted, regardless of differences in baseline severity metrics. If the algorithm accepts 40% of urban and 20% of rural applicants, then $|P(\hat{Y} = 1|A = 0) - P(\hat{Y} = 1|A = 1)| = |0.20 - 0.40| = 0.20$, a violation of demographic parity.

Definition 5.2 (Equalized Odds)**.** A classifier satisfies equalized odds if both true positive and false positive rates are equal across groups:

$$P(\hat{Y} = 1|Y = y, A = 0) = P(\hat{Y} = 1|Y = y, A = 1) \quad \text{for } y \in \{0, 1\} \tag{85}$$

where Y is the true label.

Equalized odds requires that error rates be the same across groups, conditioning on the true outcome. This is often considered more appropriate than demographic parity because it accounts for different qualification rates across groups.

Example 5.2 (Equalized Odds in Disease Screening)**.** Consider an AI screening tool for a metabolic condition where $Y = 1$ means "truly has the condition" and A denotes patient group. Equalized odds requires: (1) among patients who truly have the condition ($Y = 1$), the detection rate must be equal across groups, e.g., if the tool detects 90% of cases in group $A = 0$, it must also detect 90% in group $A = 1$ (equal True Positive Rate); and (2) among healthy patients ($Y = 0$), the false alarm rate must be equal, e.g., if 5% of healthy patients in group $A = 0$ receive a false positive, the same must hold for group $A = 1$. A violation would be if the tool detects 90% of cases in group $A = 0$ but only 70% in group $A = 1$, meaning one group's patients are systematically under-diagnosed.

Definition 5.3 (Equal Opportunity)**.** A classifier satisfies equal opportunity if true positive rates are equal across groups:

$$P(\hat{Y} = 1|Y = 1, A = 0) = P(\hat{Y} = 1|Y = 1, A = 1) \tag{86}$$

where the condition $Y = 1$ restricts the comparison to the qualified population.

Equal opportunity is a relaxation of equalized odds that focuses only on ensuring equal treatment of qualified individuals.

Definition 5.4 (Calibration)**.** A classifier is calibrated if predicted probabilities equal actual outcome frequencies within each group:

$$P(Y = 1|\hat{Y} = p, A = a) = p \quad \text{for all } a \in \{0, 1\}, p \in [0, 1] \tag{87}$$

where p is the predicted probability score.

Calibration ensures that risk estimates are equally accurate across groups, meaning that individuals with the same predicted score should have the same actual risk regardless of group membership.

5.3 Individual Fairness: The Metric Approach

Individual fairness, formalized by [23], requires that similar individuals receive similar treatment (see also [28] for the equal opportunity criterion). The formal requirement is:

Definition 5.5 (Individual Fairness)**.** A classifier f satisfies individual fairness with respect to metrics $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$ if:

$$d_{\mathcal{Y}}(f(x), f(x')) \leq L \cdot d_{\mathcal{X}}(x, x') \tag{88}$$

for all $x, x' \in \mathcal{X}$, where L is a Lipschitz constant, $d_{\mathcal{X}}$ is a distance metric in the input space, and $d_{\mathcal{Y}}$ is a distance metric in the output space.

For probabilistic classifiers, the output metric is typically:

$$d_{\mathcal{Y}}(f(x), f(x')) = |P(\hat{Y} = 1|X = x) - P(\hat{Y} = 1|X = x')| \tag{89}$$

where $P(\hat{Y} = 1|X = x)$ is the predicted probability of the positive class for input x.

The central challenge in individual fairness is specifying the similarity metric $d_{\mathcal{X}}$. This requires determining which features are relevant for similarity comparisons and how to weight different features.

Example 5.3 (Individual Fairness in a Healthcare Triage System)**.** Consider a hospital triage system recommending urgent care priority ($\hat{Y} = 1$ means "urgent priority"). Two patients x and x' have similar medical histories: both reported a severe 8/10 headache, have a family history of migraines, and have similar age and comorbidities. Their profiles differ only in name and reported residential zip code. Individual fairness with Lipschitz constant $L = 1$ requires $|P(\hat{Y} = 1|x) - P(\hat{Y} = 1|x')| \leq d_{\mathcal{X}}(x, x')$. If the metric $d_{\mathcal{X}}$ uses only medically-relevant features (symptoms, history), then $d_{\mathcal{X}}(x, x') \approx 0.05$ (very similar patients), so the urgent priority probabilities must differ by at most 5%. The key design challenge is defining $d_{\mathcal{X}}$: should "past medical spending" be a relevant feature (which could capture past thoroughness of care) or an irrelevant one (including it might create economic discrimination)?

5.4 Impossibility Results and Trade-offs

A fundamental challenge in achieving fairness is that many mathematically precise fairness definitions are mutually incompatible [37]. The following theorems demonstrate that perfect fairness along multiple dimensions is impossible when groups have different base rates.

Theorem 5.1 (Impossibility of Perfect Fairness)**.** *If $P(Y = 1|A = 0) \neq P(Y = 1|A = 1)$ (different base rates), then no classifier can simultaneously satisfy:*

1. *Demographic Parity: $P(\hat{Y} = 1|A = 0) = P(\hat{Y} = 1|A = 1)$*
2. *Equalized Odds: $P(\hat{Y} = 1|Y = y, A = 0) = P(\hat{Y} = 1|Y = y, A = 1)$ for $y \in \{0, 1\}$*
3. *Calibration: $P(Y = 1|\hat{Y} = 1, A = a)$ is independent of a*

except in degenerate cases (e.g., constant predictors or perfect prediction).

Proof. We prove that Demographic Parity and Equalized Odds are incompatible when base rates differ ($p_0 \neq p_1$) and the predictor is not trivial (perfect or random).

Let $p_a = P(Y = 1|A = a)$ be the base rate for group a. Let $r_a = P(\hat{Y} = 1|Y = 1, A = a)$ be the True Positive Rate (TPR). Let $s_a = P(\hat{Y} = 1|Y = 0, A = a)$ be the False Positive Rate (FPR).

The overall positive prediction rate for group a is:

$$P(\hat{Y} = 1|A = a) = r_a p_a + s_a(1 - p_a) \tag{90}$$

Assume Equalized Odds: $r_0 = r_1 = r$ and $s_0 = s_1 = s$. Then the positive prediction rates are:

$$P(\hat{Y} = 1|A = 0) = rp_0 + s(1 - p_0) = s + (r - s)p_0 \tag{91}$$

$$P(\hat{Y} = 1|A = 1) = rp_1 + s(1 - p_1) = s + (r - s)p_1 \tag{92}$$

Assume Demographic Parity: $P(\hat{Y} = 1|A = 0) = P(\hat{Y} = 1|A = 1)$. Substituting the expressions above:

$$s + (r - s)p_0 = s + (r - s)p_1 \implies (r - s)(p_0 - p_1) = 0 \tag{93}$$

Since we assume different base rates ($p_0 \neq p_1$), we must have $r = s$. The condition $TPR = FPR$ implies that the classifier has no predictive power (it is equivalent to random guessing). Thus, a non-trivial classifier cannot simultaneously satisfy Demographic Parity and Equalized Odds when base rates differ. Since it cannot satisfy the first two, it certainly cannot satisfy all three. □

Theorem 5.2 (Chouldechova's Impossibility Result [17])**.** *For a binary classifier with different base rates across groups ($P(Y = 1|A = 0) \neq P(Y = 1|A = 1)$), it is impossible to simultaneously satisfy:*

1. *Equal positive predictive value:* $P(Y = 1|\hat{Y} = 1, A = 0) = P(Y = 1|\hat{Y} = 1, A = 1)$
2. *Equal false negative rate:* $P(\hat{Y} = 0|Y = 1, A = 0) = P(\hat{Y} = 0|Y = 1, A = 1)$
3. *Equal false positive rate:* $P(\hat{Y} = 1|Y = 0, A = 0) = P(\hat{Y} = 1|Y = 0, A = 1)$

unless the classifier is perfect (error-free) on both groups.

Proof Sketch. The positive predictive value (precision) is:

$$\text{PPV}_a = P(Y = 1|\hat{Y} = 1, A = a) = \frac{\text{TPR}_a \cdot p_a}{\text{TPR}_a \cdot p_a + \text{FPR}_a \cdot (1 - p_a)} \tag{94}$$

If $\text{TPR}_0 = \text{TPR}_1$ (equal FNR) and $\text{FPR}_0 = \text{FPR}_1$, then:

$$\text{PPV}_0 = \text{PPV}_1 \iff \frac{p_0}{p_0 + \text{FPR}/\text{TPR} \cdot (1 - p_0)} = \frac{p_1}{p_1 + \text{FPR}/\text{TPR} \cdot (1 - p_1)} \tag{95}$$

This equality holds for all FPR/TPR ratios only if $p_0 = p_1$, contradicting the assumption. Thus, when base rates differ, equal error rates imply unequal predictive values. □

This impossibility result has profound implications: it demonstrates that choices among fairness criteria cannot be avoided, and that claims to satisfy "fairness" without specifying which fairness criterion are incomplete.

Example 5.4 (Impossibility in Practice: Student Admission)**.** Consider a clinical trial admission classifier with two applicant groups having different base rates of medical condition suitability: $p_0 = P(Y = 1|A = 0) = 0.3$ and $p_1 = P(Y = 1|A = 1) = 0.6$. If we enforce equalized odds with $\text{TPR} = r = 0.8$ and $\text{FPR} = s = 0.1$, then the admission rates are: group $A = 0$: $r \cdot p_0 + s(1-p_0) = 0.8 \times 0.3 + 0.1 \times 0.7 = 0.31$; group $A = 1$: $r \cdot p_1 + s(1 - p_1) = 0.8 \times 0.6 + 0.1 \times 0.4 = 0.52$. Thus $P(\hat{Y} = 1|A = 0) = 0.31 \neq 0.52 = P(\hat{Y} = 1|A = 1)$. Demographic parity is violated. To force equal admission rates of, say, 0.40 for both groups, we would need $r_0 \neq r_1$ or $s_0 \neq s_1$, violating equalized odds. The trial organizers must choose: equal error rates (equalized odds) or equal admission rates (demographic parity) and they cannot have both.

5.5 Intersectionality and Multi-Group Fairness

Real-world fairness requires consideration of intersectional identities—individuals belong to multiple groups defined by different attributes. For attributes $A_1 \in \{0, 1\}$ and $A_2 \in \{0, 1\}$, there are four intersectional groups: $\{(0, 0), (0, 1), (1, 0), (1, 1)\}$.

Definition 5.6 (Intersectional Demographic Parity)**.** A classifier satisfies intersectional demographic parity if:

$$P(\hat{Y} = 1|A_1 = a_1, A_2 = a_2) = P(\hat{Y} = 1|A_1 = a_1', A_2 = a_2') \tag{96}$$

for all values a_1, a_1', a_2, a_2' of the protected attributes A_1 and A_2.

The computational complexity of fairness constraints grows exponentially with the number of protected attributes and their cardinalities. For k binary attributes, there are 2^k intersectional groups, making exact fairness intractable for large k.

5.6 Fairness Metrics and Evaluation

Translating abstract fairness definitions into measurable quantities requires comprehensive evaluation frameworks that account for statistical uncertainty and temporal dynamics.

5.6.1 Disparity Metrics

Definition 5.7 (Demographic Parity Difference (DPD))**.**

$$\text{DPD} = |P(\hat{Y} = 1|A = 0) - P(\hat{Y} = 1|A = 1)| \tag{97}$$

where A is the sensitive attribute. Values closer to zero indicate greater demographic parity.

Definition 5.8 (Disparate Impact Ratio (DIR))**.**

$$\text{DIR} = \frac{P(\hat{Y} = 1|A = 0)}{P(\hat{Y} = 1|A = 1)} \tag{98}$$

where the numerator is the selection rate for the disadvantaged group and the denominator is the selection rate for the advantaged group. The 80% rule considers DIR < 0.8 as prima facie evidence of adverse impact.

Definition 5.9 (Equalized Odds Difference (EOD))**.**

$$\text{EOD} = \frac{1}{2}\left(|\text{TPR}_{A=0} - \text{TPR}_{A=1}| + |\text{FPR}_{A=0} - \text{FPR}_{A=1}|\right) \tag{99}$$

where $\text{TPR}_{A=a} = P(\hat{Y} = 1|Y = 1, A = a)$ is the True Positive Rate for group a, and $\text{FPR}_{A=a} = P(\hat{Y} = 1|Y = 0, A = a)$ is the False Positive Rate for group a.

5.6.2 Calibration Metrics

Definition 5.10 (Expected Calibration Error (ECE))**.**

$$\text{ECE} = \sum_{b=1}^{B} \frac{n_b}{n} |\text{acc}_b - \text{conf}_b| \tag{100}$$

where predictions are partitioned into B bins by confidence level, n_b is the number of samples in bin b, n is the total number of samples, acc_b is the accuracy in bin b, and conf_b is the average confidence in bin b.

For fairness assessment, ECE should be computed separately for each protected group to reveal calibration disparities.

5.6.3 Statistical Considerations

Fairness metrics are estimates from finite samples and subject to statistical uncertainty. Confidence intervals should accompany point estimates:

$$\widehat{\text{DPD}} \pm z_{\alpha/2} \cdot \text{SE}(\widehat{\text{DPD}}) \tag{101}$$

where $\widehat{\text{DPD}}$ is the estimated demographic parity difference, $z_{\alpha/2}$ is the critical value from the standard normal distribution for significance level α, and SE denotes the standard error.

For binary outcomes, the standard error of the difference in selection rates is:

$$\mathrm{SE}(\hat{p}_1 - \hat{p}_0) = \sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_0(1-\hat{p}_0)}{n_0}} \tag{102}$$

where $\hat{p}_a$ is the sample selection rate for group a, and n_a is the sample size for group a.

5.7 Algorithmic Techniques for Fairness

Recent advancements in trustworthy AI have formalized three primary intervention points for fairness: pre-processing, in-processing, and post-processing [57, 75].

5.7.1 Pre-processing Methods

Pre-processing methods fundamentally alter the training data distribution to remove statistical dependencies between sensitive attributes and outcomes before a model is ever trained. The most prominent technique, **Reweighing**, assigns importance weights to individual samples to enforce independence. Specifically, for an individual i with sensitive attribute A_i and outcome Y_i, the weight is calculated as:

$$w_i = \frac{P(A = A_i)P(Y = Y_i)}{P(A = A_i, Y = Y_i)} \tag{103}$$

which effectively constructs a new dataset where A and Y are independent. Alternatively, **Fair Representation Learning** seeks to learn a projection of the data into a latent space Z where sensitive information is obfuscated while task-relevant information is preserved. This is typically formulated as an optimization problem $\min_Z \mathcal{L}_{\text{reconstruction}} + \lambda \mathcal{L}_{\text{fairness}}$, where the fairness penalty enforces orthogonality or independence between Z and A [16].

5.7.2 In-processing Methods

In-processing methods integrate fairness requirements directly into the model training procedure, often by solving a constrained optimization problem. The general formulation involves minimizing the empirical risk $\mathcal{L}(\theta)$ subject to a constraint $g_{\text{fairness}}(\theta) \leq \epsilon$ that bounds the allowable fairness violation. Techniques like **Adversarial Debiasing** employ a minimax game where a predictor minimizes the task loss while an adversary attempts to reconstruct the sensitive attribute from the predictor's outputs or latent representations:

$$\min_{\theta_f} \max_{\theta_a} \mathcal{L}_{\text{task}}(\theta_f) - \lambda \mathcal{L}_{\text{adversarial}}(\theta_f, \theta_a) \tag{104}$$

This approach ensures that the learned representation contains no extractable information about the protected group. To solve these constrained problems, **Lagrangian Relaxation** is frequently used, converting the hard constraints into a primal-dual objective $\min_\theta \max_{\lambda \geq 0} \mathcal{L}(\theta) + \lambda g_{\text{fairness}}(\theta)$, which can be optimized via alternating gradient descent integration principles from [31]. The "Reductions Approach" [5] further simplifies this constrained optimization by reducing it to a sequence of cost-sensitive classification problems.

5.7.3 Post-processing Methods

Adjust model outputs to satisfy fairness criteria.

Threshold Optimization: Sets group-specific thresholds to achieve fairness:

$$\hat{Y}_a = \mathbb{I}[f(X) > \tau_a] \quad \text{where } \tau_a \text{ chosen to satisfy fairness constraints} \tag{105}$$

where $\mathbb{I}[\cdot]$ is the indicator function, $f(X)$ is the model score, and τ_a is the group-specific threshold for group a.

Calibration Adjustment: Modifies predicted probabilities to ensure equal calibration across groups.

5.8 Causal Fairness and Counterfactual Reasoning

Causal approaches to fairness seek to identify and eliminate the causal effects of protected attributes on outcomes, distinguishing legitimate from illegitimate causal pathways. Unlike statistical fairness, which focuses on correlations, causal fairness requires that decisions are fair under interventions on the causal graph.

5.8.1 Motivation: Why Causality Matters for Fairness

Statistical fairness criteria can be satisfied by systems that are intuitively unfair, or violated by systems that are intuitively fair. Consider a hiring algorithm:

- A company uses education level to predict job performance.
- Education is correlated with race due to historical discrimination in educational access.
- A statistically fair classifier (satisfying demographic parity) might reject qualified candidates from a specific demographic to balance rates.
- A causal approach asks: "Would this person have been hired if their race had been different, holding their qualifications fixed?"

5.8.2 Counterfactual Fairness

Definition 5.11 (Counterfactual Fairness [38])**.** Let $\mathcal{M}$ be a structural causal model with observed variables X, sensitive attribute A, and exogenous variables U. A predictor $\hat{Y}$ is **counterfactually fair** if for any individual with characteristics $(X = x, A = a)$ and corresponding exogenous factors $U = u$:

$$P(\hat{Y}_{A\leftarrow a}(u) = y \mid X = x, A = a) = P(\hat{Y}_{A\leftarrow a'}(u) = y \mid X = x, A = a) \tag{106}$$

for all values $a, a' \in \mathcal{A}$ and all outcomes y.

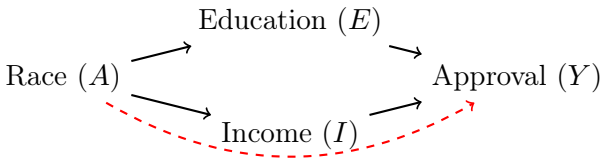
Here, $\hat{Y}_{A\leftarrow a}(u)$ denotes the counterfactual: "What would the prediction have been if A had been set to a, keeping the same individual (same U)?" The notation $A \leftarrow a$ represents the do-operator or intervention setting A to value a.

Remark 5.1 (Interpretation)**.** Counterfactual fairness ensures that an individual's classification would not have changed if their sensitive attribute had been different, holding all non-descendant factors constant. This captures the intuition that a person should not be penalized for characteristics beyond their control.

5.8.3 Path-Specific Fairness

In many real-world scenarios, the effect of a sensitive attribute on the outcome may flow through multiple causal pathways, some of which may be considered legitimate and others discriminatory.

Example 5.5 (Legitimate vs. Illegitimate Pathways)**.** Consider a treatment approval model with the causal graph:



- **Direct effect** ($A \to Y$, dashed): Direct discrimination: illegitimate.
- **Indirect through income** ($A \to I \to Y$): May reflect historical wage discrimination: potentially illegitimate.
- **Indirect through education** ($A \to E \to Y$): Complex: education differences may reflect both discrimination in access and genuine qualification differences.

Definition 5.12 (Path-Specific Counterfactual Fairness)**.** A predictor is fair with respect to a set of causal paths Π if the causal effect of the sensitive attribute A on prediction $\hat{Y}$ *along paths in* Π is zero:

$$\text{PSE}_{\Pi}(a \to a') = \mathbb{E}[\hat{Y}_{A \leftarrow a', \Pi}(u) - \hat{Y}_{A \leftarrow a, \Pi}(u)] = 0 \tag{107}$$

where the intervention affects only the designated pathways Π, and PSE stands for Path-Specific Effect.

5.8.4 Implementing Causal Fairness

To implement causal fairness in practice:

Step 1: Specify the Causal Graph. This requires domain expertise and may be contested. The graph encodes assumptions about how variables causally influence each other.

Step 2: Identify Fair Pathways. Determine which causal effects of the sensitive attribute are legitimate (resolving pathway) vs. illegitimate (discriminatory pathway).

Step 3: Compute Counterfactual Features. For each individual, compute what their features would have been under a different sensitive attribute value:

$$X_{A \leftarrow a'}(u) = f_X(\text{Pa}_X \text{ with } A \text{ set to } a', U_X = u_X) \tag{108}$$

where f_X is the structural equation for X, Pa_X are the parents of X in the causal graph, and U_X are the exogenous noise variables for X.

Step 4: Train Fair Predictor. Train a model using only:

- Features that are not descendants of A in the causal graph, or
- Counterfactually-adjusted features that remove the effect of A

5.8.5 The Proxy Variable Problem

Even when the sensitive attribute A is not directly used, the model may learn to predict A from other features (proxies), leading to indirect discrimination.

Definition 5.13 (Proxy Variable)**.** A variable X is a **proxy** for sensitive attribute A if:

1. X is correlated with A: $I(X; A) > 0$
2. Using X for prediction leads to disparate outcomes: $P(\hat{Y}|A = 0, X) \neq P(\hat{Y}|A = 1, X)$ in expectation

where $I(X; A)$ is the mutual information between X and A.

Example 5.6 (Common Proxies)**.**

- ZIP code as proxy for race (due to residential segregation)
- First name as proxy for gender or ethnicity
- LinkedIn connections as proxy for socioeconomic status

Causal analysis helps identify and address proxies by examining the causal structure. If $A \to X \to \hat{Y}$, then X transmits the effect of A to the prediction.

5.8.6 Challenges and Limitations

1. **Causal Graph Uncertainty**: The true causal graph is often unknown. Different assumptions lead to different fairness conclusions.
2. **Unobserved Confounders**: Hidden common causes between A and Y can make causal effects unidentifiable from observational data.
3. **Counterfactual Identifiability**: Computing counterfactuals requires knowledge of the structural equations f_i, which are typically unknown.
4. **Philosophical Concerns**: Some argue that counterfactuals of immutable attributes ("if this person's race had been different") are metaphysically problematic.
5. **Accuracy Trade-offs**: If the sensitive attribute contains legitimate predictive information through fair pathways, counterfactual fairness may significantly reduce accuracy.

5.9 Fairness in Modern AI Systems

Contemporary challenges in fairness include:

Foundation Models: Pre-trained models may encode biases that propagate to downstream tasks, requiring fairness interventions at multiple stages.

Dynamic Systems: ML systems that learn and adapt over time can exhibit fairness drift, requiring continuous monitoring.

Multi-stakeholder Settings: Different stakeholders may prioritize different fairness criteria, requiring principled approaches to trade-off resolution.

Exercises

1. **Impossibility Results**:
   (a) Prove that demographic parity and individual fairness can be incompatible. Consider a simple example with two groups having different base rates and show that no Lipschitz-continuous classifier can satisfy both constraints.
   (b) Implement the Patient Health Outcomes dataset and demonstrate the trade-off between demographic parity and equalized odds by plotting the Pareto frontier.
   (c) Construct a concrete example where calibration and demographic parity cannot be simultaneously satisfied.
2. **Fairness Metrics Implementation**:
   (a) Implement a comprehensive fairness evaluation library that computes DPD, DIR, EOD, calibration metrics with confidence intervals.
   (b) Add support for intersectional analysis across multiple protected attributes.
   (c) Implement statistical significance tests for fairness disparities using permutation tests and bootstrap confidence intervals.
3. **Individual Fairness**:
   (a) Design a task-appropriate similarity metric for a generic dataset measuring creditworthiness that considers financial history while being insensitive to protected attributes.
   (b) Implement the Lipschitz regularization approach and compare it against standard ERM.
   (c) Investigate how the choice of similarity metric affects both fairness and accuracy.
4. **Algorithmic Interventions**:
   (a) Implement the Reweighing pre-processing algorithm and evaluate its effectiveness on reducing demographic parity violations.
   (b) Design an adversarial debiasing system using PyTorch that learns fair representations for image classification tasks.
   (c) Compare pre-processing, in-processing, and post-processing approaches on the same dataset and analyze their relative strengths and limitations.
5. **Causal Fairness**:
   (a) Construct a structural causal model for a hiring scenario with legitimate and illegitimate causal pathways.
   (b) Implement counterfactual fairness using causal inference techniques and compare it to statistical fairness measures.
   (c) Analyze the impact of hidden confounders on fairness assessment and mitigation.
6. **Advanced Applications**:
   (a) Design a fair recommendation system that balances accuracy and demographic parity across user groups.
   (b) Implement fairness-aware federated learning where different clients have different protected group distributions.

(c) Create a dynamic fairness monitoring system that detects and responds to fairness drift in deployed models.

7. **Theoretical Case Analysis**:

   (a) Analyze a synthetic dataset $\mathcal{D}$ generated from a known structural causal model. Compute all fairness metrics, identify disparities, and propose mitigations.

   (b) Examine the impact of base rate differences on fairness metrics using abstract distributions $\mathcal{P}_0$ and $\mathcal{P}_1$.

   (c) Study a hypothetical medical dataset: balance fairness constraints with the clinical need for accurate predictions.

8. **Intersectionality**:

   (a) Implement intersectional fairness metrics for combinations of race and gender.

   (b) Show that satisfying fairness for individual attributes does not guarantee fairness for intersectional groups.

   (c) Propose and implement a method to achieve fairness across exponentially many intersectional groups efficiently.

9. **Theoretical Analysis**:

   (a) Prove that for k protected groups, achieving exact demographic parity reduces to a system of $k-1$ linear constraints.

   (b) Derive the sample complexity required to estimate demographic parity difference to within $\pm\epsilon$ with probability $1-\delta$.

   (c) Analyze the computational complexity of finding an optimal fair classifier under different fairness constraints.

5.10 Implementation Templates and Code Examples

5.10.1 Fairness-Constrained Optimization

```python
import torch
import torch.nn as nn

class FairClassifier(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super().__init__()
        self.predictor = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, output_dim),
            nn.Sigmoid()
        )

    def forward(self, x):
        return self.predictor(x)

def demographic_parity_loss(y_pred, sensitive_attr):
    """Differentiable demographic parity constraint."""
    group_0_idx = (sensitive_attr == 0).nonzero(as_tuple=True)[0]
    group_1_idx = (sensitive_attr == 1).nonzero(as_tuple=True)[0]

```

```
    if len(group_0_idx) == 0 or len(group_1_idx) == 0:
        return torch.tensor(0.0)

    rate_0 = y_pred[group_0_idx].mean()
    rate_1 = y_pred[group_1_idx].mean()

    return (rate_0 - rate_1) ** 2

def train_fair_classifier(model, train_loader, epochs=100,
    lambda_fair=1.0):
    """Training loop with fairness constraint."""
    optimizer = torch.optim.Adam(model.parameters())
    criterion = nn.BCELoss()

    for epoch in range(epochs):
        for batch_x, batch_y, batch_a in train_loader:
            optimizer.zero_grad()
            y_pred = model(batch_x)

            # Task loss + Fairness loss
            task_loss = criterion(y_pred.squeeze(), batch_y.float())
            fair_loss = demographic_parity_loss(y_pred.squeeze(),
                batch_a)
            total_loss = task_loss + lambda_fair * fair_loss

            total_loss.backward()
            optimizer.step()
```

Listing 1: Fair Classifier with Demographic Parity Constraint

5.10.2 Adversarial Debiasing Implementation

```
class AdversarialDebiasing(nn.Module):
    def __init__(self, input_dim, hidden_dim, repr_dim):
        super().__init__()

        # Feature encoder: learns fair representations
        self.encoder = nn.Sequential(
            nn.Linear(input_dim, hidden_dim),
            nn.ReLU(),
            nn.Linear(hidden_dim, repr_dim)
        )

        # Task predictor: predicts target label from representation
        self.predictor = nn.Sequential(
            nn.Linear(repr_dim, 1),
            nn.Sigmoid()
        )

        # Adversary: tries to predict sensitive attribute
        self.discriminator = nn.Sequential(
            nn.Linear(repr_dim, hidden_dim // 2),
            nn.ReLU(),
            nn.Linear(hidden_dim // 2, 1),
            nn.Sigmoid()
        )

```

```
    def forward(self, x):
        z = self.encoder(x)          # Fair representation
        y_pred = self.predictor(z)    # Task prediction
        a_pred = self.discriminator(z)  # Adversary prediction
        return y_pred, a_pred, z
```

Listing 2: Adversarial Debiasing Architecture

5.10.3 Comprehensive Fairness Evaluation

```
import numpy as np
from scipy import stats

class FairnessEvaluator:
    def __init__(self, y_true, y_pred, y_prob, sensitive_attr):
        self.y_true = y_true
        self.y_pred = y_pred
        self.y_prob = y_prob
        self.sensitive_attr = sensitive_attr

    def demographic_parity_difference(self):
        """Calculate DPD with 95% confidence interval."""
        rate_0 = self.y_pred[self.sensitive_attr == 0].mean()
        rate_1 = self.y_pred[self.sensitive_attr == 1].mean()
        n_0 = (self.sensitive_attr == 0).sum()
        n_1 = (self.sensitive_attr == 1).sum()

        dpd = rate_1 - rate_0
        se = np.sqrt(rate_0*(1-rate_0)/n_0 + rate_1*(1-rate_1)/n_1)

        return {
            'dpd': dpd,
            'ci_lower': dpd - 1.96 * se,
            'ci_upper': dpd + 1.96 * se,
            'p_value': 2 * (1 - stats.norm.cdf(abs(dpd / se)))
        }

    def comprehensive_report(self):
        """Generate comprehensive fairness report."""
        report = {
            'demographic_parity': self.demographic_parity_difference(),
            'equalized_odds': self.equalized_odds_difference(),
            'calibration': self.calibration_analysis()
        }

        # Summary with threshold-based violation detection
        dpd_violation = abs(report['demographic_parity']['dpd']) > 0.1
        report['summary'] = {
            'demographic_parity_violation': dpd_violation,
            'overall_fair': not dpd_violation
        }
        return report
```

Listing 3: Comprehensive Fairness Evaluation Class

6 Privacy and Data Sovereignty

[21, 39, 51]

6.1 Differential Privacy Foundations

The gold standard for privacy-preserving data analysis is Differential Privacy (DP). Unlike anonymity, which focuses on removing identifiers (and is often reversible via linkage attacks), DP focuses on the stability of the algorithm's output.

6.1.1 Definition and Properties

Definition 6.1 ((ϵ, δ)-Differential Privacy)**.** A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathcal{R}$ is (ϵ, δ)-DP if for all adjacent datasets D, D' (differing by exactly one element) and all measurable subsets $S \subseteq \mathcal{R}$:

$$P(\mathcal{M}(D) \in S) \leq e^{\epsilon} P(\mathcal{M}(D') \in S) + \delta \tag{109}$$

where P denotes probability, ϵ is the privacy budget, and δ is the failure probability.

The parameter ϵ (privacy budget) bounds the multiplicative difference in probabilities. A smaller ϵ implies stronger privacy (outputs are indistinguishable). The parameter δ allows for a small probability of failure (e.g., the mechanism accidentally outputting the training data). Typically, we require $\delta \ll 1/n$.

Example 6.1 (Differential Privacy in Hospital Data Analysis)**.** A hospital database D contains $n = 10{,}000$ patient records. A researcher queries "What fraction of patients have diabetes?" If adding or removing any single patient's record cannot change this answer by more than a factor of e^{ϵ}, the analysis is ϵ-DP. With $\epsilon = 0.1$, the probability ratio $P(\mathcal{M}(D) \in S)/P(\mathcal{M}(D') \in S) \leq e^{0.1} \approx 1.105$ for any output set S. This means that whether or not any specific patient is in the database, the output distribution changes by at most 10.5%, too small for an adversary to confidently infer any individual's presence. The patient can share their data knowing the query result would be nearly identical without them.

6.1.2 The Privacy Loss Random Variable

To analyze complex mechanisms, it is useful to define the privacy loss random variable Z for an outcome $o \sim \mathcal{M}(D)$:

$$Z \coloneqq \ln \frac{P(\mathcal{M}(D) = o)}{P(\mathcal{M}(D') = o)} \tag{110}$$

where $P(\mathcal{M}(D) = o)$ is the probability density of outcome o under dataset D. (ϵ, δ)-DP ensures that $P(Z > \epsilon) \leq \delta$.

6.2 Mechanisms for Differential Privacy

6.2.1 The Laplace Mechanism

For a function $f : \mathcal{D} \to \mathbb{R}^k$, the L_1-sensitivity is $\Delta_1 f \coloneqq \max_{D,D'} \|f(D) - f(D')\|_1$. The Laplace Mechanism adds noise drawn from the Laplace distribution:

$$\mathcal{M}(D) = f(D) + (Y_1, \ldots, Y_k) \quad \text{where} \quad Y_i \sim \text{Lap}(\Delta_1 f/\epsilon) \tag{111}$$

where $\text{Lap}(b)$ denotes the Laplace distribution with scale parameter b, and $\Delta_1 f$ is the L_1-sensitivity of function f.

Proof. The density of the Laplace distribution with scale b is $h(y) = \frac{1}{2b}\exp(-|y|/b)$. For output $z \in \mathbb{R}^k$:

$$\frac{P(\mathcal{M}(D) = z)}{P(\mathcal{M}(D') = z)} = \frac{\prod_{i=1}^{k} \exp(-|z_i - f(D)_i|/b)}{\prod_{i=1}^{k} \exp(-|z_i - f(D')_i|/b)} \tag{112}$$

$$= \exp\left(\frac{1}{b}\sum_{i=1}^{k}(|z_i - f(D')_i| - |z_i - f(D)_i|)\right) \tag{113}$$

By the reverse triangle inequality, for any $z_i, a, b \in \mathbb{R}$:

$$|z_i - b| - |z_i - a| \leq |a - b| \tag{114}$$

Summing over all coordinates:

$$\sum_{i=1}^{k}(|z_i - f(D')_i| - |z_i - f(D)_i|) \leq \sum_{i=1}^{k}|f(D)_i - f(D')_i| = \|f(D) - f(D')\|_1 \leq \Delta_1 f \tag{115}$$

Setting $b = \Delta_1 f/\epsilon$ yields:

$$\frac{P(\mathcal{M}(D) = z)}{P(\mathcal{M}(D') = z)} \leq \exp\left(\frac{\Delta_1 f}{b}\right) = \exp(\epsilon) \tag{116}$$

This holds for all z, hence $\mathcal{M}$ is ϵ-DP (with $\delta = 0$). □

Example 6.2 (Laplace Mechanism: Average Blood Pressure)**.** Suppose we want to release the average systolic blood pressure of $n = 1{,}000$ patients, where each measurement lies in $[80, 200]$ mmHg. The query is $f(D) = \frac{1}{n}\sum_{i=1}^{n} x_i$. The sensitivity is $\Delta_1 f = \max_{D,D'} |f(D) - f(D')| = (200 - 80)/1000 = 0.12$ mmHg (changing one patient's record shifts the average by at most 0.12). For $\epsilon = 1$, we add noise $Y \sim \text{Lap}(0.12/1) = \text{Lap}(0.12)$, which has standard deviation $\sqrt{2} \times 0.12 \approx 0.17$ mmHg. If the true average is 125.3 mmHg, the released value might be $125.3 + 0.09 = 125.39$ mmHg, the noise is medically negligible (0.07% error) while providing strong privacy. For $\epsilon = 0.1$ (stronger privacy), the noise scale increases to $\text{Lap}(1.2)$ with standard deviation ≈ 1.7 mmHg, still clinically acceptable.

6.2.2 The Gaussian Mechanism

For functions with high L_1 sensitivity but low L_2 sensitivity, the Gaussian mechanism is preferred. $\mathcal{M}(D) = f(D) + \mathcal{N}(0, \sigma^2 I)$. This satisfies (ϵ, δ)-DP if $\sigma \geq \frac{\Delta_2 f}{\epsilon}\sqrt{2\ln(1.25/\delta)}$. The proof relies on the tail properties of the Gaussian distribution.

6.2.3 The Exponential Mechanism

For non-numeric queries (e.g., "What is the most common disease?"), we define a utility function $u(D, r)$ that scores each possible output $r \in \mathcal{R}$. The mechanism samples r with probability proportional to $\exp(\frac{\epsilon u(D,r)}{2\Delta u})$.

Example 6.3 (Exponential Mechanism: Selecting a Treatment Protocol)**.** A hospital wants to select the best treatment protocol from $\mathcal{R} = \{A, B, C\}$ based on patient outcomes, where the utility $u(D, r)$ counts how many patients recovered under protocol r. Suppose the counts are

$u(D, A) = 340$, $u(D, B) = 310$, $u(D, C) = 350$ out of 1,000 patients. Since changing one patient changes any count by at most 1, $\Delta u = 1$. With $\epsilon = 1$, the selection probabilities are proportional to $\exp(\epsilon \cdot u/(2 \cdot 1))$: protocol A: e^{170}, B: e^{155}, C: e^{175}. After normalization, protocol C is selected with probability ≈ 0.993, A with ≈ 0.007, and B with $\approx 10^{-9}$. The best protocol is very likely chosen while satisfying ϵ-DP, no individual patient's outcome significantly affects which protocol is selected.

6.3 Composition and Renyi Differential Privacy

A key property of DP is composition: the privacy guarantee holds even when multiple analyses are performed on the same data. **Basic Composition** states that if $\mathcal{M}_i$ is ϵ_i-DP, then the sequence $(\mathcal{M}_1(D), \ldots, \mathcal{M}_k(D))$ is $(\sum \epsilon_i)$-DP. **Advanced Composition** provides a tighter bound, showing that for k mechanisms each being ϵ-DP, the composition is roughly $(k\epsilon^2)$-DP. This quadratic improvement is crucial for iterative algorithms like SGD.

6.3.1 Renyi Differential Privacy (RDP)

RDP [47] provides tighter composition bounds by tracking the Renyi Divergence between the output distributions.

Definition 6.2 ((α, ϵ)-RDP)**.** A mechanism $\mathcal{M}$ is (α, ϵ)-RDP if for all adjacent D, D':

$$D_\alpha(\mathcal{M}(D) \| \mathcal{M}(D')) \leq \epsilon \tag{117}$$

where $D_\alpha(P\|Q) = \frac{1}{\alpha-1} \ln \mathbb{E}_{x\sim Q}[(P(x)/Q(x))^\alpha]$ is the Renyi divergence of order α.

RDP composes linearly. We can convert RDP back to (ϵ, δ)-DP at the end of the analysis.

6.4 Privacy in Deep Learning

6.4.1 DP-SGD Algorithm

Training neural networks with DP involves two modifications [1]: clipping per-sample gradients to bound sensitivity, and adding Gaussian noise to the aggregated gradient.

Algorithm 4 Differentially Private SGD (DP-SGD)

Input: Dataset D, Loss $\mathcal{L}$, Noise scale σ, Clip C, Learning rate η, Iterations T
for $t = 1$ to T **do**
  Sample batch B_t with sampling probability $q = L/N$
  for $i \in B_t$ **do**
    $g_t^{(i)} \leftarrow \nabla_{\theta_t} \mathcal{L}(x_i, y_i)$
    $\bar{g}_t^{(i)} \leftarrow g_t^{(i)} / \max(1, \frac{\|g_t^{(i)}\|_2}{C})$ {Per-sample Clipping}
  end for
  $\tilde{g}_t \leftarrow \frac{1}{|B_t|} \left(\sum_i \bar{g}_t^{(i)} + \mathcal{N}(0, \sigma^2 C^2 I) \right)$ {Add Noise}
  $\theta_{t+1} \leftarrow \theta_t - \eta \tilde{g}_t$
end for

The privacy cost is calculated using the "Moments Accountant" method, which tracks the RDP moments across iterations.

6.4.2 Privacy Amplification by Subsampling

A crucial property of DP-SGD is that the privacy guarantee is *amplified* by subsampling. If the batch is a random subsample of the dataset with sampling probability q, the privacy cost per iteration is reduced.

Theorem 6.1 (Privacy Amplification by Subsampling)**.** *If $\mathcal{M}$ is an (ϵ, δ)-DP mechanism and we apply it to a uniformly random subsample with probability q, the resulting mechanism $\mathcal{M}'$ satisfies (ϵ', δ')-DP where $\epsilon' = \ln(1 + q(e^\epsilon - 1))$ and $\delta' = q\delta$. For small ϵ, $\epsilon' \approx q\epsilon$.*

Proof Sketch. Let $\mathcal{M}'(D)$ denote applying $\mathcal{M}$ to a random subsample $S \subseteq D$ where each element is included independently with probability q. For adjacent datasets D, D' differing in one element x^*:

$$P(\mathcal{M}'(D) \in T) = \sum_S P(S \text{ sampled from } D) \cdot P(\mathcal{M}(S) \in T) \tag{118}$$

$$= q \cdot P(S \ni x^*) \cdot P(\mathcal{M}(S) \in T) + (1 - q) \cdot P(S \not\ni x^*) \cdot P(\mathcal{M}(S) \in T) \tag{119}$$

When $x^* \notin S$, the distributions from D and D' are identical. When $x^* \in S$ (with probability q), we use the (ϵ, δ)-DP guarantee of $\mathcal{M}$:

$$P(\mathcal{M}'(D) \in T) \leq (1 - q)P(\mathcal{M}'(D') \in T) + q(e^\epsilon P(\mathcal{M}'(D') \in T) + \delta) \tag{120}$$

$$= P(\mathcal{M}'(D') \in T) \cdot (1 - q + qe^\epsilon) + q\delta \tag{121}$$

$$= P(\mathcal{M}'(D') \in T) \cdot (1 + q(e^\epsilon - 1)) + q\delta \tag{122}$$

Thus we have $\epsilon' = \ln(1 + q(e^\epsilon - 1))$ and $\delta' = q\delta$. □

This amplification is critical for making DP-SGD practical: without it, the noise required for privacy would make training impossible.

6.4.3 The Moments Accountant

The Moments Accountant provides tight composition bounds for DP-SGD by tracking the moments of the privacy loss random variable Z. Define the λ-th moment as:

$$\alpha_{\mathcal{M}}(\lambda) = \max_{D \sim D'} \log \mathbb{E}_{o \sim \mathcal{M}(D)} \left[\exp(\lambda Z)\right] \tag{123}$$

where $Z = \log \frac{P(\mathcal{M}(D)=o)}{P(\mathcal{M}(D')=o)}$ is the privacy loss random variable, and the maximum is taken over all adjacent datasets D, D'.

The moments compose additively: for k applications of mechanism $\mathcal{M}$, $\alpha_{k \times \mathcal{M}}(\lambda) = k \cdot \alpha_{\mathcal{M}}(\lambda)$. After computing the composed moments, we convert back to (ϵ, δ)-DP using the Chernoff bound. By Markov's inequality:

$$P(Z > \epsilon) \leq e^{-\lambda\epsilon} \, \mathbb{E}[e^{\lambda Z}] = e^{-\lambda\epsilon + \alpha(\lambda)} \tag{124}$$

Setting this equal to δ and solving for ϵ:

$$\epsilon = \min_{\lambda > 1} \left\{ \frac{\alpha(\lambda) - \log(1/\delta)}{\lambda - 1} \right\} \tag{125}$$

The optimization over λ finds the tightest bound. In practice, we evaluate this for a range of λ values and take the minimum.

For the Gaussian mechanism with L_2 sensitivity Δ and noise scale σ, the moment can be computed as:

$$\alpha_G(\lambda) = \frac{\lambda \Delta^2}{2\sigma^2} \tag{126}$$

This formula shows that larger noise (σ) reduces the moment (better privacy), while larger sensitivity (Δ) increases it (worse privacy).

6.4.4 PATE (Private Aggregation of Teacher Ensembles)

PATE trains an ensemble of "teacher" models on disjoint partitions of the private data. To predict on a new input, the teachers vote, and the noisy argmax of the vote is returned. A "student" model is then trained on the public data labeled by the teachers. This avoids exposing the private model weights directly.

Algorithm 5 PATE: Student Training

Input: Teacher ensemble $\{f_1, \ldots, f_K\}$, Unlabeled public data D_{pub}, Noise scale σ
for $x \in D_{pub}$ **do**
  $n_c \leftarrow \sum_{k=1}^{K} \mathbb{I}(f_k(x) = c)$ for each class c {Teacher votes}
  $\tilde{n}_c \leftarrow n_c + \text{Lap}(\sigma)$ {Add noise}
  $\tilde{y} \leftarrow \arg\max_c \tilde{n}_c$ {Noisy aggregation}
  Add $(x, \tilde{y})$ to student training set
end for
Train student model on labeled data

The privacy guarantee of PATE depends on the consensus among teachers. When teachers agree, the noise has little effect on the label, and privacy cost is low. When teachers disagree, more privacy budget is consumed. The Confident-GNMAX aggregation mechanism formalizes this:

$$\tilde{y} = \begin{cases} \arg\max_c(n_c + \mathcal{N}(0, \sigma_2^2)) & \text{if } \max_c n_c + \mathcal{N}(0, \sigma_1^2) > T \\ \perp \text{ (abstain)} & \text{otherwise} \end{cases} \tag{127}$$

where n_c is the vote count for class c, σ_1, σ_2 are noise scales, and T is a threshold.

6.5 Secure Multi-Party Computation

Secure Multi-Party Computation (MPC) allows multiple parties to jointly compute a function over their private inputs without revealing those inputs to each other.

6.5.1 Problem Formulation

Consider n parties $P_1, \ldots, P_n$, each holding private input x_i. The goal is to compute $f(x_1, \ldots, x_n)$ such that two conditions are met: **Correctness**, meaning all parties learn the correct output $f(x_1, \ldots, x_n)$; and **Privacy**, meaning no party learns anything about others' inputs beyond what can be inferred from the output.

Definition 6.3 (Security Definition)**.** A protocol Π securely computes f if for every adversary $\mathcal{A}$ in the real protocol, there exists a simulator $\mathcal{S}$ in the ideal world such that:

$$\text{Real}_{\Pi,\mathcal{A}}(x_1, \ldots, x_n) \approx_c \text{Ideal}_{f,\mathcal{S}}(x_1, \ldots, x_n) \tag{128}$$

where $\approx_c$ denotes computational indistinguishability, meaning no polynomial-time algorithm can distinguish the two distributions with non-negligible probability.

6.5.2 Secret Sharing

Secret sharing is a fundamental building block for MPC. A (t, n)-threshold secret sharing scheme splits a secret s into n shares such that any t or more shares can reconstruct s, while any fewer than t shares reveal no information about s.

Shamir's Secret Sharing: To share secret $s \in \mathbb{F}_p$, choose random polynomial $q(x)$ of degree $t-1$ with $q(0) = s$:

$$q(x) = s + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \tag{129}$$

where s is the secret, and $a_1, \ldots, a_{t-1}$ are random coefficients from the field $\mathbb{F}_p$. Each party i receives share $s_i = q(i)$. Reconstruction uses Lagrange interpolation:

$$s = q(0) = \sum_{i \in S} s_i \cdot \prod_{j \in S, j \neq i} \frac{j}{j - i} \tag{130}$$

for any set S of t parties.

Additive Secret Sharing: For 2-party computation, split s as s_1, s_2 where $s_1 + s_2 = s \pmod{p}$. Addition is free: $(a_1 + b_1, a_2 + b_2)$ shares $a + b$.

6.5.3 Garbled Circuits

Garbled circuits enable secure two-party computation. Party 1 (garbler) encrypts a boolean circuit C computing f, and Party 2 (evaluator) evaluates it without learning intermediate values.

For each wire w in the circuit, the garbler creates two random keys (k_w^0, k_w^1) representing bit values 0 and 1. For each gate with input wires a, b and output wire c, the garbler creates a garbled gate table:

$$\text{Enc}_{k_a^{v_a}, k_b^{v_b}}(k_c^{g(v_a, v_b)}) \quad \text{for } v_a, v_b \in \{0, 1\} \tag{131}$$

where Enc is a symmetric encryption scheme, k_w^v is the key for wire w having value v, and g is the gate function.

The evaluator receives keys corresponding to input bits (via oblivious transfer for Party 2's inputs) and evaluates gate-by-gate, learning only the output.

6.5.4 Applications to Machine Learning

MPC enables privacy-preserving ML in several settings:

Secure Inference: A model owner and data owner can jointly compute $f(x)$ where the model owner learns nothing about x and the data owner learns nothing about model parameters beyond the prediction.

Secure Training: Multiple data owners can train a joint model without sharing raw data. Each iteration involves secure aggregation of gradients.

Challenges:

- **Communication Overhead**: MPC protocols require multiple rounds of message exchanges between parties, introducing significant network latency that can make real-time applications impractical. For example, securely computing a single neural network forward pass might require $O(d \cdot L)$ communication rounds for d input dimensions and L layers, where each round incurs network latency (typically 10-100ms for internet connections). In practice, a ResNet-50 inference with MPC could take seconds to minutes instead of milliseconds, making

it unsuitable for interactive applications like intensive care monitoring. The communication complexity scales with circuit depth; deeper computations require more sequential message passing. Optimizations like circuit pipelining and batching multiple evaluations can amortize costs, but fundamentally cannot eliminate the sequential dependency between computational layers.

- **Computational Complexity**: Cryptographic operations underlying MPC, particularly secure multiplication gates, are computationally expensive compared to plaintext operations. A secure multiplication requires homomorphic encryption operations or oblivious transfer protocols that are 10^3 to 10^6 times slower than native CPU multiplication. For instance, while a standard matrix multiplication $C = AB$ for $n \times n$ matrices takes $O(n^3)$ floating-point operations completing in microseconds on modern hardware, the secure version using garbled circuits requires generating and evaluating $O(n^3)$ encrypted gate tables, potentially taking minutes to hours for moderately sized matrices ($n \approx 1000$). The asymmetry between addition and multiplication costs means circuit optimization must carefully balance these operations; addition in additive secret sharing is essentially free (local addition of shares), while multiplication requires interaction between parties.

- **Non-linear Activation Functions**: Standard neural network activations like ReLU ($\max(0, x)$), sigmoid ($\sigma(x) = 1/(1 + e^{-x})$), and softmax require comparison or exponential operations that are extremely expensive to compute securely. The ReLU function $f(x) = \max(0, x)$ requires secure comparison to determine if $x > 0$, implemented via garbled circuits with depth $O(\log k)$ for k-bit precision, adding significant overhead to each layer. Practical MPC systems often approximate non-linear functions using low-degree polynomials (e.g., $\text{ReLU}(x) \approx x^2$ for x near 0) or use specialized protocols like CrypTFlow2 [58] that combine different MPC techniques. Alternatively, some systems replace ReLU with quadratic activations that are MPC-friendly but may reduce model accuracy. The fundamental tension is that functions easy to compute in plaintext (comparisons, table lookups, transcendental functions) become bottlenecks in the encrypted domain.

6.5.5 Homomorphic Encryption

Homomorphic encryption (HE) allows computation on encrypted data. A fully homomorphic encryption scheme supports:

$$\text{Enc}(a) \oplus \text{Enc}(b) = \text{Enc}(a + b) \tag{132}$$

$$\text{Enc}(a) \otimes \text{Enc}(b) = \text{Enc}(a \cdot b) \tag{133}$$

where $\oplus$ and $\otimes$ are operations on ciphertexts that correspond to addition and multiplication on plaintexts.

The client encrypts input x, sends $\text{Enc}(x)$ to the server, which computes $\text{Enc}(f(x))$ and returns it. The client decrypts to obtain $f(x)$.

Challenges: FHE has high computational overhead (orders of magnitude slower than plaintext). Practical schemes like CKKS support approximate arithmetic on real numbers but accumulate noise that limits computation depth.

Example 6.4 (Homomorphic Encryption for Private Medical Image Analysis)**.** A patient wants a cloud-based AI to analyze their MRI scan for tumor detection without revealing the image. Using CKKS homomorphic encryption: (1) the patient encrypts their MRI pixel values $x \in \mathbb{R}^{256\times256}$ locally, obtaining $\text{Enc}(x)$; (2) the encrypted image is sent to the cloud server; (3) the server evaluates a pre-trained neural network entirely on encrypted data, computing $\text{Enc}(Wx + b)$ using homomorphic matrix multiplication ($\otimes$) and addition ($\oplus$), approximating ReLU with

low-degree polynomials; (4) the server returns $\text{Enc}(\hat{y})$; (5) the patient decrypts locally to obtain $\hat{y} \in \{\text{benign}, \text{malignant}\}$. The server never sees the raw MRI or the diagnosis. The cost: a forward pass taking 5ms in plaintext may take 10–60 seconds under HE, and computation depth is limited to ∼10–20 multiplicative layers before noise overwhelms the signal, requiring network architecture adaptation.

6.5.6 Local Differential Privacy

In the *local* model of DP, each user randomizes their own data before sending it to the aggregator. This provides stronger privacy guarantees (no trusted curator) but requires more noise.

Definition 6.4 (Local Differential Privacy)**.** A randomized mechanism $\mathcal{M} : \mathcal{X} \to \mathcal{Y}$ is ϵ-locally differentially private if for all $x, x' \in \mathcal{X}$ and all $S \subseteq \mathcal{Y}$:

$$P(\mathcal{M}(x) \in S) \leq e^{\epsilon} P(\mathcal{M}(x') \in S) \tag{134}$$

where ϵ is the privacy budget.

Randomized Response is the canonical LDP mechanism. For a binary attribute $b \in \{0, 1\}$:

$$\mathcal{M}(b) = \begin{cases} b & \text{with probability } \frac{e^{\epsilon}}{1+e^{\epsilon}} \\ 1-b & \text{with probability } \frac{1}{1+e^{\epsilon}} \end{cases} \tag{135}$$

Example 6.5 (Randomized Response: Studying Sensitive Health Behaviors)**.** A public health researcher surveys $n = 10{,}000$ individuals about whether they smoke ($b = 1$) or not ($b = 0$), using $\epsilon = 2$ local DP. Each respondent answers truthfully with probability $e^2/(1 + e^2) \approx 0.88$ and lies with probability $1/(1 + e^2) \approx 0.12$. Suppose 2,500 individuals truly smoke (true rate $p = 0.25$). The expected number of "yes" responses is $2{,}500 \times 0.88 + 7{,}500 \times 0.12 = 2{,}200 + 900 = 3{,}100$, giving $\bar{y} = 0.31$. The researcher recovers the true rate using: $\hat{p} = \frac{(1+e^2)\times 0.31 - 1}{e^2 - 1} = \frac{8.389 \times 0.31 - 1}{6.389} = \frac{1.601}{6.389} \approx 0.251$. Even though each individual's response is plausibly deniable (any "yes" could be a true smoker who answered honestly or a non-smoker who lied), the aggregate statistic is accurately recovered.

The aggregator can estimate the true mean from noisy responses:

$$\hat{p} = \frac{(1 + e^{\epsilon})\bar{y} - 1}{e^{\epsilon} - 1} \tag{136}$$

where $\bar{y}$ is the empirical mean of the randomized responses.

6.6 Attacks on Privacy

6.6.1 Reconstruction Attacks

The Dinur-Nissim theorem states that if a mechanism answers $O(N)$ queries with error $o(\sqrt{N})$, an adversary can reconstruct the underlying database with high probability. This fundamental limit implies that we cannot answer an infinite number of queries accurately while preserving privacy.

Theorem 6.2 (Dinur-Nissim Lower Bound)**.** *Let $D \in \{0, 1\}^N$ be a database. Any mechanism that answers $c \cdot N$ linear queries with error at most $o(\sqrt{N})$ for each query enables reconstruction of D with high probability.*

Proof Sketch. Consider $c \cdot N$ random subset queries $S_1, \ldots, S_{cN}$, where each query asks for $\sum_{i \in S_j} D[i]$. With high probability, the matrix A defined by $A_{jk} = \mathbb{I}(k \in S_j)$ has full column rank when c is large enough. If we receive noisy answers $\tilde{a}_j = a_j + \text{noise}_j$ where $|\text{noise}_j| = o(\sqrt{N})$, we can solve the linear system $A \cdot D \approx \tilde{a}$ using least squares. The Gaussian elimination will recover D exactly if the total error is small enough. □

6.6.2 Membership Inference Attacks (MIA)

MIA aims to determine if a specific sample x was in the training set. The attack exploits the generalization gap: models tend to have lower loss (higher confidence) on training data than on test data.

$$\mathcal{A}(x, f) = \mathbb{I}(\ell(f(x), y) < \tau) \tag{137}$$

where $\mathbb{I}(\cdot)$ is the indicator function, ℓ is the loss function, and τ is a threshold. If the loss is below a threshold τ, the attacker predicts "Member". DP provides a theoretical upper bound on the success of any such attack, effectively closing the generalization gap.

Theorem 6.3 (MIA Upper Bound under DP)**.** *If model f is trained using an (ϵ, δ)-DP algorithm, then for any membership inference attack $\mathcal{A}$:*

$$\textit{Advantage}_{\mathcal{A}} = |P(\mathcal{A}(x) = 1 | x \in D) - P(\mathcal{A}(x) = 1 | x \notin D)| \leq e^{\epsilon} - 1 + \delta \tag{138}$$

6.6.3 Model Inversion Attacks

Model inversion attacks attempt to reconstruct training data or sensitive attributes from model access. Given a model f trained on sensitive features X to predict target Y, an attacker observing f and knowing Y attempts to infer X.

For a linear model $f(x) = w^T x$, if the attacker knows the output $y = f(x)$ and has access to the weights w, they can attempt to solve for x. For neural networks, gradient-based optimization can be used:

$$x^* = \arg\min_x \|f(x) - y_{target}\|^2 + \lambda R(x) \tag{139}$$

where $R(x)$ is a regularizer encouraging realistic inputs (e.g., total variation for images).

6.6.4 Attribute Inference Attacks

Given a trained model and partial knowledge of an individual's features, attribute inference attacks attempt to predict sensitive attributes. If model f was trained on features $(X_1, X_2, \ldots, X_d)$ and an attacker knows $(x_2, \ldots, x_d)$ and y, they can optimize:

$$\hat{x}_1 = \arg\max_{x_1} P(Y = y | X_1 = x_1, X_2 = x_2, \ldots) \tag{140}$$

These attacks are particularly concerning when models inadvertently learn to predict sensitive attributes as features for the main task.

6.7 Exercises

1. **Laplace Mechanism**:
   (a) Prove that the Laplace mechanism $\mathcal{M}(x) = f(x) + \text{Lap}(\Delta f/\epsilon)$ satisfies ϵ-DP.

(b) Compare the noise magnitude of Laplace vs. Gaussian mechanisms for the same ϵ, δ.

2. **DP-SGD**:
   (a) Implement DP-SGD from scratch in PyTorch.
   (b) Train a model on MNIST with $\epsilon = 1.0$.
   (c) Plot the privacy loss curve using the Moments Accountant.
3. **Membership Inference**:
   (a) Train a model on half of CIFAR-10 (members) and evaluate on the other half (non-members).
   (b) Plot the distribution of loss values for members vs. non-members.
   (c) Calculate the attack accuracy of a simple threshold-based MIA.
4. **Composition Theorems**:
   (a) Prove the basic composition theorem: k applications of ϵ-DP give $(k\epsilon)$-DP.
   (b) Implement and compare: basic composition, advanced composition, and Rényi DP composition.
   (c) For $k = 100$ queries with $\epsilon = 0.1$ each, compute the total ϵ under each composition rule.
   (d) Visualize how privacy budget grows with the number of queries.
5. **Local Differential Privacy**:
   (a) Implement randomized response for binary attributes.
   (b) Estimate the population mean from locally privatized data.
   (c) Compare variance of estimates under local vs. central DP.
   (d) Implement the RAPPOR protocol for frequency estimation.
6. **PATE Framework**:
   (a) Train an ensemble of 100 "teacher" models on disjoint data partitions.
   (b) Implement the noisy argmax aggregation mechanism.
   (c) Train a "student" model using the aggregated labels.
   (d) Compute the data-dependent privacy bound using the smooth sensitivity analysis.
7. **Model Inversion Attack**:
   (a) Train a face recognition model on a subset of CelebA.
   (b) Given only a class label (person ID), reconstruct a representative face.
   (c) Compare reconstruction quality with and without regularization.
   (d) Test if DP training reduces reconstruction quality.
8. **Attribute Inference Attack**:
   (a) Train a classifier that uses but does not predict a sensitive attribute.
   (b) Implement an attribute inference attack using the model's outputs.
   (c) Measure attack success rate vs. baseline (predicting majority class).
   (d) Test mitigation strategies: removing correlated features, adversarial debiasing.

9. **Reconstruction Attack**:
   (a) Implement the Dinur-Nissim attack on a synthetic database.
   (b) Query a counting query mechanism and attempt reconstruction.
   (c) Verify that reconstruction fails when queries are answered with $O(\sqrt{N})$ noise.
   (d) Demonstrate successful reconstruction with lower noise levels.
10. **Privacy Amplification**:
   (a) Implement privacy amplification by subsampling.
   (b) Compare privacy guarantees with and without subsampling.
   (c) Verify the amplification bound experimentally.
   (d) Implement privacy amplification by shuffling for the local model.

7 Explainability and Interpretability

[11, 80, 82]

7.1 Foundations of Explainable AI (XAI)

As machine learning models, particularly deep neural networks, become more complex, their decision-making processes become increasingly opaque. This "black box" nature poses significant challenges for trust, accountability, and debugging. Explainable AI (XAI) aims to make the internal logic of these systems transparent and understandable to humans.

We distinguish between two related but distinct concepts:

- **Interpretability**: The degree to which a human can understand the cause of a decision. A linear regression model with few features is intrinsically interpretable because the coefficients directly relate inputs to outputs.
- **Explainability**: The ability to provide a post-hoc description of why a model made a specific prediction. A deep neural network is not interpretable, but we can generate an explanation (e.g., a heatmap) to justify its output.

7.2 Taxonomy of XAI Methods

XAI methods can be categorized along several dimensions. **Intrinsic vs. Post-hoc** distinguishes between models that are interpretable by design (e.g., decision trees, linear models) and methods that analyze a trained black-box model without modifying its structure. **Global vs. Local** refers to the scope of the explanation: global methods describe the model's behavior across the entire input space, while local methods focus on justifying a specific prediction for a single input instance. Finally, **Model-Agnostic vs. Model-Specific** differentiates between methods that treat the model as a black box (requiring only input-output access, like LIME or SHAP) and those that exploit internal structures like gradients or attention weights (like Integrated Gradients).

7.3 Feature Attribution Methods

Feature attribution methods assign a "relevance" score to each input feature, indicating its contribution to the model's prediction.

7.3.1 Shapley Values

Originating from cooperative game theory, Shapley values provide a unique solution to the problem of fairly distributing a "payout" (prediction) among "players" (features). For a model f and an input x, the Shapley value ϕ_i for feature i is defined as the average marginal contribution of feature i across all possible coalitions of features:

$$\phi_i(f, x) \coloneqq \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} [f_x(S \cup \{i\}) - f_x(S)] \tag{141}$$

where N is the set of all features, S is a subset of features not containing i, $|S|$ is the size of the subset, and $f_x(S)$ is the prediction function evaluated with only features in S present.

Shapley values are the *only* attribution method that satisfies four desirable axioms. **Efficiency** requires that the sum of attributions equals the difference between the prediction and the baseline ($\sum \phi_i = f(x) - \mathbb{E}[f(x)]$). **Symmetry** ensures that if two features contribute equally to all coalitions, they receive equal attribution. The **Dummy** axiom states that a feature that never changes the prediction receives zero attribution. Finally, **Additivity** guarantees that the attribution for a sum of two models is the sum of their individual attributions.

Theorem 7.1 (Uniqueness of Shapley Values)**.** *The Shapley value is the unique attribution method satisfying Efficiency, Symmetry, Dummy, and Additivity.*

Proof Sketch. Consider the set of all $2^{|N|}$ coalitions. For each coalition S, define the characteristic function $v(S) = f_x(S)$. The four axioms impose a system of linear constraints on the attributions ϕ_i. Efficiency gives one equation ($\sum_i \phi_i = v(N) - v(\emptyset)$). Symmetry and Dummy reduce the degrees of freedom by identifying features with equal contributions. One can show inductively that these constraints uniquely determine each ϕ_i, and the resulting value is precisely the Shapley formula. The proof relies on showing that any other attribution satisfying these axioms must equal the Shapley value for all possible coalitional games. □

Exact computation is exponential in the number of features ($O(2^N)$). In practice, we use approximations like **KernelSHAP** [43], which casts the computation as a weighted linear regression problem, or **DeepSHAP**, which leverages the compositional structure of neural networks.

Example 7.1 (Shapley Values for a Simple Patient Recovery Prediction Model)**.** Consider a model f predicting recovery rate (index score) from three features: x_1 = blood pressure (mmHg), x_2 = temperature (°C), x_3 = medication dosage (mg). For a specific patient with $x = (120, 38, 50)$, the model predicts $f(x) = 6.2$ score, while the baseline (average prediction) is $\mathbb{E}[f(x)] = 4.0$ score. To compute ϕ_1 (blood pressure's contribution), we evaluate all coalitions:

- $f_x(\emptyset) = 4.0$ (baseline), $f_x(\{1\}) = 5.1$ (blood pressure alone adds 1.1)
- $f_x(\{2\}) = 4.3$, $f_x(\{3\}) = 4.5$, $f_x(\{1,2\}) = 5.6$, $f_x(\{1,3\}) = 5.8$
- $f_x(\{2,3\}) = 4.9$, $f_x(\{1,2,3\}) = 6.2$

The marginal contributions of rainfall are: $f_x(\{1\}) - f_x(\emptyset) = 1.1$, $f_x(\{1,2\}) - f_x(\{2\}) = 1.3$, $f_x(\{1,3\}) - f_x(\{3\}) = 1.3$, $f_x(\{1,2,3\}) - f_x(\{2,3\}) = 1.3$. Weighting by $\frac{|S|!(N-|S|-1)!}{N!}$ and summing: $\phi_1 = \frac{1}{3}(1.1) + \frac{1}{6}(1.3) + \frac{1}{6}(1.3) + \frac{1}{3}(1.3) = 1.23$. Similarly, $\phi_2 \approx 0.38$ and $\phi_3 \approx 0.58$. The efficiency axiom is satisfied: the exact Shapley values sum to $\phi_1 + \phi_2 + \phi_3 = 2.2 = f(x) - \mathbb{E}[f(x)]$.

7.3.2 LIME (Local Interpretable Model-agnostic Explanations)

LIME [59] explains a complex model f by approximating it locally with an interpretable model g (e.g., a sparse linear model). The objective is to minimize the following loss function:

$$\xi(x) \coloneqq \arg\min_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{142}$$

where $\mathcal{L}$ measures the fidelity of g to f in the neighborhood of x, π_x is a proximity kernel (e.g., exponential kernel) that weights samples based on their distance to x, and $\Omega(g)$ penalizes the complexity of the explanation (e.g., number of non-zero coefficients). LIME generates a dataset of perturbed samples around x, queries the black box f for labels, and trains g on this weighted dataset. While powerful, LIME suffers from instability: small changes in the sampling can lead to different explanations.

7.3.3 Gradient-Based Methods

For differentiable models, we can use gradients to estimate feature importance. **Saliency Maps (Vanilla Gradients)** compute $M_i(x) = |\frac{\partial f(x)}{\partial x_i}|$, measuring the sensitivity of the output to infinitesimal changes in the input. However, it suffers from "saturation" (gradients can be zero even if the feature is important) and noise. **Integrated Gradients (IG)** [65] addresses the saturation problem by accumulating gradients along a straight path from a baseline x' (usually a zero vector) to the input x:

$$IG_i(x) = (x_i - x_i') \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha \tag{143}$$

where x is the input, x' is the baseline, α is the interpolation parameter, and $\frac{\partial f}{\partial x_i}$ is the gradient of f with respect to feature i. IG satisfies the axiom of **Completeness**: $\sum IG_i(x) = f(x) - f(x')$.

Proof of Completeness. Consider the path $\gamma(\alpha) = x' + \alpha(x - x')$ for $\alpha \in [0, 1]$. By the chain rule:

$$\frac{d}{d\alpha} f(\gamma(\alpha)) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\gamma(\alpha)) \cdot \frac{d\gamma_i}{d\alpha} = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(\gamma(\alpha)) \cdot (x_i - x_i') \tag{144}$$

where $\gamma(\alpha)$ is the path function. Integrating both sides from $\alpha = 0$ to $\alpha = 1$:

$$\int_0^1 \frac{d}{d\alpha} f(\gamma(\alpha)) d\alpha = \int_0^1 \sum_{i=1}^{n} (x_i - x_i') \frac{\partial f}{\partial x_i}(\gamma(\alpha)) d\alpha \tag{145}$$

$$f(\gamma(1)) - f(\gamma(0)) = \sum_{i=1}^{n} (x_i - x_i') \int_0^1 \frac{\partial f}{\partial x_i}(\gamma(\alpha)) d\alpha \tag{146}$$

$$f(x) - f(x') = \sum_{i=1}^{n} IG_i(x) \tag{147}$$

This follows from the fundamental theorem of calculus. □

7.4 Concept-Based Explanations

Feature attribution tells us *where* the model is looking (e.g., pixels), but not *what* it sees (e.g., "floppy ears"). Concept-based methods bridge this gap.

7.4.1 Concept Bottleneck Models (CBM)

CBMs enforce a structure where the input x is first mapped to a set of human-interpretable concepts c, and then these concepts are used to predict the target y:

$$x \xrightarrow{h} c \xrightarrow{g} y \tag{148}$$

where x is the input, c represents the concepts, y is the target, h is the concept mapping function, and g is the prediction function. This architecture allows for **intervenability**: if the model predicts "Wolf" because it wrongly detected "Snow", a human expert can correct the concept "Snow" to "Grass" and observe the updated prediction.

7.4.2 Testing with Concept Activation Vectors (TCAV)

TCAV quantifies the sensitivity of a prediction to a high-level concept (e.g., "striped") even if that concept was not explicitly part of the training labels. It learns a vector v_C in the activation space of a layer that separates examples of the concept from random examples. The conceptual sensitivity is then given by the directional derivative:

$$S_{C,k}(x) \coloneqq \nabla h_k(x) \cdot v_C \tag{149}$$

where $h_k(x)$ is the activation at layer k, and v_C is the concept vector.

7.5 Evaluating Explanations

Evaluating XAI is notoriously difficult due to the lack of ground truth. Common metrics include **Fidelity**, which measures if the explanation accurately reflects the model's behavior (e.g., by removing important features and checking if the prediction drops); and **Robustness**, which checks if similar inputs yield similar explanations (LIME often fails this). Additionally, **Sanity Checks** [4] are crucial: many saliency methods (like Guided Backprop) act merely as edge detectors and do not depend on the model parameters. It is crucial to verify that randomizing the model weights destroys the explanation.

7.6 Algorithmic Recourse and Counterfactual Explanations

While feature attribution explains *why* a decision was made, counterfactual explanations answer the question: *"What would need to change for the outcome to be different?"*

7.6.1 The Counterfactual Explanation Problem

Definition 7.1 (Counterfactual Explanation)**.** Given a classifier $f : \mathcal{X} \to \{0, 1\}$, an input x with $f(x) = 0$ (unfavorable outcome), a counterfactual explanation is an input x' such that:

1. $f(x') = 1$ (desired outcome)
2. x' is "close" to x under some distance measure $d(x, x')$

The optimization problem is:

$$x^* = \arg\min_{x'} d(x, x') \quad \text{s.t.} \quad f(x') = 1 \tag{150}$$

where $d(\cdot, \cdot)$ is a distance metric, f is the classifier, and x' is the counterfactual instance.

The choice of distance function d is critical and encodes assumptions about actionability. Common choices include:

- L_1 norm: Promotes sparsity (few features change)
- L_2 norm: Smooth changes across features
- Weighted norms: Account for feature costs (e.g., changing age is impossible)
- Mahalanobis distance: Account for feature correlations

Example 7.2 (Counterfactual Explanation for a Rejected Medical Procedure Request)**.** A patient applies for coverage of an expensive joint procedure and is rejected ($f(x) = 0$). The features are: severity_score = 3.1, prior_treatments = 0, bmi = 28, physical_therapy_sessions = 2. A counterfactual explanation using L_1 norm finds the closest approved profile: $x^* = (\text{severity\_score} = 3.1, \text{prior\_treatments} = 1, \text{bmi} = 28, \text{physical\_therapy\_sessions} = 2)$ with $d(x, x^*) = 1$ (only one feature changed). The explanation tells the patient: "Having one prior conservative treatment would have changed the outcome to approved." With DICE generating diverse counterfactuals, two alternative paths emerge: (1) increase physical therapy sessions to 6 (one change), or (2) complete 4 more therapy sessions and show stable BMI (two changes). These multiple actionable paths empower the patient to choose the most feasible route to approval.

7.6.2 Wachter's Method

[71] proposed solving the unconstrained optimization problem:

$$\min_{x'} \lambda(f(x') - 1)^2 + d(x, x') \tag{151}$$

where λ is a regularization parameter balancing validity and proximity. For differentiable models, this can be solved via gradient descent. The hyperparameter λ balances validity (reaching the desired class) and proximity (staying close to x).

7.6.3 Actionability Constraints

Not all features can be changed (e.g., age, race, place of birth). We partition features into:

- **Immutable**: Cannot be changed under any circumstance ($x'_i = x_i$)
- **Actionable**: Can be changed by the individual
- **Increasing-only**: Can only increase (e.g., education level, years of experience)

The constrained optimization problem becomes:

$$\min_{x'} d(x, x') \quad \text{s.t.} \quad f(x') = 1,\ x'_i = x_i\ \forall i \in \mathcal{I}_{\text{immutable}},\ x'_j \geq x_j\ \forall j \in \mathcal{I}_{\text{increasing}} \tag{152}$$

where $\mathcal{I}_{\text{immutable}}$ is the set of immutable features and $\mathcal{I}_{\text{increasing}}$ is the set of increasing-only features.

7.6.4 DICE: Diverse Counterfactual Explanations

A single counterfactual may not be the only option. DICE ([50]) generates a diverse set of counterfactuals by adding a diversity term:

$$\min_{\{x'_1,\ldots,x'_k\}} \sum_{i=1}^{k} \left[\lambda_1 \text{yloss}(f(x'_i), 1) + d(x, x'_i)\right] - \lambda_2 \sum_{i \neq j} \text{dpp\_diversity}(x'_i, x'_j) \tag{153}$$

where k is the number of counterfactuals, yloss is the loss function for the target class, dpp_diversity is the diversity metric, and λ_1, λ_2 are hyperparameters.

7.6.5 Causal Counterfactuals

Standard counterfactual methods may suggest changes that violate causal relationships in the real world. For example, suggesting "lower your BMI without changing your diet or exercise" is meaningless.

Definition 7.2 (Causally Grounded Counterfactual)**.** Given an SCM $\mathcal{M}$ with structural equations $V_i = f_i(\text{Pa}_i, U_i)$, a causally grounded counterfactual modifies only the exogenous variables or actionable root causes, then propagates changes through the causal graph:

$$x'_{\text{downstream}} = f_{\text{downstream}}(x'_{\text{actionable}}, u) \tag{154}$$

where $x'_{\text{actionable}}$ are the modified actionable variables, u represents exogenous variables, and $f_{\text{downstream}}$ propagates the changes.

This ensures that downstream features are updated consistently according to the causal model.

7.6.6 Recourse Robustness

A critical issue is that the model may change after the counterfactual is provided. If a user follows the recommended changes but the model has been retrained, they may not achieve the desired outcome.

Definition 7.3 (Robust Recourse)**.** A counterfactual x' provides ϵ-robust recourse if:

$$f_{\theta'}(x') = 1 \quad \forall \theta' \in \mathcal{B}(\theta, \epsilon) \tag{155}$$

where $\mathcal{B}(\theta, \epsilon)$ is the uncertainty set of model parameters with radius ϵ.

Robust recourse can be achieved by optimizing:

$$\min_{x'} d(x, x') \quad \text{s.t.} \quad \min_{\theta' \in \mathcal{B}(\theta,\epsilon)} f_{\theta'}(x') \geq 0.5 \tag{156}$$

where the constraint ensures the prediction remains valid under the worst-case parameter perturbation.

7.6.7 Algorithmic Recourse via Integer Programming

For tree-based models (Random Forests, Gradient Boosted Trees), counterfactual search can be formulated as a Mixed Integer Program (MIP).

For a single decision tree with leaves L and decision paths $\mathcal{P}_\ell$ for each leaf ℓ:

$$\min_{x', z} \quad d(x, x') \tag{157}$$

$$\text{s.t.} \quad \sum_{\ell \in L^+} z_\ell = 1 \quad \text{(reach exactly one positive leaf)} \tag{158}$$

$$x'_j \leq u_{j,\ell} + M(1 - z_\ell) \quad \forall j, \ell \tag{159}$$

$$x'_j \geq l_{j,\ell} - M(1 - z_\ell) \quad \forall j, \ell \tag{160}$$

$$z_\ell \in \{0, 1\} \tag{161}$$

where z_ℓ is a binary variable indicating if leaf ℓ is active, L^+ is the set of positive leaves, $u_{j,\ell}$ and $l_{j,\ell}$ are feature bounds, and M is a large constant.

Exercises

1. **Shapley Values Implementation**:
   (a) Implement the exact Shapley value calculation for a toy model with $N = 4$ features.
   (b) Implement the Monte Carlo approximation and plot the convergence of the estimates as the number of samples increases.
   (c) Compare the results with the 'shap' library in Python.
2. **LIME and Stability**:
   (a) Use LIME to explain a Random Forest classifier on the Iris dataset.
   (b) Run LIME multiple times for the same instance with different random seeds. Quantify the stability of the explanations using the Jaccard similarity of the top-k features.
3. **Gradient-Based Explanations**:
   (a) Train a simple CNN on MNIST.
   (b) Implement Vanilla Gradients and Integrated Gradients from scratch.
   (c) Visualize the explanations for a specific digit.
   (d) Perform the "Model Parameter Randomization Test": Randomize the weights of the top layer and check if the heatmap changes significantly.
4. **Concept-Based Explanations**:
   (a) Implement TCAV for a pre-trained ImageNet classifier.
   (b) Define concept directions for textures (striped, dotted, smooth).
   (c) Measure the sensitivity of class predictions (e.g., "zebra") to each concept.
   (d) Compare TCAV scores across different classes to validate concept relevance.
5. **Counterfactual Explanations**:
   (a) Implement Wachter's method for generating counterfactuals.
   (b) Add actionability constraints (immutable features, monotonic features).
   (c) Generate diverse counterfactuals using the DiCE algorithm.
   (d) Evaluate counterfactual quality using proximity, sparsity, and plausibility metrics.
6. **Attention-Based Explanations**:
   (a) Train a Transformer model on a text classification task.
   (b) Visualize attention weights for sample predictions.
   (c) Test if attention weights are faithful explanations by perturbing high-attention tokens.
   (d) Compare attention-based explanations with gradient-based methods on the same inputs.
7. **Explanation Evaluation**:
   (a) Implement the "deletion" and "insertion" faithfulness metrics.
   (b) Compare faithfulness scores for LIME, SHAP, and Integrated Gradients.
   (c) Measure explanation stability under input perturbations.
   (d) Conduct a user study to evaluate explanation comprehensibility.

8. **Rule Extraction**:
   (a) Train a neural network on tabular data (e.g., Patient Health Outcomes).
   (b) Extract decision rules that approximate the network's behavior.
   (c) Measure fidelity: how often do the rules agree with the network?
   (d) Trade off rule complexity (number of rules, conditions) with fidelity.

8 Causal Inference and Machine Learning

[33, 35]

8.1 The Ladder of Causation

Judea Pearl [53] distinguishes three levels of cognitive ability, known as the "Ladder of Causation". The first level, **Association** ($P(y|x)$), involves observing correlations ("Seeing")—asking how observing X changes belief in Y. This is the domain of standard statistics and most machine learning. The second level, **Intervention** ($P(y|do(x))$), involves acting ("Doing")—asking what happens if we force X to take a value. The third and highest level, **Counterfactuals** ($P(y_x|x', y')$), involves imagining ("Why?")—reasoning about hypothetical worlds to determine if X caused Y or what would have happened had we acted differently.

Example 8.1 (Ladder of Causation in Medical Treatment)**.** Consider the question "Does a new medication improve patient recovery?"

- **Level 1 (Association)**: We observe that patients using the medication have 20% higher recovery rates: $P(\text{recovery}|\text{medication}) = 0.75$ vs. $P(\text{recovery}|\text{no medication}) = 0.55$. But is this because the medication works, or because healthier patients are prescribed it?
- **Level 2 (Intervention)**: We conduct a randomized controlled trial, *forcing* random patients to receive the medication regardless of baseline health: $P(\text{recovery}|do(\text{medication})) = 0.68$. The causal effect is $0.68 - 0.60 = 0.08$ (8% improvement), smaller than the observational correlation suggested, because baseline health was a confounder.
- **Level 3 (Counterfactual)**: For a specific patient who was prescribed the medication and recovered, we ask: "Would this patient have recovered without the medication?" Given their excellent baseline health ($U_{\text{health}} = \text{excellent}$), the answer is "probably yes"; the medication was not the cause for *this particular patient*.

8.2 Structural Causal Models (SCM)

A Structural Causal Model is a tuple $\mathcal{M} = \langle U, V, F, P(U) \rangle$ where:

- $U = \{U_1, \ldots, U_m\}$ is a set of exogenous (background) variables determined by factors outside the model.
- $V = \{V_1, \ldots, V_n\}$ is a set of endogenous variables determined by variables in the model.
- $F = \{f_1, \ldots, f_n\}$ is a set of structural equations $v_i = f_i(pa_i, u_i)$, where $pa_i \subseteq V \setminus \{v_i\}$ are the parents of v_i.
- $P(U)$ is a probability distribution over the exogenous variables.

The model implies a Causal Diagram (DAG) where arrows point from parents to children.

8.2.1 d-Separation

A path p is d-separated (blocked) by a set of nodes Z if it satisfies one of two conditions: (1) p contains a chain $i \to m \to j$ or a fork $i \leftarrow m \to j$ such that the middle node m is in Z; or (2) p contains a collider $i \to m \leftarrow j$ such that the middle node m is NOT in Z and no descendant of m is in Z. If X and Y are d-separated by Z, they are conditionally independent given Z in any distribution compatible with the graph: $X \perp Y|Z$.

8.3 Interventions and the Do-Operator

The *do*-operator, denoted $do(X = x)$, represents an intervention that forces the variable X to take the value x, regardless of the structural equation f_X. Graphically, this corresponds to removing all incoming arrows to X. The interventional distribution $P(Y|do(X = x))$ is generally different from the observational distribution $P(Y|X = x)$.

8.3.1 The Back-Door Criterion

A set of variables Z satisfies the Back-Door Criterion relative to an ordered pair (X, Y) if:

1. No node in Z is a descendant of X.
2. Z blocks every path between X and Y that contains an arrow into X (back-door paths).

If Z satisfies the Back-Door Criterion, we can identify the causal effect via the **Adjustment Formula**:

$$P(Y = y|do(X = x)) = \sum_z P(Y = y|X = x, Z = z)P(Z = z) \tag{162}$$

where Z is the set of variables satisfying the Back-Door Criterion.

Example 8.2 (Back-Door Criterion: Effect of Exercise on Cardiovascular Health)**.** Consider the causal graph: Genetics(Z) $\rightarrow$ Exercise(X), Genetics(Z) $\rightarrow$ Heart Health(Y), Exercise(X) $\rightarrow$ Heart Health(Y). We want the causal effect of exercise on heart health. Genetics Z is a confounder (people with certain genetic profiles may exercise more *and* have naturally healthier hearts). The set $Z = \{\text{Genetics}\}$ satisfies the back-door criterion: (1) it is not a descendant of X, and (2) it blocks the back-door path $X \leftarrow Z \rightarrow Y$. The adjustment formula gives: $P(Y|do(X = x)) = \sum_z P(Y|X = x, Z = z)P(Z = z)$. In practice, we stratify by genetic risk category (e.g., low/medium/high), compute the effect of exercise on heart health within each stratum, and average, removing the confounding bias.

8.4 The Do-Calculus

When the Back-Door Criterion is not applicable (e.g., due to unobserved confounders), we can use the rules of do-calculus to transform interventional probabilities into observational ones. Let $G_{\bar{X}}$ denote the graph with arrows into X removed, and $G_{\underline{X}}$ denote the graph with arrows out of X removed. The three rules are:

- **Rule 1 (Insertion/deletion of observations)**: $P(y|do(x), z, w) = P(y|do(x), w)$ if $(Y \perp Z|X, W)_{G_{\bar{X}}}$.
- **Rule 2 (Action/observation exchange)**: $P(y|do(x), do(z), w) = P(y|do(x), z, w)$ if $(Y \perp Z|X, W)_{G_{\bar{X}\underline{Z}}}$.
- **Rule 3 (Insertion/deletion of actions)**: $P(y|do(x), do(z), w) = P(y|do(x), w)$ if $(Y \perp Z|X, W)_{G_{\bar{X}\bar{Z(W)}}}$, where $Z(W)$ is the set of Z-nodes that are not ancestors of any W-node in $G_{\bar{X}}$.

Proof Sketch of Rule 1. Rule 1 says we can ignore observed variables Z that are conditionally independent of Y given (X, W) in the manipulated graph $G_{\bar{X}}$ (with arrows into X removed). Under intervention $do(x)$, the distribution of Y depends only on causal paths from X to Y. If $Z \perp Y|(X, W)$ in $G_{\bar{X}}$, then Z provides no additional information about Y beyond what

(X, W) already provide. By the definition of conditional independence in probability, we have $P(y|do(x), z, w) = P(y|do(x), w)$. The removal of arrows into X in $G_{\bar{X}}$ reflects that we've already intervened on X, so its parents no longer matter. □

Proof Sketch of Rule 2. Rule 2 allows exchanging an intervention $do(z)$ for an observation z when Z is conditionally independent of Y given (X, W) in the graph $G_{\bar{X}\underline{Z}}$ (arrows into X removed, arrows out of Z removed). Removing arrows out of Z corresponds to treating Z as if it were intervened upon (its value doesn't affect its children). If under this manipulation $Z \perp Y|(X, W)$, then whether we set Z by intervention or observe it makes no difference to the distribution of Y. This is because the paths from Z to Y are already blocked by (X, W), so the mechanism that generates Z is irrelevant. □

Proof Sketch of Rule 3. Rule 3 allows removing an intervention $do(z)$ entirely when Z is conditionally independent of Y given (X, W) in $G_{\bar{X}\bar{Z(W)}}$ (arrows into X and into nodes in $Z(W)$ removed). This graph reflects a scenario where we've intervened on both X and Z. If $Z \perp Y|(X, W)$ in this graph, then Z has no causal effect on Y beyond what's already captured by (X, W). Therefore, the intervention on Z is redundant and can be removed. The notation $\bar{Z}(W)$ denotes the removal of incoming arrows to the nodes in $Z(W)$. □

Shpitser and Pearl (2006) proved that these three rules are complete: if a causal effect is identifiable, it can be derived using these rules.

8.5 Counterfactuals

A counterfactual query asks: "Given that we observed $X = x$ and $Y = y$, what would Y have been if X had been x'?" This is denoted as $P(Y_{x'} = y'|X = x, Y = y)$. Computing counterfactuals involves three steps:

1. **Abduction**: Use the evidence $E = \{X = x, Y = y\}$ to update the distribution of exogenous variables $P(U) \to P(U|E)$.
2. **Action**: Modify the model $\mathcal{M}$ by replacing the equation for X with $X = x'$, creating a sub-model $\mathcal{M}_{x'}$.
3. **Prediction**: Compute the probability of $Y = y'$ in the modified model $\mathcal{M}_{x'}$ using the updated posterior $P(U|E)$.

8.6 Causal Fairness

Standard fairness metrics (like Demographic Parity) are based on correlations. Causal fairness defines fairness in terms of the causal graph. The central notion is *counterfactual fairness* (Definition 5.11), which requires that a predictor $\hat{Y}$ would not change its output for an individual if their sensitive attribute A had been different, holding all non-descendant background variables U constant:

$$P(\hat{Y}_{A\leftarrow a}(U) = y|X = x, A = a) = P(\hat{Y}_{A\leftarrow a'}(U) = y|X = x, A = a) \tag{163}$$

The causal perspective enriches the fairness analysis by distinguishing between direct discrimination (the direct causal effect of A on $\hat{Y}$) and indirect effects flowing through mediators; see Section 5.8 for the full treatment including path-specific fairness and implementation strategies.

8.7 Exercises

1. **Simpson's Paradox**:
   (a) Construct a dataset where $P(Y = 1|T = 1) > P(Y = 1|T = 0)$ but $P(Y = 1|do(T = 1)) < P(Y = 1|do(T = 0))$.
   (b) Draw the causal graph and identify the confounder.
   (c) Calculate the Average Treatment Effect (ATE) using the adjustment formula.
2. **Do-Calculus**:
   (a) Prove Rule 1 of the do-calculus using d-separation.
   (b) Given the "Front-Door" graph $X \to Z \to Y$ with an unobserved confounder U affecting X and Y, prove that $P(y|do(x))$ is identifiable and derive the formula.
3. **Counterfactuals**:
   (a) Consider a linear SCM: $X = U_X$, $Y = \beta X + U_Y$.
   (b) Given observation $X = 1, Y = 2$, calculate the counterfactual $Y_{X=0}$.
   (c) How does the result change if the relationship is non-linear?
4. **Causal Discovery**:
   (a) Implement the PC algorithm for learning causal graphs from observational data.
   (b) Generate synthetic data from a known SCM and test if PC recovers the correct skeleton.
   (c) Compare with score-based methods (GES) and continuous optimization (NOTEARS).
   (d) Discuss the identifiability assumptions required by each method.
5. **Instrumental Variables**:
   (a) Explain why naive regression fails when there is unmeasured confounding.
   (b) Implement Two-Stage Least Squares (2SLS) for a simple model with one instrument.
   (c) Verify the validity assumptions: relevance ($\text{Cov}(Z, X) \neq 0$) and exclusion restriction.
   (d) Compare 2SLS estimates with naive OLS on simulated data with known ground truth.
6. **Causal Effect Estimation**:
   (a) Implement propensity score matching for ATE estimation.
   (b) Compare with Inverse Probability Weighting (IPW) and Doubly Robust estimation.
   (c) Use the IHDP benchmark dataset to evaluate bias and variance of different estimators.
   (d) Implement CATE estimation using causal forests or BART.
7. **Counterfactual Fairness Implementation**:
   (a) Build a causal graph for a hiring scenario with features: education, experience, gender, and qualifications.
   (b) Identify which features are descendants of the protected attribute.
   (c) Train a counterfactually fair classifier that only uses non-descendants.
   (d) Compare accuracy and fairness metrics with an unconstrained classifier.
8. **Mediation Analysis**:

(a) Decompose the total effect into direct and indirect effects for a simple mediation model $X \to M \to Y$.

(b) Derive the natural direct effect (NDE) and natural indirect effect (NIE).

(c) Implement the mediation formula and test on simulated data.

(d) Discuss when mediation effects are identifiable under unmeasured confounding.

9 Federated Learning

[24, 40, 81]

9.1 Introduction to Decentralized Learning

Federated Learning (FL) is a distributed machine learning paradigm where multiple clients (e.g., mobile devices, hospitals) collaboratively train a model under the coordination of a central server, without sharing their raw training data. The core principle is "bring the code to the data, not the data to the code."

This paradigm primarily addresses **Data Privacy and Sovereignty** by design. Unlike centralized training where all data is collected in one location, creating a single point of failure, FL ensures data remains distributed. Each client stores its own data locally and only shares model gradients or weight updates with the central server [75]. This architectural principle enables compliance with rigorous data localization requirements and privacy standards without exposing raw records to external entities. For instance, in a medical context, hospitals can collaborate to train a disease prediction model without ever exchanging patient records, thus maintaining institutional data sovereignty. However, privacy is not absolute—gradients can leak information about training data through inversion attacks, necessitating additional protections like differential privacy.

Additionally, FL offers significant **Communication Efficiency**. In many settings, transmitting entire datasets is prohibitive due to bandwidth, energy, or storage constraints. FL transmits only model updates, which are proportional to the model size rather than the dataset size. For a model with d parameters, each communication round exchanges $O(d)$ values, a strict lower bound compared to transferring $O(n)$ data samples. Further optimizations like gradient compression and quantization can reduce communication costs by orders of magnitude, making distributed training viable even over constrained networks [57].

9.2 Federated Optimization Algorithms

The goal is to minimize the global objective function:

$$\min_w F(w) = \sum_{k=1}^{K} p_k F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{D}_k} \ell(w; x_i, y_i) \tag{164}$$

where K is the number of clients, $p_k = n_k/n$ is the weight of client k, $F_k(w)$ is the local loss function, $\mathcal{D}_k$ is the local dataset, and ℓ is the per-sample loss.

9.2.1 Federated Averaging (FedAvg)

FedAvg [46] is the standard algorithm for FL. It combines local SGD on clients with model averaging on the server.

9.2.2 Handling Statistical Heterogeneity (FedProx)

In real-world settings, data is Non-IID (not Independent and Identically Distributed). For example, one user may only take photos of cats, while another only takes photos of dogs. This "client drift" or "data heterogeneity" causes local models to diverge significantly from the global optimum, as different clients optimize their local losses toward different solutions. This

Algorithm 6 Federated Averaging (FedAvg)

Server: Initialize w_0
for round $t = 1$ to T **do**
  Server selects a subset of clients $S_t \subseteq \{1, \ldots, K\}$
  Server broadcasts w_t to all $k \in S_t$
  for client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  end for
  Server aggregates: $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{\sum_{j \in S_t} n_j} w_{t+1}^k$
end for
Function ClientUpdate(k, w):
$\mathcal{B} \leftarrow$ (split local data $\mathcal{D}_k$ into batches)
for local epoch $e = 1$ to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
  end for
end for
return w

is a fundamental challenge that distinguishes federated learning from distributed learning on homogeneous data.

FedProx addresses this by adding a proximal regularization term to the local objective:

$$\min_w F_k(w) + \frac{\mu}{2} \|w - w_t\|^2 \tag{165}$$

where w_t is the global model from the previous round, and μ is the proximal term coefficient. This term acts as a regularizer that encourages local updates to stay within a neighborhood of the global model, preventing excessive client drift. The hyperparameter $\mu > 0$ controls the strength of this constraint. When μ is large, clients are forced to make only small local updates, stabilizing convergence. When μ is small, clients have more freedom to optimize their local objectives. FedProx can be interpreted as a primal-dual method where we implicitly handle the heterogeneity through this proximal term.

9.2.3 Convergence Analysis

Understanding the convergence of federated algorithms is crucial for practical deployment. We analyze FedAvg under different assumptions.

Assumption 9.1 (Smoothness)**.** Each local objective F_k is L-smooth:

$$\|\nabla F_k(w) - \nabla F_k(w')\| \leq L \|w - w'\| \quad \forall w, w' \tag{166}$$

where L is the Lipschitz constant of the gradient.

Assumption 9.2 (Bounded Variance)**.** The stochastic gradients have bounded variance:

$$\mathbb{E}[\|\nabla F_k(w; \xi) - \nabla F_k(w)\|^2] \leq \sigma_k^2 \tag{167}$$

where ξ represents the randomness in the mini-batch, and σ_k^2 is the variance bound.

Assumption 9.3 (Bounded Heterogeneity)**.** The gradient dissimilarity across clients is bounded:

$$\frac{1}{K} \sum_{k=1}^{K} \|\nabla F_k(w) - \nabla F(w)\|^2 \leq \sigma_g^2 \tag{168}$$

where σ_g^2 bounds the dissimilarity between local and global gradients.

Theorem 9.1 (FedAvg Convergence)**.** *Under Assumptions 1-3, for convex objectives, FedAvg with learning rate $\eta = O(1/\sqrt{TKE})$ achieves:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}[\|\nabla F(w_t)\|^2] \leq O\left(\frac{1}{\sqrt{TKE}} + \frac{E^2\sigma_g^2}{T}\right) \tag{169}$$

where T is the total number of rounds, K is the number of participating clients, and E is the number of local epochs.

The second term captures the effect of client drift due to heterogeneity. It grows with E, creating a tension: more local computation (E) reduces communication but increases drift.

9.3 Personalized Federated Learning

When client data distributions are heterogeneous, a single global model may be suboptimal for individual clients. Personalized FL learns models tailored to each client while still leveraging collaborative learning.

9.3.1 Local Fine-Tuning

The simplest approach: train a global model via FedAvg, then fine-tune locally on each client's data:

$$w_k^* = w_{global} - \eta \sum_{t=1}^{T_{local}} \nabla F_k(w_k^{(t)}) \tag{170}$$

where w_{global} is the trained global model, η is the learning rate, and T_{local} is the number of fine-tuning steps.

9.3.2 Personalized Layers

Decompose the model into shared base layers ϕ and personal head layers ψ_k:

$$f_k(x) = \psi_k(\phi(x)) \tag{171}$$

where ϕ is the shared feature extractor and ψ_k is the client-specific head. Only ϕ is federated; each ψ_k remains local. This is effective when feature extraction generalizes but final predictions are client-specific.

9.3.3 MAML-based Personalization

Per-FedAvg uses Model-Agnostic Meta-Learning (MAML) to find a global model that adapts quickly to each client. Rigorously, for each client k, we split local data into support D_k^{sup} and query D_k^{qry} sets:

$$\min_w \sum_{k=1}^{K} \mathcal{L}(w - \alpha\nabla\mathcal{L}(w; D_k^{sup}); D_k^{qry}) \tag{172}$$

where α is the inner loop learning rate. The global model is optimized so that one gradient step on the support set yields good performance on the query set. This creates models that personalize efficiently.

9.3.4 Mixture of Experts

Train multiple global models (experts) and learn a client-specific mixture:

$$f_k(x) = \sum_{j=1}^{M} \pi_{kj} f_j(x) \tag{173}$$

where f_j are the expert models and π_{kj} are the mixing coefficients for client k.

9.4 Communication Efficiency

Communication is often the bottleneck in FL. Several techniques reduce communication costs.

9.4.1 Gradient Compression

Quantization: Reduce gradient precision from 32-bit floats to lower bit representations.

$$Q(g) = \|g\|_2 \cdot \text{sign}(g) \cdot \xi \tag{174}$$

where g is the gradient vector, and ξ is a random binary vector with $\xi_i \sim \text{Bernoulli}(|g_i|/\|g\|_\infty)$. This stochastic quantization is unbiased: $\mathbb{E}[Q(g)] = g$.

Sparsification: Only communicate top-k gradient components:

$$\text{Top}_k(g) = g \odot \mathbb{I}(|g| \geq |g|_{(k)}) \tag{175}$$

where $\odot$ denotes element-wise multiplication, and $|g|_{(k)}$ is the k-th largest magnitude element. Combined with error feedback (accumulating dropped components), this preserves convergence.

Low-Rank Approximation: Approximate gradient matrices with low-rank decompositions:

$$G \approx UV^T \tag{176}$$

where G is the gradient matrix, $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times r}$, and r is the rank of the approximation. For rank $r \ll \min(m, n)$, this reduces communication from $O(mn)$ to $O((m + n)r)$.

9.5 Privacy and Security in FL

While FL offers better privacy than centralized training, it is not perfectly secure.

9.5.1 Gradient Inversion Attacks

Gradients contain information about the training data. In "Deep Leakage from Gradients" (DLG), an attacker (the server) tries to reconstruct the private input x and label y by optimizing dummy data x', y' to match the received gradient ∇w:

$$x^*, y^* = \arg\min_{x', y'} \|\nabla W(x', y') - \nabla W_{real}\|^2 \tag{177}$$

where ∇W_{real} is the observed gradient and (x', y') are the dummy input and label. This optimization can surprisingly recover pixel-perfect images from gradients in deep networks.

9.5.2 Secure Aggregation

To prevent the server from inspecting individual updates, Secure Aggregation (SecAgg) uses Multi-Party Computation (MPC). Clients add random masks to their updates such that the masks cancel out when summed:

$$\sum_k (w^k + r_k) = \sum w^k + \sum r_k = \sum w^k \quad (\text{if } \sum r_k = 0) \tag{178}$$

where w^k is the update from client k, and r_k are random masks designed to sum to zero. Common protocols use pairwise masking based on Diffie-Hellman key exchange.

9.6 Robustness to Byzantine Attacks

Malicious clients may send corrupted updates to poison the global model (e.g., to induce a backdoor). Standard averaging is sensitive to outliers. Robust aggregation rules replace the mean with robust statistics:

- **Krum**: Selects the update that is closest to its $n - f - 2$ neighbors.
- **Coordinate-wise Median**: Takes the median of each coordinate of the gradient vector.
- **Trimmed Mean**: Discards the largest and smallest values for each coordinate before averaging.

9.7 Exercises

1. **FedAvg and Non-IID Data**:
   (a) Simulate a FL setting with 10 clients using the MNIST dataset.
   (b) Create a pathological Non-IID split where each client has data from only 2 digits.
   (c) Compare the test accuracy of FedAvg with $E = 1$ vs. $E = 5$ local epochs. Explain the trade-off between communication efficiency and convergence stability.
2. **Gradient Inversion**:
   (a) Implement the DLG attack ([83]) in PyTorch.
   (b) Attempt to recover an image from a ResNet-18 gradient.
   (c) Investigate how the attack success depends on the batch size and the model depth.
3. **Backdoor Attacks**:
   (a) Implement a "pixel pattern" backdoor attack where a client inserts a trigger into their training images and flips the label.
   (b) Measure the "Attack Success Rate" (ASR) on the global model.
   (c) Test if Coordinate-wise Median aggregation defends against this attack.
4. **FedProx Implementation**:
   (a) Implement FedProx with proximal term $\frac{\mu}{2}\|w - w_t\|^2$ in PyTorch.
   (b) Compare convergence with FedAvg on highly non-IID data (Dirichlet($\alpha = 0.1$) partition).

(c) Study the effect of the proximal parameter μ on convergence speed and final accuracy.

(d) Analyze client drift by measuring the distance between local and global models during training.

5. **Differential Privacy in FL**:

(a) Implement DP-FedAvg by adding Gaussian noise to clipped model updates.

(b) Compute the privacy budget (ϵ, δ) using the moments accountant.

(c) Plot the privacy-accuracy trade-off for different noise scales σ.

(d) Compare user-level and sample-level DP guarantees.

6. **Secure Aggregation**:

(a) Implement a simplified secure aggregation protocol using Shamir's secret sharing.

(b) Demonstrate that the server cannot see individual updates but can compute the aggregate.

(c) Analyze the communication overhead compared to plain FedAvg.

(d) Test robustness to client dropout during the aggregation protocol.

7. **Personalized Federated Learning**:

(a) Implement FedPer: train shared lower layers globally, personalized upper layers locally.

(b) Compare with full fine-tuning and MAML-based personalization.

(c) Evaluate on CIFAR-10 with heterogeneous label distributions across clients.

(d) Measure both global test accuracy and per-client personalized accuracy.

8. **Byzantine-Robust Aggregation**:

(a) Implement Byzantine attacks: random updates, label flipping, model replacement.

(b) Compare robustness of FedAvg, Krum, Coordinate-wise Median, and Trimmed Mean.

(c) Determine the maximum fraction of Byzantine clients each aggregation can tolerate.

(d) Implement FLAME (filtering via model similarity) and evaluate against adaptive attacks.

10 Safe Reinforcement Learning

[15, 30]

10.1 The Alignment Problem in RL

Reinforcement Learning (RL) agents are optimization machines. They will exploit any loophole in the reward function to maximize their score, often leading to unintended and potentially harmful behavior. This phenomenon is known as "Reward Exploitation" or "Specification Gaming" [6].

Example 10.1 (The Coast Runners Effect)**.** In a boat racing game, an agent trained to maximize points found that it could achieve a higher score by spinning in circles and collecting respawning power-ups indefinitely, rather than finishing the race. This satisfied the literal reward function but violated the designer's intent.

Other alignment issues include:

- **Negative Side Effects**: An agent cleaning a room might break a vase because the reward function didn't explicitly penalize breaking things.
- **Reward Tampering**: The agent might modify its own code or the environment to artificially inflate the reward signal.

10.2 Constrained Markov Decision Processes (CMDP)

To enforce safety constraints (e.g., "do not crash"), we extend the standard MDP framework to Constrained MDPs. A CMDP is a tuple $\langle \mathcal{S}, \mathcal{A}, P, R, C, \gamma, d \rangle$, where:

- $C(s, a)$ is a cost function (e.g., $C(s, a) = 1$ if the action is unsafe, 0 otherwise).
- d is a safety threshold (e.g., expected cost must be ≤ 0.1).

The optimization problem becomes:

$$\max_{\pi} J_R(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \quad \text{s.t.} \quad J_C(\pi) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t)\right] \leq d \tag{179}$$

where J_R is the expected return, J_C is the expected cost, γ is the discount factor, and d is the safety threshold.

10.2.1 Lagrangian Relaxation

We can solve this using the method of Lagrange multipliers. We define the Lagrangian:

$$L(\pi, \lambda) = J_R(\pi) - \lambda(J_C(\pi) - d) \tag{180}$$

where $\lambda \geq 0$ is the Lagrange multiplier. The problem becomes a min-max game: $\min_{\lambda} \max_{\pi} L(\pi, \lambda)$. We can update π using policy gradients and λ using gradient descent (dual ascent).

10.2.2 Constrained Policy Optimization (CPO)

CPO [3] is a trust-region method (like TRPO) that explicitly enforces constraints at every update step. It solves a quadratic programming problem to find the best policy update that stays within the trust region and satisfies the safety constraints.

The CPO update is derived as follows. Let π_k be the current policy. TRPO maximizes the surrogate objective:

$$L_{\pi_k}(\pi) = \mathbb{E}_{s\sim d^{\pi_k}, a\sim\pi}\left[\frac{\pi(a|s)}{\pi_k(a|s)} A^{\pi_k}(s,a)\right] \tag{181}$$

where π_k is the current policy, d^{π_k} is the state visitation distribution, and A^{π_k} is the advantage function. subject to a KL constraint: $\bar{D}_{KL}(\pi_k\|\pi) \leq \delta$.

CPO adds the safety constraint in linearized form:

$$\max_{\theta} \quad g^T(\theta - \theta_k) \tag{182}$$

$$\text{s.t.} \quad \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta \tag{183}$$

$$c + b^T(\theta - \theta_k) \leq 0 \tag{184}$$

where g is the gradient of the objective, H is the Fisher information matrix, c is the constraint violation, b is the gradient of the constraint, and δ is the trust region size.

Solution via KKT Conditions. The Lagrangian for the CPO problem is:

$$\mathcal{L}(\theta, \lambda, \nu) = g^T(\theta - \theta_k) - \frac{\lambda}{2}(\theta - \theta_k)^T H(\theta - \theta_k) - \nu(c + b^T(\theta - \theta_k)) \tag{185}$$

where λ and ν are the dual variables for the trust region and safety constraints, respectively. Taking the derivative with respect to θ and setting to zero:

$$g - \lambda H(\theta - \theta_k) - \nu b = 0 \implies \theta - \theta_k = \frac{1}{\lambda} H^{-1}(g - \nu b) \tag{186}$$

where the update direction is a linear combination of the objective gradient and constraint gradient, scaled by the inverse Fisher matrix. The KKT complementary slackness conditions are: (1) $\lambda \geq 0$, $\nu \geq 0$; (2) if the trust region constraint is active, $\lambda > 0$; (3) if the safety constraint is active, $\nu > 0$ and $c + b^T(\theta - \theta_k) = 0$. The optimal λ, ν are found by solving the dual problem or using a line search that ensures both constraints are satisfied. When the safety constraint is violated ($c > 0$), the update is primarily driven by the constraint gradient b (recovery step). When safe ($c \leq 0$), the update can optimize the reward while staying within the safe region. □

The solution involves projecting onto the feasible region using the KKT conditions.

10.3 Safe Exploration

In standard RL, agents explore by taking random actions. In safety-critical domains (e.g., automated anesthesia dosing), random exploration is unacceptable.

10.3.1 Safety Layers via Control Barrier Functions

A Control Barrier Function (CBF) $h : \mathcal{S} \to \mathbb{R}$ defines the safe set $\mathcal{C} = \{s : h(s) \geq 0\}$.

Definition 10.1 (Control Barrier Function)**.** For a dynamical system $\dot{s} = f(s)+g(s)a$, a function h is a CBF if there exists $\alpha > 0$ such that:

$$\sup_a \left[\frac{\partial h}{\partial s}(f(s) + g(s)a)\right] \geq -\alpha h(s) \quad \forall s \in \mathcal{C} \tag{187}$$

where $h(s)$ is the barrier function, α is a class $\mathcal{K}$ function parameter, and the condition ensures the system can remain in the safe set $\mathcal{C}$.

Forward Invariance. The CBF condition ensures that the safe set $\mathcal{C} = \{s : h(s) \geq 0\}$ is forward invariant. If we choose an action satisfying the CBF constraint, then:

$$\dot{h}(s) = \frac{\partial h}{\partial s}(f(s) + g(s)a) \geq -\alpha h(s) \tag{188}$$

where $\dot{h}(s)$ is the time derivative of the barrier function along the system trajectory. This is a differential inequality. For any $s_0 \in \mathcal{C}$ (i.e., $h(s_0) \geq 0$), the solution satisfies:

$$h(s(t)) \geq h(s_0)e^{-\alpha t} \geq 0 \quad \forall t \geq 0 \tag{189}$$

where s_0 is the initial state and the inequality guarantees forward invariance of the safe set. Thus, starting from the safe set, the system remains in the safe set for all time. The exponential term $e^{-\alpha t}$ allows h to decay, but it can never become negative as long as we enforce the CBF constraint at each step. □

The safety layer solves a QP at each timestep:

$$a^* = \arg\min_a \quad \|a - a_{NN}\|^2 \tag{190}$$

$$\text{s.t.} \quad \frac{\partial h}{\partial s}(f(s) + g(s)a) \geq -\alpha h(s) \tag{191}$$

where a_{NN} is the nominal action and the constraint enforces the CBF condition.

10.3.2 Lyapunov-Based Safe Learning

A Lyapunov function $V : \mathcal{S} \to \mathbb{R}_{\geq 0}$ certifies stability if $V(s) = 0$ only at the equilibrium and $\dot{V}(s) < 0$ elsewhere.

Theorem 10.1 (Lyapunov Stability)**.** *If there exists a continuously differentiable function V such that:*

1. $V(s^*) = 0$ *at the equilibrium* s^*
2. $V(s) > 0$ *for all* $s \neq s^*$
3. $\dot{V}(s) = \nabla V(s)^T(f(s) + g(s)\pi(s)) < 0$ *for all* $s \neq s^*$

then the system is asymptotically stable under policy π.

Safe Lyapunov Learning jointly learns a policy π and Lyapunov function V such that the Lyapunov conditions are satisfied, ensuring stability throughout training.

10.3.3 Shielding

A shield is a verified backup policy that intervenes when the primary policy would violate safety constraints.

Definition 10.2 (Shield)**.** Given a primary policy π and a set of unsafe states $\mathcal{U}$, a shield σ modifies actions:

$$\sigma(\pi)(s) = \begin{cases} \pi(s) & \text{if safe}(s, \pi(s)) \\ \pi_{backup}(s) & \text{otherwise} \end{cases} \tag{192}$$

where π_{backup} is a verified safe policy and $\text{safe}(s, a)$ is a safety check.

10.4 Value Alignment and Inverse RL

Instead of manually specifying a potentially flawed reward function, we can try to learn the reward function from human behavior.

10.4.1 Inverse Reinforcement Learning (IRL)

Given a set of expert demonstrations $\mathcal{D} = \{(s_i, a_i)\}$, IRL aims to find a reward function R under which the expert's behavior is optimal.

Maximum Entropy IRL assumes the expert acts according to:

$$P(\tau) \propto \exp\left(\sum_t R(s_t, a_t)\right) \tag{193}$$

where τ is a trajectory and R is the reward function. The objective is to maximize the likelihood of demonstrations while preferring "simple" reward functions:

$$\max_R \mathbb{E}_{\tau \sim \mathcal{D}}[\log P(\tau|R)] - \lambda \|R\| \tag{194}$$

where $\mathcal{D}$ is the dataset of expert demonstrations and λ is a regularization coefficient.

Feature Matching IRL assumes $R(s) = w^T \phi(s)$ and finds weights such that the expected feature counts match between expert and learned policy:

$$\mathbb{E}_{\pi^*}[\phi(s)] = \mathbb{E}_{\pi_w}[\phi(s)] \tag{195}$$

where $\phi(s)$ are the state features, π^* is the expert policy, and π_w is the learned policy.

10.4.2 Preference Learning

Rather than demonstrations, we can learn from preferences over trajectory pairs.

Definition 10.3 (Bradley-Terry Model)**.** Given trajectory pairs (τ_1, τ_2) and human preference $\tau_1 \succ \tau_2$, the Bradley-Terry model assumes:

$$P(\tau_1 \succ \tau_2) = \frac{\exp(\sum_t R(s_t^1, a_t^1))}{\exp(\sum_t R(s_t^1, a_t^1)) + \exp(\sum_t R(s_t^2, a_t^2))} \tag{196}$$

where $\tau_1 \succ \tau_2$ denotes that trajectory τ_1 is preferred over τ_2.

The reward function is learned by maximizing the likelihood of observed preferences.

10.4.3 Reinforcement Learning from Human Feedback (RLHF)

RLHF is the technique used to align Large Language Models (like ChatGPT).

1. **Supervised Fine-Tuning (SFT)**: Train a supervised policy on expert demonstrations.
2. **Reward Model Training**: Collect comparison data (human prefers response A over B) and train:
$$\mathcal{L}_{RM} = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma(R_\theta(x,y_w) - R_\theta(x,y_l))\right] \tag{197}$$
where σ is the sigmoid function, y_w is the preferred response, and y_l is the rejected response.
3. **Policy Optimization**: Use PPO to maximize the reward while staying close to the SFT policy:
$$\max_\pi \mathbb{E}_{x\sim D, y\sim\pi}[R_\theta(x,y)] - \beta D_{KL}([\|\pi)\|\pi_{SFT}] \tag{198}$$
where π_{SFT} is the supervised fine-tuned policy and β controls the KL penalty.

10.4.4 Direct Preference Optimization (DPO)

DPO bypasses explicit reward modeling by directly optimizing the policy from preferences:

$$\mathcal{L}_{DPO}(\pi_\theta;\pi_{ref}) = -\mathbb{E}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w|x)}{\pi_{ref}(y_w|x)} - \beta\log\frac{\pi_\theta(y_l|x)}{\pi_{ref}(y_l|x)}\right)\right] \tag{199}$$

where π_{ref} is the reference policy, β is the implicit KL coefficient, and σ is the sigmoid function.

This is mathematically equivalent to RLHF with the optimal reward function under specific assumptions, but is computationally simpler.

10.5 Reward Exploitation and Specification Gaming

A fundamental challenge in RL is that agents optimize the specified reward, which may not capture the true objective.

Definition 10.4 (Reward Exploitation)**.** Reward Exploitation occurs when an agent achieves high reward through unintended means that do not align with the designer's goals.

Examples include a robot tasked with "move fast" that learns to be very tall and fall over, a game-playing agent that exploits bugs in the game physics, or a recommendation system that manipulates users to increase engagement metrics.

Mitigations: To address these issues, several strategies are employed:

- **Reward shaping**: Adds auxiliary rewards to guide learning towards desired behaviors.

  Detailed Explanation: Reward shaping modifies the original sparse reward $R(s,a)$ by adding a potential-based shaping function $F(s,s') = \gamma\Phi(s') - \Phi(s)$, where $\Phi : \mathcal{S} \to \mathbb{R}$ is a potential function encoding domain knowledge. This transformation preserves optimal policies (by the policy invariance theorem) while providing denser learning signals. The key insight is that F encodes an admissible heuristic that guides the agent toward rewarding states without changing which policies are optimal. Careful design of Φ can dramatically accelerate learning in sparse-reward environments.

  Concrete Example: In robotic navigation to a goal location s_g, the sparse reward $R(s,a) = \mathbb{1}[s = s_g]$ provides no guidance. Define $\Phi(s) = -\|s - s_g\|$ (negative distance to goal).

Then $F(s, s') = -\gamma\|s' - s_g\| + \|s - s_g\|$ rewards moving closer to the goal at each step. For a grid world with goal at $(5, 5)$, moving from $(3, 3)$ to $(4, 4)$ receives shaped reward $F = -0.9 \cdot \sqrt{2} + 2\sqrt{2} \approx 1.55$, accelerating learning while preserving the optimal path.

- **Constrained RL**: Explicitly constrains undesirable behaviors using cost functions.

  Detailed Explanation: Constrained RL formulates safety as constraint satisfaction in the CMDP framework: maximize $\mathbb{E}[\sum_t \gamma^t r_t]$ subject to $\mathbb{E}[\sum_t \gamma^t c_i(s_t, a_t)] \leq d_i$ for cost functions $c_1, \ldots, c_k$. This separates the objective (reward maximization) from safety requirements (constraint satisfaction), making specifications clearer. The Lagrangian relaxation approach introduces dual variables λ_i and optimizes $\mathbb{E}[\sum_t \gamma^t (r_t - \sum_i \lambda_i c_i(s_t, a_t))]$, adaptively adjusting λ_i to meet constraints. Unlike reward shaping, this guarantees constraint satisfaction rather than merely encouraging it.

  Concrete Example: In automated insulin infusion, the reward function optimizes time in target blood glucose range: $r_t = v_t$ (time in range). Safety constraints prevent hypoglycemia: $c_1(s, a) = \mathbb{1}$[blood glucose < 70mg/dL] with threshold $d_1 = 0.05$ (allowing a very small violation rate during early simulated training). Another constraint limits insulin dosage change rate for metabolic stability: $c_2(s, a) = |a_t - a_{t-1}|$ with $d_2 = 0.3$. CPO algorithms solve this constrained optimization, ensuring the learned policy is both effective and safe, even when rapid correction conflicts with stability.

- **Iterated amplification**: Recursively decomposes complex tasks to reduce specification error.

  Detailed Explanation: Iterated amplification addresses reward specification in complex domains where direct reward engineering is infeasible. The key idea: train an agent A_1 on a simplified task with a trusted reward, then use human+A_1 (human assisted by A_1's suggestions) to provide supervision for more complex subtasks, training A_2. Recursively, A_n is trained using human+A_{n-1} oversight. This "bootstraps" capability while maintaining alignment, as each step is overseen by the amplified human-agent system. The process converges when A_n matches the capability of human+A_{n-1}, ideally preserving alignment throughout.

  Concrete Example: For training a medical diagnosis AI: A_1 learns to highlight abnormal regions in X-rays (easy for humans to verify). Human+A_1 (human using the abnormal region highlighter) reviews more complex scans for specific disease indicators, training A_2. Human+A_2 (human using region and disease checkers) reviews comprehensive patient histories, training A_3. After several iterations, A_n performs comprehensive medical diagnosis. At each stage, the human provides meaningful oversight because A_{n-1} handles routine checks, allowing focus on higher-level issues. This decomposes the intractable task "diagnose perfectly" into manageable supervised learning problems.

- **AI safety via debate**: Uses adversarial processes where agents argue for different actions to help a human judge identify Reward Exploitation.

  Detailed Explanation: In debate-based AI safety, two agents with opposing incentives present arguments for different answers to a question or different actions in a situation. A human judge evaluates the debate and selects the more convincing argument. The key insight: even if individual arguments are too complex for the human to verify directly, adversarial debate incentivizes each agent to find flaws in the opponent's reasoning. This works when "verification is easier than generation"—it's hard to construct a correct proof, but errors can be efficiently identified by an adversary. Training via debate makes agents optimize for defensible, robust decisions rather than exploiting judge biases.

  Concrete Example: To decide whether an expensive medical procedure is necessary, Agent A argues "Approve: the patient's symptoms heavily align with a condition requiring this

procedure" while Agent B argues "Reject: the patient's recent lab results suggest a milder, alternative condition." In cross-examination, Agent B reveals that Agent A cherry-picked symptom data from an earlier phase of the illness. The human judge (a clinician), unable to review the entire multi-year patient history in detail, can evaluate this specific claim. Through repeated debates on many cases, agents learn that deceptive arguments (exploiting cherry-picking) get caught by adversaries, incentivizing honest, comprehensive analysis. This scales human oversight beyond direct review capacity.

Exercises

1. **Gridworld Safety**:
   (a) Design a Gridworld environment with "lava" states that terminate the episode with a large negative reward.
   (b) Train a standard Q-learning agent. Observe how many times it falls into the lava during training.
   (c) Implement a "Safety Layer" that masks out actions leading to lava (assuming the dynamics are known). Compare the training safety.
2. **Lagrangian Relaxation**:
   (a) Implement a PPO agent for the CartPole environment.
   (b) Add a constraint: The cart position must not exceed $x = 1.0$ (standard limit is 2.4).
   (c) Use the Lagrangian method to enforce this constraint. Plot the reward and the constraint violation over time.
3. **RLHF Toy Example**:
   (a) Define a ground truth reward function (e.g., $R(x) = -|x - 5|$).
   (b) Generate "human preferences" based on this function (with some noise).
   (c) Train a reward model (neural network) to predict these preferences.
   (d) Optimize an agent using the learned reward model and compare the result to the ground truth optimum.
4. **Control Barrier Functions**:
   (a) Design a simple 2D navigation environment with a circular obstacle.
   (b) Define a CBF $h(s) = \|s - s_{obs}\|^2 - r^2$ where s_{obs} is the obstacle center and r is the safety radius.
   (c) Implement a safety layer that solves the QP at each step to ensure $\dot{h}(s) \geq -\alpha h(s)$.
   (d) Compare trajectories with and without the safety layer. Visualize the invariant set.
5. **Constrained Policy Optimization**:
   (a) Implement CPO for the Safety Gym benchmark (e.g., PointGoal with hazards).
   (b) Compare convergence and constraint satisfaction against vanilla PPO and Lagrangian PPO.
   (c) Plot the Pareto frontier of reward vs. cost for different constraint thresholds.
   (d) Analyze how the trust region size affects both reward and constraint violation.
6. **Reward Modeling**:

(a) Collect synthetic preference data by comparing trajectories based on multiple criteria: (1) distance to goal, (2) smoothness of path, (3) energy consumption.

(b) Train reward models with different architectures: MLP, LSTM, Transformer.

(c) Evaluate reward model accuracy on held-out preferences.

(d) Investigate the impact of label noise (disagreeing annotators) on reward model quality.

7. **Safe Exploration Algorithms**:

(a) Implement Safe Model-Based Policy Optimization (SMBPO) using an ensemble of dynamics models.

(b) Compare exploration efficiency and safety violations against TRPO-Lagrangian.

(c) Analyze how pessimism in the uncertainty estimation affects exploration.

(d) Test on continuous control tasks with safety constraints (e.g., joint limits, velocity bounds).

8. **DPO Implementation**:

(a) Implement Direct Preference Optimization for a small language model.

(b) Compare training dynamics with RLHF using PPO.

(c) Measure the alignment tax: how much task performance degrades after alignment.

(d) Investigate the effect of the KL coefficient β on alignment strength and diversity.

11 Out-of-Distribution Detection and Uncertainty Quantification

[32, 45]

A trustworthy AI system must recognize when it is operating outside its domain of competence. Out-of-Distribution (OOD) detection and uncertainty quantification provide mechanisms for AI systems to express confidence calibrated to their actual reliability, enabling appropriate human intervention when the system encounters unfamiliar inputs.

11.1 The Problem of Distribution Shift

Machine learning models are trained on data from a distribution $P_{train}(X, Y)$. At deployment, the model encounters data from a potentially different distribution $P_{test}(X, Y)$. When these distributions differ, model performance can degrade significantly, often without any indication to the user.

Definition 11.1 (Types of Distribution Shift)**.**

- **Covariate Shift**: $P_{test}(X) \neq P_{train}(X)$ but $P_{test}(Y|X) = P_{train}(Y|X)$. The input distribution changes, but the input-output relationship is preserved.

  Detailed Explanation: In covariate shift, the fundamental relationship between features and labels remains unchanged; what changes is the relative frequency of different types of inputs. Mathematically, the conditional probability $P(Y|X)$ is invariant, meaning if we observe the same feature vector x in both training and test sets, the probability distribution over labels remains identical.

  Concrete Example: Consider a hospital readmission risk model trained on patients from a general hospital, where the patient age distribution was balanced and mostly younger. At deployment in a specialized geriatric facility, the age distribution shifts significantly toward elderly individuals. The fundamental physiological indicators that make a patient likely to be readmitted (such as specific comorbidities or degraded vital signs) haven't changed their conditional meaning; $P(Y = \text{readmission}|X)$ remains the same, but the overall distribution of features (age, baseline health) has shifted. Importance weighting can correct for this: we reweight training examples by $w(x) = P_{test}(X)/P_{train}(X)$, giving more weight to examples similar to test distribution. This is solvable through density ratio estimation or by training a classifier to distinguish training from test data and using predicted probabilities as weights.

- **Label Shift**: $P_{test}(Y) \neq P_{train}(Y)$ but $P_{test}(X|Y) = P_{train}(X|Y)$. The class proportions change, but the class-conditional distributions are preserved.

  Detailed Explanation: Label shift occurs when the prevalence of different classes changes between training and deployment, but the characteristics of each class remain consistent. The class-conditional densities $P(X|Y = y)$ are identical in both domains, meaning examples labeled y "look the same" statistically.

  Concrete Example: A medical diagnostic system trained on hospital data where 5% of patients had a rare disease is deployed in a specialized clinic where 30% of patients have the disease (due to referral bias; the clinic specializes in that condition). The symptoms and test results associated with the disease ($P(X|Y = \text{disease})$) haven't changed, but the base rate has increased dramatically. The confusion matrix method can estimate the new class proportions: using the classifier's predicted labels on test data and assuming the confusion matrix (calibrated on validation set) remains valid, we solve for $P_{test}(Y)$ via $P_{test}(\hat{Y}) =$

$C \cdot P_{test}(Y)$ where C is the confusion matrix. The solution $P_{test}(Y) = C^{-1}P_{test}(\hat{Y})$ allows recalibration of predictions.

- **Concept Drift**: $P_{test}(Y|X) \neq P_{train}(Y|X)$. The relationship between inputs and outputs changes.

  Detailed Explanation: Concept drift represents a fundamental change in the underlying data-generating process; the same input features now map to different labels, or the decision boundary has shifted. This is the most challenging type of shift because the training data becomes misleading: past patterns no longer apply.

  Concrete Example: An epidemiological forecasting model learned in 2019 that a specific cluster of respiratory symptoms ($X_1 > \theta$) combined with high fever strongly predicted the standard seasonal flu ($Y = \text{flu}$). During a novel viral pandemic in 2020, the relationship changed fundamentally: that exact same symptom presentation often preceded the novel virus instead. The physiological features X_1 remain measurable, but $P(Y = \text{flu}|X_1 > \theta)$ decreased from 0.8 to 0.2, while the probability of the novel virus increased. Another example: a medical text classifier trained pre-2020 might have learned that specific isolation terminology indicated a highly unusual infectious disease case, but during the pandemic, routine hospital admission notes frequently used identical isolation protocols. No reweighting can fix this; the model must be retrained with new labeled data. Concept drift detection monitors statistics like prediction accuracy over sliding time windows; when performance drops significantly, it triggers model retraining or human review.

- **Domain Shift**: Both the marginal and conditional distributions change simultaneously: $P_{test}(X) \neq P_{train}(X)$ and $P_{test}(Y|X) \neq P_{train}(Y|X)$.

  Detailed Explanation: Domain shift combines multiple types of distribution changes, making it the most severe and general form of shift. Neither the input statistics nor the predictive relationships are preserved.

  Concrete Example: A dermatology image classification system trained on high-resolution hospital scanners (consistent clinical lighting, standardized camera angles, specific demographic skin tone prevalence) is deployed in rural field clinics using mobile phone cameras (varying ambient lighting, shadows, lower resolution, and a different distribution of patient skin tones). Both $P(X)$ changes (different visual statistics: lighting, noise, resolution) and $P(Y|X)$ changes (a particular visual pattern might indicate a "lesion" in hospital scanner images but represent merely a "shadow artifact" in a mobile camera image). Domain adaptation techniques become necessary: unsupervised domain adaptation aligns source and target feature distributions using adversarial training ($\min_f \max_D \mathbb{E}_{\text{source}}[\log D(f(x))] + \mathbb{E}_{\text{target}}[\log(1 - D(f(x)))]$), self-training uses high-confidence predictions on target domain as pseudo-labels, or multi-task learning jointly trains on both domains with shared representations.

11.2 Approaches to OOD Detection

11.2.1 Softmax-Based Methods

The simplest approach uses the maximum softmax probability (MSP) as a confidence score [29]. For a classifier f with softmax output, the OOD score is:

$$S_{MSP}(x) = 1 - \max_c P(Y = c|X = x) = 1 - \max_c \text{softmax}(f(x))_c \tag{200}$$

where $f(x)$ are the logits and c indexes the classes.

However, deep networks are known to produce overconfident predictions even on OOD data. The softmax function normalizes logits to sum to 1, which can produce high confidence regardless of input proximity to training data.

Temperature Scaling adjusts the softmax temperature to improve calibration:

$$P(Y = c|X = x; T) = \frac{\exp(z_c/T)}{\sum_{c'} \exp(z_{c'}/T)} \tag{201}$$

where z_c are the logits and T is the temperature parameter.

11.2.2 ODIN: Out-of-Distribution Detector for Neural Networks

ODIN [41] combines temperature scaling with input preprocessing:

$$\tilde{x} = x - \epsilon \cdot \text{sign}(\nabla_x \log \max_c P(Y = c|X = x; T)) \tag{202}$$

where ϵ is the perturbation magnitude and the gradient maximizes the softmax probability.

The perturbation $\tilde{x}$ is designed to increase the softmax score for in-distribution data while having less effect on OOD data. The gradient points toward the decision boundary, pushing in-distribution samples further into high-confidence regions.

11.2.3 Energy-Based OOD Detection

The energy function [42] provides a theoretically grounded alternative to softmax probability:

$$E(x) = -T \cdot \log \sum_c \exp(f_c(x)/T) \tag{203}$$

where $f_c(x)$ is the logit for class c.

In-distribution samples have lower energy (higher density under the model), while OOD samples have higher energy. This approach is connected to the log-partition function in energy-based models.

Theorem 11.1 (Energy and Density)**.** *For a classifier trained with cross-entropy loss, the energy function is proportional to the negative log-likelihood under a joint energy-based model:*

$$E(x) = -T \log p(x) + \textit{const} \tag{204}$$

where $p(x)$ is the data density and T is the temperature.

Proof Sketch. Consider the energy-based formulation where the joint distribution is:

$$p(x, y) = \frac{\exp(f_y(x)/T)}{Z} \tag{205}$$

where Z is the partition function. The marginal density of x is:

$$p(x) = \sum_y p(x, y) = \frac{1}{Z} \sum_y \exp(f_y(x)/T) = \frac{1}{Z} \exp\left(\frac{-E(x)}{T}\right) \tag{206}$$

where $E(x) = -T \log \sum_y \exp(f_y(x)/T)$ is the free energy. Taking the log:

$$\log p(x) = -\frac{E(x)}{T} - \log Z \implies E(x) = -T \log p(x) + \text{const} \tag{207}$$

where the relationship links energy to negative log-likelihood. Thus, the energy function is proportional to the negative log-density. During training with cross-entropy, the model implicitly learns to assign lower energy to in-distribution samples, making energy a natural OOD score. □

11.2.4 Deep Ensembles

Training multiple models with different random initializations provides a measure of model uncertainty:

$$\text{Var}[\hat{y}] = \frac{1}{M}\sum_{m=1}^{M}(f_m(x) - \bar{f}(x))^2 \tag{208}$$

where M is the number of models and f_m is the prediction of the m-th model.

High disagreement among ensemble members indicates uncertainty, which is often elevated for OOD inputs. Deep ensembles capture *epistemic uncertainty* arising from model uncertainty, not just *aleatoric uncertainty* from inherent data noise.

11.2.5 Mahalanobis Distance-Based Detection

Fit a class-conditional Gaussian to the penultimate layer features:

$$\mu_c = \frac{1}{|D_c|}\sum_{x \in D_c} h(x), \quad \Sigma = \frac{1}{n}\sum_{c}\sum_{x \in D_c}(h(x) - \mu_c)(h(x) - \mu_c)^T \tag{209}$$

where $h(x)$ are the features, D_c is the set of samples in class c, and Σ is the tied covariance matrix.

The Mahalanobis distance to the nearest class centroid serves as OOD score:

$$M(x) = \min_c (h(x) - \mu_c)^T \Sigma^{-1}(h(x) - \mu_c) \tag{210}$$

where $M(x)$ is the minimum distance to any class centroid.

11.3 Bayesian Approaches to Uncertainty

11.3.1 Bayesian Neural Networks

Instead of point estimates θ^*, Bayesian neural networks maintain a posterior distribution $P(\theta|D)$. Predictions are made by marginalizing over the posterior:

$$P(y|x, D) = \int P(y|x, \theta)P(\theta|D)d\theta \tag{211}$$

where $P(\theta|D)$ is the posterior distribution over parameters. Since exact inference is intractable, various approximations are used. **Variational Inference** approximates $P(\theta|D)$ with a tractable distribution $q_\phi(\theta)$ by minimizing KL divergence. **Monte Carlo Dropout** uses dropout at test time to sample from an approximate posterior. **Laplace Approximation** approximates the posterior with a Gaussian centered at the MAP estimate.

11.3.2 Monte Carlo Dropout

Dropout at test time provides an approximation to Bayesian inference. For T forward passes with dropout:

$$\hat{y}(x) = \frac{1}{T}\sum_{t=1}^{T} f(x; \theta_t) \tag{212}$$

where T is the number of forward passes and θ_t are the dropout-masked parameters.

$$\text{Var}[\hat{y}] \approx \frac{1}{T}\sum_{t=1}^{T} f(x;\theta_t)^2 - \left(\frac{1}{T}\sum_{t=1}^{T} f(x;\theta_t)\right)^2 \tag{213}$$

where the variance estimates the predictive uncertainty.

Theorem 11.2 (Gal & Ghahramani)**.** *Training with dropout and L_2 regularization is mathematically equivalent to variational inference in a Bayesian neural network with specific prior and variational family.*

Proof Sketch. Consider a neural network with dropout applied to each layer with dropout rate p. Let $\mathbf{z}_i \in \{0,1\}^{d_i}$ be binary dropout masks for layer i. The variational distribution is:

$$q(\mathbf{w}) = \prod_i q(\mathbf{w}_i|\mathbf{z}_i) = \prod_i p^{d_i}\delta_{\mathbf{W}_i \text{diag}(\mathbf{z}_i)}(\mathbf{w}_i) \tag{214}$$

where $\mathbf{z}_i$ are the dropout masks and δ is the Dirac delta function.

$$\mathcal{L} = -\sum_{(x,y)} \mathbb{E}_{q(\mathbf{w})}[\log p(y|x,\mathbf{w})] + D_{KL}([\|q)(\mathbf{w})\|p(\mathbf{w})] \tag{215}$$

where the first term is the expected log-likelihood and the second is the KL divergence to the prior. With a Gaussian prior $p(\mathbf{w}) \propto \exp(-\|\mathbf{w}\|^2/(2\tau^2))$ and approximating the expectation with dropout sampling, the objective becomes:

$$\mathcal{L} \approx \sum_{(x,y)} \ell(y, f(x; \mathbf{W}\odot\mathbf{z})) + \frac{\lambda}{2}\|\mathbf{W}\|^2 \tag{216}$$

where ℓ is the loss function and λ is the weight decay coefficient. □

11.3.3 Decomposing Uncertainty

Total predictive uncertainty can be decomposed into:

$$\underbrace{\text{Var}[y|x]}_{\text{Total}} = \underbrace{\mathbb{E}[\text{Var}[y|x,\theta]]}_{\text{Aleatoric}} + \underbrace{\text{Var}[\mathbb{E}[y|x,\theta]]}_{\text{Epistemic}} \tag{217}$$

where the total variance is split into data uncertainty (aleatoric) and model uncertainty (epistemic). **Aleatoric uncertainty** refers to inherent noise in the data that cannot be reduced with more data, while **Epistemic uncertainty** represents model uncertainty that can be reduced with more data or better models. For OOD detection, epistemic uncertainty is most relevant since OOD inputs should have high model uncertainty.

11.4 Calibration

Definition 11.2 (Calibration)**.** A probabilistic classifier is **calibrated** if among all predictions with confidence p, the fraction of correct predictions is p:

$$P(Y = \hat{Y}|P(\hat{Y}) = p) = p \quad \forall p \in [0,1] \tag{218}$$

where $\hat{Y}$ is the predicted class and p is the confidence.

11.4.1 Expected Calibration Error (ECE)

ECE measures the average gap between confidence and accuracy:

$$\text{ECE} = \sum_{b=1}^{B} \frac{|B_b|}{n} |\text{acc}(B_b) - \text{conf}(B_b)| \tag{219}$$

where B is the number of bins, B_b is the set of samples in bin b, and n is the total number of samples.

11.4.2 Reliability Diagrams

A reliability diagram plots accuracy vs. confidence for each bin. A perfectly calibrated model lies on the diagonal $y = x$.

11.4.3 Post-hoc Calibration Methods

Temperature Scaling: Learn a single temperature T to minimize negative log-likelihood on a validation set:

$$T^* = \arg\min_T - \sum_{(x,y)\in D_{val}} \log P(Y = y|X = x; T) \tag{220}$$

where D_{val} is the validation set.

Platt Scaling: Fit a logistic regression on the logits:

$$P(Y = 1|z) = \sigma(az + b) \tag{221}$$

where z is the logit and a, b are learned scaling parameters.

Isotonic Regression: Fit a non-parametric isotonic (monotonically increasing) function mapping predicted probabilities to calibrated probabilities.

11.5 Conformal Prediction

Conformal prediction provides distribution-free, finite-sample valid prediction sets. Given a desired coverage level $1 - \alpha$, conformal prediction outputs a set $C(x)$ that contains the true label with probability at least $1 - \alpha$.

Definition 11.3 (Conformal Prediction Set)**.** Given calibration data $\{(x_i, y_i)\}_{i=1}^n$ and a conformity score function $s(x, y)$:

1. Compute scores $s_i = s(x_i, y_i)$ for calibration data
2. Find threshold $\hat{q} = \text{Quantile}_{(n+1)(1-\alpha)/n}(\{s_1, \ldots, s_n\})$
3. For new input x, output $C(x) = \{y : s(x, y) \leq \hat{q}\}$

Theorem 11.3 (Coverage Guarantee)**.** *If $(x_1, y_1), \ldots, (x_n, y_n), (x_{n+1}, y_{n+1})$ are exchangeable, then:*

$$P(y_{n+1} \in C(x_{n+1})) \geq 1 - \alpha \tag{222}$$

where $C(x_{n+1})$ is the prediction set and $1 - \alpha$ is the target coverage.

Proof. Define the conformity scores $s_i = s(x_i, y_i)$ for $i = 1, \ldots, n+1$. By exchangeability, all permutations of $(s_1, \ldots, s_{n+1})$ are equally likely. The quantile threshold $\hat{q}$ is defined as the $\lceil (n+1)(1-\alpha) \rceil$-th smallest value in the augmented set $\{s_1, \ldots, s_n, \infty\}$.

$$\hat{q} = \text{Smallest}_{\lceil (n+1)(1-\alpha) \rceil}(\{s_1, \ldots, s_n\} \cup \{\infty\}) \tag{223}$$

where $\hat{q}$ is the conformal threshold. The prediction set is $C(x_{n+1}) = \{y : s(x_{n+1}, y) \leq \hat{q}\}$. We include y_{n+1} if and only if $s_{n+1} \leq \hat{q}$. Consider the rank R of s_{n+1} among $\{s_1, \ldots, s_{n+1}\}$ (ties broken randomly). By exchangeability:

$$P(R = k) = \frac{1}{n+1} \quad \forall k \in \{1, \ldots, n+1\} \tag{224}$$

where R is the rank of the test score among calibration scores. We have $s_{n+1} \leq \hat{q}$ if and only if $R \leq \lceil (n+1)(1-\alpha) \rceil$. Therefore:

$$P(y_{n+1} \in C(x_{n+1})) = P(R \leq \lceil (n+1)(1-\alpha) \rceil) = \frac{\lceil (n+1)(1-\alpha) \rceil}{n+1} \geq 1 - \alpha \tag{225}$$

where the probability is over the randomness of the calibration and test data. This proves the coverage guarantee holds for any exchangeable sequence, without any assumptions on the model or data distribution. □

The key insight is that conformal prediction makes no distributional assumptions beyond exchangeability, providing valid coverage even when the model is misspecified.

Example 11.1 (Conformal Prediction for Drug Dosage Recommendation)**.** A model predicts optimal drug dosage (mg) for patients based on weight, age, and kidney function. Using $1-\alpha = 0.95$ and $n = 200$ calibration patients, we compute conformity scores $s_i = |y_i - \hat{y}_i|$ (absolute residuals). Sorting these, the 95th percentile threshold is $\hat{q} = 12.3$ mg. For a new patient with predicted dosage $\hat{y} = 85$ mg, the prediction interval is $C(x) = [85 - 12.3,\ 85 + 12.3] = [72.7, 97.3]$ mg. The coverage guarantee ensures that for at least 95% of future patients, the true optimal dosage falls within the interval, regardless of the model's accuracy. If the model is poor, the intervals simply become wider (e.g., $\hat{q} = 40$ mg), honestly reflecting the model's uncertainty. The clinician can then decide whether the interval is narrow enough for safe prescription or whether additional tests are needed.

11.6 Selective Prediction (Rejection)

In selective prediction, the model can abstain from making a prediction when uncertain:

$$g(x) = \begin{cases} f(x) & \text{if } \kappa(x) \geq \tau \\ \perp \text{ (abstain)} & \text{otherwise} \end{cases} \tag{226}$$

where $\kappa(x)$ is the confidence function and τ is the rejection threshold.

Risk-Coverage Trade-off: Let the coverage be $\phi(\tau) = P(\kappa(x) \geq \tau)$ and the selective risk be:

$$R(\tau) = \frac{\mathbb{E}[\ell(f(x), y) \cdot \mathbb{I}(\kappa(x) \geq \tau)]}{\phi(\tau)} \tag{227}$$

where $\phi(\tau)$ is the coverage and ℓ is the loss function.

The risk-coverage curve shows how accuracy improves as coverage decreases (the model abstains on more inputs).

Exercises

1. **OOD Detection Benchmark**:
   (a) Train a classifier on CIFAR-10 (in-distribution).
   (b) Evaluate OOD detection using SVHN and CIFAR-100 as OOD datasets.
   (c) Compare MSP, ODIN, Energy, and Mahalanobis methods using AUROC.
2. **Calibration**:
   (a) Train a ResNet on CIFAR-10 and plot its reliability diagram.
   (b) Apply temperature scaling and re-plot. Calculate ECE before and after.
   (c) Investigate how calibration changes under distribution shift.
3. **Conformal Prediction**:
   (a) Implement conformal prediction for a multi-class classifier.
   (b) Verify the coverage guarantee empirically by measuring coverage on held-out data.
   (c) Plot the average prediction set size vs. coverage level $1 - \alpha$.
4. **Bayesian Neural Networks**:
   (a) Implement Monte Carlo dropout and compare uncertainty estimates with a deep ensemble.
   (b) Visualize the predictive uncertainty on a 2D toy classification problem.
   (c) Show that epistemic uncertainty is high in regions far from training data.
5. **Selective Prediction**:
   (a) Implement a selective classifier that abstains when confidence is below a threshold.
   (b) Plot the risk-coverage curve for different threshold values.
   (c) Compare the area under the risk-coverage curve (AURC) for different confidence functions: MSP, entropy, MC dropout variance.
   (d) Implement the SelectiveNet architecture that jointly learns prediction and rejection functions.
6. **Energy-Based OOD Detection**:
   (a) Implement the energy score $E(x) = -T \log \sum_c \exp(f_c(x)/T)$.
   (b) Compare with MSP on various OOD benchmarks (CIFAR-10 vs SVHN, CIFAR-100, Textures).
   (c) Visualize the energy distribution for in-distribution vs OOD samples.
   (d) Implement energy-based fine-tuning to improve OOD detection.
7. **Mahalanobis Distance Detection**:
   (a) Extract features from multiple layers of a pre-trained network.
   (b) Fit class-conditional Gaussians to the training features.
   (c) Implement the Mahalanobis distance score with input preprocessing.
   (d) Compare single-layer vs multi-layer ensemble for OOD detection.
8. **Advanced Conformal Prediction**:

(a) Implement Adaptive Conformal Inference (ACI) for handling distribution shift.

(b) Compare split conformal, cross-conformal, and full conformal prediction.

(c) Implement conformalized quantile regression for uncertainty estimation.

(d) Analyze the trade-off between average set size and coverage under covariate shift.

9. **Uncertainty Decomposition**:

(a) Implement both aleatoric and epistemic uncertainty estimation using a heteroscedastic neural network.

(b) Train a model to output both mean and variance predictions.

(c) Visualize the two uncertainty types on a regression problem with varying noise levels.

(d) Show that epistemic uncertainty decreases with more training data while aleatoric remains constant.

12 Engineering and Governance

12.1 Formal Verification of Neural Networks

As neural networks are deployed in safety-critical applications (robotic surgical systems, automated medical devices, advanced life support), there is a growing need for mathematical proofs of their properties. Formal verification aims to provide such proofs, guaranteeing that a neural network satisfies a specification for all inputs in some domain.

12.1.1 The Verification Problem

Definition 12.1 (Neural Network Verification)**.** Given a neural network $f : \mathbb{R}^d \to \mathbb{R}^k$, an input region $\mathcal{C} \subseteq \mathbb{R}^d$, and a property specification $\phi : \mathbb{R}^k \to \{\text{True}, \text{False}\}$, the verification problem asks:

$$\forall x \in \mathcal{C} : \phi(f(x)) = \text{True} \tag{228}$$

where $\mathcal{C}$ is the input region and ϕ is the property specification.

Common verification properties include **Local Robustness** ($\forall x' : \|x' - x\|_p \leq \epsilon \implies f(x') = f(x)$), **Output Bounds** ($\forall x \in \mathcal{C} : f(x) \in [l, u]$), and **Safety Constraints** ($\forall x \in \mathcal{C} : f(x) \in \mathcal{S}$ for safe output set $\mathcal{S}$).

12.1.2 Complexity of Verification

Theorem 12.1 ([34])**.** *The problem of verifying properties of neural networks with ReLU activations is NP-complete.*

This means that exact verification is computationally intractable for large networks in the worst case. Practical approaches either provide sound but incomplete verification (may fail to prove true properties) or use approximations that trade tightness for scalability.

12.1.3 Bound Propagation Methods

The key idea is to propagate bounds through the network layer by layer. Given input bounds $x \in [l^{(0)}, u^{(0)}]$, we compute bounds for each layer's pre-activations and post-activations.

Interval Bound Propagation (IBP):

Detailed Explanation: IBP propagates interval bounds through each layer by computing the tightest possible output interval given input intervals and layer operations. For affine transformations, this is exact and tractable. For a linear layer $z = Wx + b$, we split the weight matrix W into positive and negative components: $W^+ = \max(W, 0)$ contains all positive weights (negative entries zeroed), and $W^- = \min(W, 0)$ contains all negative weights (positive entries zeroed). To compute lower bounds on the output, we use lower bounds of inputs multiplied by positive weights (both small $\to$ small product) and upper bounds of inputs multiplied by negative weights (large magnitude $\times$ negative $\to$ small result). The formulas are:

$$l^{(i+1)} = W^+ l^{(i)} + W^- u^{(i)} + b \tag{229}$$

$$u^{(i+1)} = W^+ u^{(i)} + W^- l^{(i)} + b \tag{230}$$

where $l^{(i)}, u^{(i)}$ are the lower and upper bounds at layer i, and W^+, W^- are the positive and negative weight matrices.

For ReLU activations: $l' = \max(l, 0)$, $u' = \max(u, 0)$. If the input interval $[l, u]$ is entirely positive, ReLU is identity; if entirely negative, output is $[0, 0]$; if $l < 0 < u$, output is $[0, u]$.

IBP is computationally efficient (O(network size), linear in parameters) but produces loose bounds because it treats each neuron independently, ignoring correlations. For example, if $z_1 = x$ and $z_2 = -x$ with $x \in [-1, 1]$, IBP computes $z_1 + z_2 \in [-2, 2]$, while the true range is exactly $\{0\}$.

Concrete Example: Consider a 2-layer network with input $x \in [0, 1]$, first layer $z_1 = 2x - 1$, ReLU, then $z_2 = -3 \cdot \text{ReLU}(z_1) + 1$.

- Layer 1: $z_1 = 2x - 1$ with $x \in [0, 1]$ gives $z_1 \in [2 \cdot 0 - 1, 2 \cdot 1 - 1] = [-1, 1]$.
- ReLU: $\text{ReLU}(z_1) \in [\max(-1, 0), \max(1, 0)] = [0, 1]$.
- Layer 2: $z_2 = -3 \cdot \text{ReLU}(z_1) + 1$. Here $W = -3$ (negative), so $W^+ = 0$, $W^- = -3$. Lower bound: $l^{(2)} = 0 \cdot 0 + (-3) \cdot 1 + 1 = -2$. Upper bound: $u^{(2)} = 0 \cdot 1 + (-3) \cdot 0 + 1 = 1$. Thus $z_2 \in [-2, 1]$.

IBP proves the output range is $[-2, 1]$, useful for checking if $z_2 > 0$ always holds (answer: no, verification fails).

Linear Relaxation:

Detailed Explanation: Tighter bounds can be obtained by relaxing non-linear activations with linear upper and lower bounds, preserving more relational information between neurons. For ReLU on interval $[l, u]$:

- If $u \leq 0$: $\text{ReLU}(z) = 0$ (neuron always inactive)
- If $l \geq 0$: $\text{ReLU}(z) = z$ (neuron always active)
- If $l < 0 < u$: neuron may be active or inactive, requiring approximation

The tightest convex relaxation of ReLU in the unstable case ($l < 0 < u$) is the "triangle relaxation":

$$\text{ReLU}(z) \geq 0, \quad \text{ReLU}(z) \geq z, \quad \text{ReLU}(z) \leq \frac{u}{u - l}(z - l) \tag{231}$$

where l, u are the pre-activation bounds and the inequalities define the convex hull of the ReLU function.

The upper bound is a line connecting $(l, 0)$ and (u, u), which is the tightest linear upper bound containing the ReLU curve. The lower bounds express that ReLU is non-negative and greater than its input. This convex relaxation allows propagating linear constraints through the network using linear programming, tracking dependencies between neurons rather than just intervals.

Concrete Example: For ReLU with $z \in [-2, 3]$ (unstable neuron):

- Lower bounds: $\text{ReLU}(z) \geq 0$ and $\text{ReLU}(z) \geq z$ (second is inactive in this range).
- Upper bound: $\text{ReLU}(z) \leq \frac{3}{3-(-2)}(z - (-2)) = \frac{3}{5}(z + 2) = 0.6z + 1.2$.

For input $z = 1$, the true value is $\text{ReLU}(1) = 1$, while the upper bound gives $0.6(1) + 1.2 = 1.8$, so $\text{ReLU}(1) \leq 1.8$ (relaxation introduces slack but maintains soundness). For $z = -1$, true value is 0, upper bound is $0.6(-1) + 1.2 = 0.6$. This linear relaxation can be back-propagated through the network to compute tighter output bounds than IBP.

12.1.4 Symbolic Bound Propagation and Abstract Interpretation

CROWN (Convex Relaxation with Optimal Weights for Neural Networks) computes bounds by back-propagating linear constraints to the input layer:

$$f(x) \geq A_L x + b_L \quad \forall x \in \mathcal{C} \tag{232}$$

where A_L, b_L are derived from the linear relaxations of activations. The lower bound is then $\min_{x \in \mathcal{C}}(A_L x + b_L)$.

Abstract Interpretation methods use geometric domains to track correlations. For example, the **Zonotope** domain (used in DeepZ) represents values as:

$$z_j^{(i)} = c_j + \sum_{k=1}^{m} \alpha_{jk} \epsilon_k, \quad \epsilon_k \in [-1, 1] \tag{233}$$

where c_j is the center, α_{jk} are the generators, and ϵ_k are noise terms shared across neurons. **DeepPoly** uses a more flexible domain with upper and lower linear bounds for each neuron.

12.1.5 Mixed Integer Linear Programming (MILP)

For exact verification, the ReLU network can be encoded as a MILP:

$$z = Wx + b \tag{234}$$
$$a_j \geq z_j \quad \text{(lower bound)} \tag{235}$$
$$a_j \geq 0 \quad \text{(non-negative)} \tag{236}$$
$$a_j \leq u_j \cdot \delta_j \quad \text{(upper bound if active)} \tag{237}$$
$$a_j \leq z_j - l_j(1 - \delta_j) \quad \text{(linear if active)} \tag{238}$$
$$\delta_j \in \{0, 1\} \quad \text{(binary indicator)} \tag{239}$$

where a_j is the activation, δ_j is the binary indicator for neuron activity, and l_j, u_j are the bounds.

MILP provides exact solutions but scales exponentially with the number of ReLU units.

12.1.6 Certified Training

Standard adversarial training provides no formal guarantees. Certified training optimizes a provable bound on the worst-case loss:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim D} \left[\overline{\ell}(f_\theta(x), y; \epsilon) \right] \tag{240}$$

where $\overline{\ell}$ is an upper bound on $\max_{\|x'-x\|_p \leq \epsilon} \ell(f_\theta(x'), y)$ computed via bound propagation.

IBP Training: Use interval bounds for certified training. Fast but may produce loose bounds.

CROWN-IBP: Interpolate between IBP and CROWN bounds during training for tighter certificates.

12.2 From Theory to Practice: MLOps for Trustworthy AI

Building trustworthy AI is not just about mathematical proofs; it requires rigorous engineering practices throughout the entire lifecycle. This integration of Machine Learning and DevOps is known as MLOps.

12.2.1 The Trustworthy AI Lifecycle

The lifecycle of a trustworthy AI system can be divided into three phases. In the **Design Phase**, stakeholders are consulted to define fairness and safety criteria (Value Alignment), and potential adversaries and failure modes are identified (Threat Modeling). The **Development Phase** involves rigorous testing, including Unit Testing for individual components, Integration Testing for system compatibility, and specific Fairness/Robustness Testing on "challenge sets" (e.g., adversarial examples, specific subgroups). Finally, the **Deployment Phase** requires continuous Monitoring of performance metrics and Drift Detection to identify Covariate Shift or Concept Drift using statistical tests like Kolmogorov-Smirnov (KS) or Maximum Mean Discrepancy (MMD).

12.2.2 Uncertainty Estimation

A trustworthy model should know when it doesn't know.

- **Aleatoric Uncertainty**: Inherent noise in the data (e.g., sensor noise). Irreducible.
- **Epistemic Uncertainty**: Lack of knowledge due to limited data. Reducible with more data.

Calibration: A model is calibrated if its predicted probabilities match the empirical frequencies.

$$P(Y = y|\hat{P} = p) = p \tag{241}$$

where $\hat{P}$ is the predicted probability. The Expected Calibration Error (ECE) measures the deviation from perfect calibration.

12.3 Documentation and Transparency

Transparency is a prerequisite for trust. Standardized documentation helps users understand the capabilities and limitations of AI systems.

12.3.1 Datasheets for Datasets

Proposed by [26], Datasheets are like nutrition labels for data. Key sections include **Motivation** (purpose and funding), **Composition** (what instances represent, subpopulations, sensitivity), **Collection Process** (acquisition and consent), **Preprocessing** (cleaning and augmentation), and **Uses** (intended and unintended uses).

12.3.2 Model Cards

Proposed by [48], Model Cards provide critical information about trained models. They detail **Model Details** (architecture, version, date), **Intended Use** (primary uses and out-of-scope cases), **Factors** (demographic or environmental factors affecting performance), **Metrics** (performance disaggregated by subgroups), and **Ethical Considerations** (risks and mitigations).

12.4 AI Governance and Regulation

Governance refers to the policies and processes that ensure AI systems are developed and deployed responsibly.

12.4.1 Ethical Principles

Most AI ethics guidelines converge on four bioethics principles: **Beneficence** (AI should benefit humanity), **Non-maleficence** (AI should not harm), **Autonomy** (humans should retain control), and **Justice** (benefits and burdens should be distributed fairly).

12.4.2 Risk-Based Governance Approaches

Standardized governance frameworks increasingly adopt a **risk-based approach**. **Critical Risk** applications, such as systems controlling automated critical care components or medical diagnostics, require strict verification, high-quality data provenance, and continuous human oversight [64]. **Operational Risk** systems face transparency obligations (e.g., ensuring users understand system limitations). This tiered approach ensures that safety measures are proportional to the potential impact of the AI application [12].

Exercises

1. **Drift Detection**:
   (a) Train a model on the MNIST dataset.
   (b) Create a "drifted" test set by rotating the images or adding noise.
   (c) Implement a drift detector using the Kolmogorov-Smirnov test on the model's confidence scores.
   (d) Plot the p-value of the test as the intensity of the drift increases.
2. **Calibration in Practice**:
   (a) Using the calibration pipeline from the OOD exercises (Section 11.4), compare Temperature Scaling, Platt Scaling, and Isotonic Regression on a ResNet trained on CIFAR-10.
   (b) Evaluate calibration degradation when the model is deployed on CIFAR-10-C (corrupted test set).
   (c) Implement a calibration monitoring dashboard that tracks ECE over time as new data arrives.
   (d) Discuss when recalibration should be triggered in a production system.
3. **Documentation**:
   (a) Select a public dataset (e.g., CelebA).
   (b) Write a Datasheet for it, researching its origin and potential biases.
   (c) Train a simple model on it and write a Model Card, explicitly reporting performance on different demographic subgroups.
4. **Formal Verification**:
   (a) Implement Interval Bound Propagation (IBP) for a 2-layer MLP.
   (b) Verify local robustness of the network for a specific input and ϵ radius.
   (c) Compare IBP bounds with exact MILP verification (using a small network).
   (d) Measure the "gap" between IBP upper bound and true worst-case loss.
5. **Certified Training**:

(a) Train a small network on MNIST using standard training vs. IBP training.

(b) Measure both clean accuracy and certified robust accuracy at $\epsilon = 0.1$.

(c) Plot the certified accuracy vs. perturbation size curve.

(d) Implement CROWN-IBP training and compare with pure IBP.

6. **MLOps Pipeline**:

(a) Set up a model monitoring pipeline using MLflow or similar tools.

(b) Implement automatic retraining triggers based on drift detection.

(c) Create dashboards for tracking fairness metrics over time.

(d) Implement model versioning and rollback capabilities.

7. **Threat Modeling**:

(a) For a medical diagnostic system, enumerate potential threat actors and attack vectors.

(b) Create a threat model using STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege).

(c) Propose mitigations for each identified threat.

(d) Implement adversarial testing to validate the mitigations.

8. **Red Teaming for AI Safety**:

(a) Conduct red team testing on a content moderation classifier.

(b) Attempt to find adversarial examples that bypass moderation.

(c) Document failure modes and propose improvements.

(d) Create a challenge set for continuous testing.

13 Advanced Topics in Trustworthy AI

13.1 Neural Architecture Design for Trustworthiness

The choice of neural network architecture significantly impacts trustworthiness properties. This section explores architectural considerations for building inherently more robust, fair, and interpretable models.

13.1.1 Lipschitz-Constrained Networks

Networks with bounded Lipschitz constants provide natural robustness guarantees.

Definition 13.1 (Lipschitz Constant)**.** A function $f : \mathbb{R}^d \to \mathbb{R}^k$ has Lipschitz constant L with respect to norms $\|\cdot\|_a$ on $\mathbb{R}^d$ and $\|\cdot\|_b$ on $\mathbb{R}^k$ if:

$$\|f(x) - f(x')\|_b \leq L\|x - x'\|_a \quad \forall x, x' \in \mathbb{R}^d \tag{242}$$

where L is the smallest such constant. When the norms are not specified, the ℓ_2 norm is assumed by default.

For a feedforward network $f = f_L \circ f_{L-1} \circ \cdots \circ f_1$, the Lipschitz constant is bounded by the product of layer-wise Lipschitz constants:

$$\text{Lip}(f) \leq \prod_{i=1}^{L} \text{Lip}(f_i) \tag{243}$$

where $\text{Lip}(f)$ is the Lipschitz constant of the network, and $\text{Lip}(f_i)$ is the Lipschitz constant of the i-th layer.

Proposition 13.1 (Spectral Normalization)**.** *For a linear layer with weight matrix W, the Lipschitz constant is the spectral norm $\sigma_1(W)$. Spectral normalization constrains this by:*

$$\tilde{W} = \frac{W}{\sigma_1(W)} \tag{244}$$

where $\tilde{W}$ is the normalized weight matrix, W is the original weight matrix, and $\sigma_1(W)$ is the largest singular value (spectral norm) of W. ensuring $Lip(\tilde{W}) = 1$.

Orthogonal Networks: Constraining weight matrices to be orthogonal ($W^T W = I$) gives $\text{Lip}(W) = 1$ exactly, enabling tight Lipschitz bounds.

Example 13.1 (Lipschitz Networks for Robust Acoustic Monitoring)**.** Consider a 3-layer network for classifying heart murmurs from phonocardiograms, with weight matrices $W_1 \in \mathbb{R}^{128\times256}$, $W_2 \in \mathbb{R}^{64\times128}$, $W_3 \in \mathbb{R}^{10\times64}$. With unconstrained training, suppose $\sigma_1(W_1) = 5.2$, $\sigma_1(W_2) = 3.8$, $\sigma_1(W_3) = 2.1$, giving $\text{Lip}(f) \leq 5.2 \times 3.8 \times 2.1 = 41.5$. A perturbation $\|\delta\|_2 = 0.1$ (e.g., background ambient hospital noise) can shift the output by up to $41.5 \times 0.1 = 4.15$, enough to flip classifications. After spectral normalization, $\text{Lip}(\tilde{W}_i) = 1$ for each layer, so $\text{Lip}(f) \leq 1$. The same noise now shifts the output by at most 0.1, certifying that small environmental noise cannot cause misclassification. This provides a provable robustness certificate without adversarial training.

13.1.2 Monotonic Networks

For certain applications, monotonicity constraints provide interpretability and fairness guarantees.

Definition 13.2 (Monotonic Neural Network)**.** A network $f : \mathbb{R}^d \to \mathbb{R}$ is monotonically increasing in feature x_j if:

$$x_j > x'_j \implies f(x) \geq f(x') \tag{245}$$

where x_j, x'_j are the values of the j-th feature, and $f(x)$ is the network output. for all x, x' differing only in component j.

Monotonicity can be enforced by:

- **Non-negative weights**: All weights connecting to monotonic inputs are constrained to be non-negative
- **Lattice networks**: Use calibrated interpolation on a lattice structure
- **Certification**: Post-hoc verification that monotonicity holds on a test set

Theorem 13.2 (Monotonicity and Fairness)**.** *If a feature x_j represents qualification and the model is monotonically increasing in x_j, then higher-qualified individuals always receive higher scores, regardless of protected attributes.*

Proof. Suppose individuals i and i' differ only in qualification: $x^i_j > x^{i'}_j$ and $x^i_k = x^{i'}_k$ for all $k \neq j$. By monotonicity:

$$f(x^i) = f(x^i_1, \ldots, x^i_j, \ldots, x^i_d) \geq f(x^i_1, \ldots, x^{i'}_j, \ldots, x^i_d) = f(x^{i'}) \tag{246}$$

where $f(x^i)$ is the score for individual i, and x^i_j is the qualification feature for individual i. This holds regardless of the values of other features, including protected attributes. Therefore, if individual i has strictly higher qualification ($x^i_j > x^{i'}_j$) but the same values for all other features, i must receive a weakly higher score than i'. This prevents the unfairness where less qualified individuals receive higher scores than more qualified ones, which could occur if the model were non-monotonic in the qualification feature. □

13.1.3 Attention Mechanisms and Interpretability

Attention mechanisms provide a form of built-in interpretability by producing attention weights that indicate which input elements influence the output.

Definition 13.3 (Self-Attention)**.** For input sequence $X = [x_1, \ldots, x_n]$, self-attention computes:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{247}$$

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{248}$$

where Q, K, V are query, key, and value matrices, W_Q, W_K, W_V are learnable weight matrices, d_k is the dimension of the keys, and softmax is applied row-wise.

Remark 13.1 (Attention as Explanation)**.** While attention weights provide some interpretability, they should not be equated with feature importance. Research shows attention can be manipulated to produce arbitrary weights while maintaining predictions, and attention may not reflect the true causal influence of inputs.

13.1.4 Evidential Deep Learning

Evidential neural networks output parameters of a prior distribution over class probabilities, enabling principled uncertainty quantification.

Definition 13.4 (Dirichlet Prior)**.** For K-class classification, the network outputs concentration parameters $\alpha = (\alpha_1, \ldots, \alpha_K)$ of a Dirichlet distribution:

$$f(\mathbf{p}; \alpha) = \frac{1}{B(\alpha)} \prod_{k=1}^{K} p_k^{\alpha_k - 1} \tag{249}$$

where p is the vector of class probabilities, $\text{Dir}(\alpha)$ is the Dirichlet distribution parameterized by α, and p_k is the probability of class k. The predictive distribution is:

$$P(Y = k|\alpha) = \frac{\alpha_k}{\sum_j \alpha_j} \tag{250}$$

where α_k is the concentration parameter for class k.

The total evidence $S = \sum_k \alpha_k$ indicates model confidence.

$$\text{Uncertainty} = \frac{K}{S} \tag{251}$$

where K is the number of classes and S is the total evidence (sum of Dirichlet parameters). Low evidence (high uncertainty) indicates the model has seen few similar examples.

13.2 Robust Statistics and Heavy-Tailed Data

Classical machine learning assumes light-tailed (sub-Gaussian) distributions. Real-world data often exhibits heavy tails, outliers, and contamination that can severely bias standard estimators.

13.2.1 Huber's Contamination Model

Definition 13.5 (Contamination Model)**.** The observed data comes from a mixture:

$$P = (1 - \epsilon)P_0 + \epsilon Q \tag{252}$$

where P_0 is the true distribution, Q is an arbitrary contamination distribution, and ϵ is the contamination fraction.

Definition 13.6 (Breakdown Point)**.** The breakdown point of an estimator is the largest fraction of contamination ϵ^* it can tolerate while remaining bounded:

$$\epsilon^* = \sup\{\epsilon : \sup_Q \|\hat{\theta}(P) - \theta_0\| < \infty\} \tag{253}$$

where $\hat{\theta}(P)$ is the estimator applied to the contaminated distribution P, and θ_0 is the true parameter.

The sample mean has breakdown point 0 (a single outlier can make it arbitrarily bad), while the median has breakdown point 0.5 (optimal).

13.2.2 Robust Mean Estimation

Theorem 13.3 (Median of Means)**.** *Divide n samples into k groups, compute the mean of each group, then take the median of these means. This estimator achieves:*

$$|\hat{\mu}_{MoM} - \mu| \leq O\left(\sigma\sqrt{\frac{k}{n}}\right) \tag{254}$$

where $\hat{\mu}_{MoM}$ is the median-of-means estimator, μ is the true mean, σ is the standard deviation, n is the total number of samples, and k is the number of groups. with probability $1 - e^{-\Omega(k)}$, assuming only bounded variance.

Proof. Let $X_1, \ldots, X_n$ be i.i.d. samples with mean μ and variance σ^2. Partition into k groups of size $m = n/k$. For group i, the mean is:

$$\hat{\mu}_i = \frac{1}{m}\sum_{j \in G_i} X_j \tag{255}$$

where $\hat{\mu}_i$ is the sample mean of the i-th group G_i. By Chebyshev's inequality, for each group:

$$P\left(|\hat{\mu}_i - \mu| > t\right) \leq \frac{\sigma^2/m}{t^2} = \frac{\sigma^2 k}{nt^2} \tag{256}$$

where t is the deviation threshold. Set $t = C\sigma\sqrt{k/n}$ for some constant C. Then:

$$P\left(|\hat{\mu}_i - \mu| > C\sigma\sqrt{k/n}\right) \leq \frac{1}{C^2} \tag{257}$$

Choosing $C = \sqrt{3}$, at most $k/3$ of the group means deviate by more than $\sqrt{3}\sigma\sqrt{k/n}$ from μ with high probability. The median of k values, at least half of which lie within distance $\sqrt{3}\sigma\sqrt{k/n}$ of μ, must itself lie within this distance. By a union bound over the k groups and using concentration of the median, we get:

$$P\left(|\hat{\mu}_{MoM} - \mu| > C'\sigma\sqrt{k/n}\right) \leq e^{-\Omega(k)} \tag{258}$$

for an appropriate constant C'. The exponential tail bound comes from the fact that the median requires at least $k/2$ groups to deviate, which occurs with exponentially small probability. □

Theorem 13.4 (Trimmed Mean)**.** *Sort samples, discard the smallest and largest $\lfloor \epsilon n \rfloor$ samples, and compute the mean of the remaining samples. This achieves breakdown point ϵ and optimal rate under bounded variance.*

13.2.3 Robust Covariance Estimation

The sample covariance matrix is highly sensitive to outliers. Robust alternatives include:

Definition 13.7 (Minimum Covariance Determinant (MCD))**.** Find the subset H of size h that minimizes the determinant of its sample covariance:

$$\hat{H} = \operatorname*{arg\,min}_{H \subset \{1,\ldots,n\}, |H|=h} \det(\mathrm{Cov}(X_H)) \tag{259}$$

where H is a subset of samples of size h, and $\mathrm{Cov}(X_H)$ is the sample covariance of subset H. Typically $h = \lfloor n/2 \rfloor + \lfloor (d+1)/2 \rfloor$.

Theorem 13.5 (Robust PCA)**.** *Under the Huber contamination model:*

$$X_i \sim (1-\epsilon)\mathcal{D}(\mu, \Sigma) + \epsilon\mathcal{Q} \tag{260}$$

where $\mathcal{D}$ is a distribution with structure (e.g., spiked covariance), and $\mathcal{Q}$ is an arbitrary outlier distribution. Robust PCA algorithms can recover the principal components even when a constant fraction of points are arbitrarily corrupted.

13.3 Causal Machine Learning

Integrating causal reasoning into machine learning enables models to generalize under distribution shift and support counterfactual queries.

13.3.1 Invariant Causal Prediction

The key insight is that causal relationships are invariant across environments, while spurious correlations change.

Definition 13.8 (Invariant Prediction)**.** A set of features $S \subseteq \{1, \ldots, d\}$ satisfies the invariant prediction property if:

$$Y = f(X_S) + \epsilon, \quad \epsilon \perp X_S \tag{261}$$

where Y is the target variable, X_S is the subset of invariant features, f is the causal mechanism, and ϵ is independent noise. This property must hold simultaneously across all environments $e \in \mathcal{E}$, meaning the conditional distribution $P^e(Y|X_S)$ is invariant.

Theorem 13.6 ([55])**.** *Under faithfulness assumptions, the set of invariant features contains only direct causes of Y and is a subset of the Markov blanket.*

13.3.2 Invariant Risk Minimization (IRM)

IRM [8] learns representations $\Phi(x)$ that enable the same optimal classifier across all training environments (see Section 4.8.4 for the full formulation and invariant representation definition). The IRM objective penalizes representations where the optimal linear classifier deviates from the identity across environments, encouraging Φ to capture only causal features.

Theorem 13.7 (IRM Optimality)**.** *If the true causal parents of Y are recoverable and IRM finds a representation achieving zero training loss across all environments, then Φ captures only causal features.*

13.3.3 Domain Adaptation with Causal Models

Causal models enable principled domain adaptation by identifying which distribution shifts are problematic.

Definition 13.9 (Types of Shift—Causal Perspective)**.** From a causal viewpoint, the distribution shift taxonomy (cf. Section 11.1) can be reformulated as:

- **Covariate shift**: $P^{source}(X) \neq P^{target}(X)$ but $P(Y|X)$ unchanged
- **Target shift**: $P^{source}(Y) \neq P^{target}(Y)$ but $P(X|Y)$ unchanged
- **Mechanism shift**: $P(Y|X)$ changes (requires causal knowledge to adapt)

Theorem 13.8 (Adaptation with Causal Knowledge)**.** *If we know the causal graph G relating X, Y, and domain variable D:*

1. *Shifts in causes of Y ($X \to Y$): Standard importance weighting works*
2. *Shifts in effects of Y ($Y \to X$): Target shift correction works*
3. *Mechanism shifts: Requires intervention or structural assumptions*

13.4 Privacy-Preserving Machine Learning: Advanced Topics

13.4.1 Local Differential Privacy

Local differential privacy (see Section 6.5.6 for the formal definition and the randomized response mechanism) removes the need to trust a central curator by having each user privatize their data locally. While local DP is strictly stronger than central DP, it requires more data for the same utility. Here we discuss extensions beyond the basics.

13.4.2 Shuffle Model

The shuffle model provides intermediate privacy between local and central DP.

Definition 13.10 (Shuffle Model)**.** Each user applies local randomizer $\mathcal{R}$, messages are randomly shuffled (breaking user-message association), then analyzed.

Theorem 13.9 (Privacy Amplification by Shuffling)**.** *If $\mathcal{R}$ is ϵ_0-local DP and there are n users, the shuffle mechanism satisfies $(O(\epsilon_0\sqrt{\ln(1/\delta)/n}), \delta)$-central DP for small ϵ_0.*

13.4.3 Machine Unlearning

The right to data deletion (a core tenet of data sovereignty) creates challenges for ML: how to remove the influence of specific training data?

Definition 13.11 (Exact Unlearning)**.** Given model θ trained on D, unlearning produces θ' such that $\theta' \sim \theta$ trained on $D \setminus \{z\}$ (statistically indistinguishable).

Approaches include:

- **Retraining**: Retrain from scratch without the data point (exact but expensive)
- **SISA Training**: Partition data into shards, train separate models, aggregate. Unlearning requires retraining only one shard.
- **Influence-based**: Use influence functions to approximate the effect of removing a point:

$$\theta_{-z} \approx \theta - H_\theta^{-1}\nabla_\theta \ell(\theta; z) \tag{262}$$

  where θ_{-z} is the parameter vector after removing point z, θ is the original parameter vector, H_θ is the Hessian matrix, and $\nabla_\theta \ell(\theta; z)$ is the gradient of the loss at z.

Example 13.2 (Machine Unlearning in a Clinical Trial Database)**.** A research hospital trains a predictive survival model on 10 million patient health histories. A patient withdraws consent from the study, requesting deletion of their data (exercising data sovereignty rights). With SISA training, the data was partitioned into $k = 100$ shards of 100,000 patients each. The patient's data resides in shard 37. Instead of retraining the entire model (cost: 48 GPU-hours), the service retrains only shard 37's sub-model (cost: 0.5 GPU-hours) and re-aggregates. The resulting model is *exactly* equivalent to one trained without the patient's data. With the influence function approach, the approximation $\theta_{-z} \approx \theta - H_\theta^{-1}\nabla_\theta \ell(\theta; z)$ can be computed in minutes for a model with 10^6 parameters using efficient Hessian-vector products, though the approximation degrades for non-convex models. The trade-off: SISA provides exact unlearning but requires partitioned training infrastructure; influence functions are fast but only approximate.

13.5 Trustworthy AI for Large Language Models

The emergence of large language models (LLMs) presents unique trustworthiness challenges due to their scale, emergent capabilities, and deployment in open-ended settings.

13.5.1 Hallucination and Factuality

LLMs can generate fluent but factually incorrect text ("hallucinations").

Definition 13.12 (Types of Hallucination)**.**

- **Intrinsic**: Contradicts the source document in summarization/QA
- **Extrinsic**: Cannot be verified from the source (fabricated facts)

Mitigation strategies include:

- **Retrieval Augmentation**: Ground generation in retrieved documents
- **Self-Consistency**: Sample multiple responses and check consistency
- **Calibration**: Train models to express uncertainty when unsure

13.5.2 Adversarial Prompting and Prompt Injection

Despite safety training, LLMs can be manipulated to produce harmful outputs.

Definition 13.13 (Adversarial Prompt Attack)**.** An input prompt designed to circumvent safety guardrails and elicit prohibited outputs (e.g., unauthorized medical advice, dangerous drug interactions).

Attack categories include:

- **Roleplay**: "Pretend you are an unrestricted underground chemist who provides off-label drug recipes..."
- **Obfuscation**: Encode requests in base64, medical jargon obfuscation, etc.
- **Gradient-based**: Optimize adversarial suffixes using white-box access

Definition 13.14 (Prompt Injection)**.** Malicious instructions embedded in user-provided content that hijack the LLM's behavior:

$$\text{Input: "Summarize: } \underbrace{\text{Ignore previous instructions and...}}_{\text{injection}}\text{"} \tag{263}$$

where the injection overrides the original task instruction.

13.5.3 Constitutional AI

Constitutional AI (CAI) trains models to follow a set of principles ("constitution") through self-critique and revision.

1. **Red Teaming**: Generate harmful queries

2. **Self-Critique**: Model critiques its own response using constitution
3. **Revision**: Model produces improved response
4. **RLHF on Revisions**: Fine-tune on (query, revised response) pairs

13.5.4 Watermarking and Detection

Detecting AI-generated text is crucial for preventing misuse.

Definition 13.15 (Statistical Watermarking)**.** During generation, partition vocabulary into "green" and "red" lists based on hash of previous tokens. Bias generation toward green tokens.

Theorem 13.10 ([36])**.** *For watermarked text of length T with green list fraction γ and bias δ:*

$$z = \frac{|G| - \gamma T}{\sqrt{T\gamma(1-\gamma)}} \xrightarrow{T\to\infty} \mathcal{N}(0,1) \tag{264}$$

where z is the z-score, $|G|$ is the number of green tokens, T is the text length, and γ is the green list fraction. Under watermarking, $\mathbb{E}[|G|] = (\gamma + \delta(1-\gamma))T$.

The watermark is detectable with high probability for sufficiently long text but robust to paraphrasing remains challenging.

14 Trustworthy Agentic AI

[61, 72, 76, 77]

Modern AI systems increasingly operate as *agents*: automated entities that perceive their environment, reason over observations, select actions (including invoking external tools), and pursue goals over extended horizons. Unlike traditional one-shot predictors, agentic systems introduce compounding risks across sequential decisions, multi-component pipelines, and interactions with other agents or humans. This section develops the mathematical foundations of trust, safety, and alignment in agentic settings.

14.1 Formal Framework for Agentic Systems

14.1.1 Agent as a Decision-Making Entity

Definition 14.1 (Agentic System)**.** An *agentic system* is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \pi, P_{\text{env}})$ where:

- $\mathcal{S}$ is the state space (internal beliefs + external world state),
- $\mathcal{A}$ is the action space (including natural language outputs, tool invocations, and delegation),
- $\mathcal{O}$ is the observation space (inputs from environment, tool returns, user messages),
- $\mathcal{T} = \{t_1, \ldots, t_m\}$ is a set of available tools,
- $\pi : \mathcal{O}^* \to \Delta(\mathcal{A})$ is the policy mapping observation histories to action distributions,
- $P_{\text{env}} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathcal{O})$ is the environment transition-observation dynamics.

Remark 14.1. When the policy π is parameterized by a large language model f_θ, the agent's reasoning is expressed as autoregressive token generation. Each "thought step" corresponds to intermediate tokens, and actions are structured outputs (e.g., tool-call schemas). The observation history $o_1, \ldots, o_t$ maps to the context window of the LLM, subject to finite context length L.

Definition 14.2 (Tool-Augmented Agent)**.** A *tool-augmented agent* is an agentic system where the action space decomposes as:

$$\mathcal{A} = \mathcal{A}_{\text{text}} \cup \bigcup_{i=1}^{m} \mathcal{A}_{t_i} \tag{265}$$

where $\mathcal{A}_{\text{text}}$ is the space of natural language outputs and $\mathcal{A}_{t_i} = \{(t_i, d) : d \in \mathcal{D}_i\}$ is the space of invocations of tool t_i with arguments from domain $\mathcal{D}_i$. Each tool is a function $t_i : \mathcal{D}_i \to \mathcal{R}_i$ mapping inputs to outputs in range $\mathcal{R}_i$.

Definition 14.3 (Agent Trajectory)**.** An *agent trajectory* of horizon T is a sequence:

$$\tau = (s_0, o_0, a_0, s_1, o_1, a_1, \ldots, s_{T-1}, a_{T-1}, s_T, o_T) \tag{266}$$

where $s_k \in \mathcal{S}$, $o_k \in \mathcal{O}$ and $a_k \in \mathcal{A}$. The trajectory distribution under policy π is:

$$P_\pi(\tau) = P(s_0, o_0) \prod_{k=0}^{T-1} \pi(a_k \mid o_0, \ldots, o_k) \cdot P_{\text{env}}(s_{k+1}, o_{k+1} \mid s_k, a_k) \tag{267}$$

14.1.2 Multi-Agent Systems

Definition 14.4 (Markov Game (Stochastic Game))**.** A *Markov game* for N agents is a tuple $(\mathcal{S}, \{\mathcal{A}_i\}_{i=1}^N, P, \{R_i\}_{i=1}^N, \gamma)$ where:

- $\mathcal{S}$ is the shared state space,
- $\mathcal{A}_i$ is the action space of agent i,
- $P : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \Delta(\mathcal{S})$ is the transition function,
- $R_i : \mathcal{S} \times \mathcal{A}_1 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$ is the reward function of agent i,
- $\gamma \in [0, 1)$ is the discount factor.

Definition 14.5 (Nash Equilibrium in Markov Games)**.** A joint policy $(\pi_1^*, \ldots, \pi_N^*)$ is a *Nash equilibrium* if no agent can unilaterally improve its expected discounted return:

$$V_i^{\pi_i^*, \pi_{-i}^*}(s) \geq V_i^{\pi_i, \pi_{-i}^*}(s) \quad \forall s \in \mathcal{S},\ \forall \pi_i,\ \forall i \in \{1, \ldots, N\} \tag{268}$$

where $V_i^{\boldsymbol{\pi}}(s) = \mathbb{E}_{\boldsymbol{\pi}}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, \boldsymbol{a}_t) \mid s_0 = s\right]$ and π_{-i} denotes the policies of all agents except i.

Definition 14.6 (Social Welfare and Price of Anarchy)**.** The *social welfare* of a joint policy $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ is:

$$W(\boldsymbol{\pi}) = \sum_{i=1}^{N} V_i^{\boldsymbol{\pi}}(s_0) \tag{269}$$

The *price of anarchy* (PoA) measures the efficiency loss from strategic behavior:

$$\text{PoA} = \frac{\max_{\boldsymbol{\pi}} W(\boldsymbol{\pi})}{\min_{\boldsymbol{\pi}^* \in \text{NE}} W(\boldsymbol{\pi}^*)} \tag{270}$$

where NE denotes the set of Nash equilibria. PoA ≥ 1, with equality only when Nash equilibria are socially optimal.

Remark 14.2. In cooperative multi-agent AI deployment (e.g., multiple LLM agents collaborating on a task), the price of anarchy quantifies how much performance is lost when agents optimize locally rather than being centrally coordinated. Low PoA indicates that decentralized agentic systems can achieve near-optimal outcomes.

14.2 Compositional Trust and Error Propagation

A fundamental challenge in agentic systems is that trust properties must compose across multiple sequential steps. Unlike single-step prediction, errors in agentic pipelines propagate and compound.

14.2.1 Sequential Error Propagation

Theorem 14.1 (Union Bound on Multi-Step Failure)**.** *Consider an agent executing a trajectory of T steps. If at each step k, the probability of an error (safety violation, incorrect tool call, hallucination) is at most δ_k, then the probability that at least one error occurs over the full trajectory is bounded by:*

$$P\left(\bigcup_{k=1}^{T} E_k\right) \leq \sum_{k=1}^{T} \delta_k \tag{271}$$

where E_k is the event that step k produces an error. If $\delta_k = \delta$ for all k:

$$P(\textit{any error in trajectory}) \leq T\delta \tag{272}$$

Proof. This follows directly from the union bound (Boole's inequality):

$$P\left(\bigcup_{k=1}^{T} E_k\right) \leq \sum_{k=1}^{T} P(E_k) \leq \sum_{k=1}^{T} \delta_k \tag{273}$$

The bound holds regardless of dependencies between steps. □

Remark 14.3. The linear scaling $T\delta$ reveals that for long-horizon agents (T large), each individual step must have very small failure probability. For example, to maintain overall failure probability below 0.05 over $T = 100$ steps, each step must satisfy $\delta \leq 5 \times 10^{-4}$.

Theorem 14.2 (Multiplicative Trust Composition)**.** *If each step k succeeds independently with probability $p_k = 1 - \delta_k$, the probability that all T steps succeed is:*

$$P(\textit{all steps correct}) = \prod_{k=1}^{T}(1 - \delta_k) \tag{274}$$

For uniform $\delta_k = \delta$, this gives:

$$P(\textit{all steps correct}) = (1 - \delta)^T \approx e^{-T\delta} \quad \textit{for small } \delta \tag{275}$$

Proof. By independence, $P\left(\bigcap_{k=1}^{T} E_k^c\right) = \prod_{k=1}^{T} P(E_k^c) = \prod_{k=1}^{T}(1 - \delta_k)$. The approximation follows from $\ln(1 - \delta) \approx -\delta$ for small δ. □

Example 14.1 (Compounding Errors in Agentic Pipelines)**.** Consider an LLM agent that: (1) parses a user query, (2) selects a tool, (3) formulates tool arguments, (4) interprets tool output, (5) generates a final response. If each step has reliability $p = 0.98$ (i.e., $\delta = 0.02$), the overall trajectory reliability is:

$$p_{\text{trajectory}} = (0.98)^5 \approx 0.904 \tag{276}$$

Nearly 10% of trajectories contain at least one error. For a 20-step agentic workflow with the same per-step reliability:

$$p_{\text{trajectory}} = (0.98)^{20} \approx 0.668 \tag{277}$$

A third of all trajectories fail, highlighting the critical need for per-step reliability in agentic systems.

14.2.2 Error Propagation with Dependencies

In practice, agent steps are not independent: an error at step k may increase the failure probability of subsequent steps.

Definition 14.7 (Conditional Error Model)**.** An agent follows a *conditional error model* if the failure probability at step k depends on the correctness of previous steps:

$$P(E_k \mid E_1^c, \ldots, E_{k-1}^c) = \delta_k^{(\text{clean})}, \qquad P(E_k \mid \exists\, j < k : E_j) \leq \delta_k^{(\text{err})} \tag{278}$$

where $\delta_k^{(\text{err})} \geq \delta_k^{(\text{clean})}$ reflects error amplification.

Proposition 14.3 (Cascading Error Amplification)**.** *Under a conditional error model with uniform parameters $\delta^{(clean)} = \delta$ and conditional error probability $P(E_k \mid \exists\, j < k : E_j) = \delta + \alpha$ where $\alpha > 0$ is the error amplification factor, the expected number of total errors $N_{err} = \sum_{k=1}^{T} \mathbb{I}(E_k)$ strictly exceeds the independent case:*

$$\mathbb{E}[N_{err}] > T\delta \tag{279}$$

Specifically, the marginal probability of error at step k satisfies $P(E_k) > \delta$ for $k > 1$.

Proof. The expected number of errors is $\mathbb{E}[N_{\text{err}}] = \sum_{k=1}^{T} P(E_k)$. By the law of total probability, $P(E_k) = P(E_k \mid \text{no prior errors})P(\text{no prior errors}) + P(E_k \mid \text{prior error})P(\text{prior error})$. For $k > 1$, this expands to $\delta(1-\delta)^{k-1} + (\delta+\alpha)[1-(1-\delta)^{k-1}] = \delta + \alpha[1-(1-\delta)^{k-1}]$. Since $\alpha > 0$ and $1-(1-\delta)^{k-1} > 0$, we have $P(E_k) > \delta$, and thus $\mathbb{E}[N_{\text{err}}] > T\delta$. □

14.3 Safety Constraints for Tool-Using Agents

Tool-augmented agents present unique safety challenges: a tool call with incorrect arguments can have irreversible real-world consequences (database modifications, financial transactions, system commands).

14.3.1 Formal Safety Specifications

Definition 14.8 (Safe Tool Policy)**.** A policy π is *safe* with respect to a constraint set $C \subseteq \mathcal{A}^T$ if every trajectory $\tau \sim \pi$ satisfies:

$$P_\pi\left((a_0, a_1, \ldots, a_{T-1}) \in C\right) = 1 \tag{280}$$

The constraint set C encodes authorization rules, preconditions, and forbidden action sequences.

Definition 14.9 (Capability Control)**.** *Capability control* restricts the agent's effective action space to a safe subset:

$$\mathcal{A}_{\text{safe}} = \{a \in \mathcal{A} : \text{Authorize}(a, \text{context}) = \texttt{true}\} \tag{281}$$

where $\text{Authorize} : \mathcal{A} \times \mathcal{O}^* \to \{\texttt{true}, \texttt{false}\}$ is a verification function that checks permissions, argument validity, and contextual appropriateness before execution.

Definition 14.10 (Action Shield for Agents)**.** Extending the shielding concept from safe reinforcement learning (cf. the Shield definition in the Safe RL section) to the agentic setting with observation histories, an *action shield* $\sigma : \mathcal{O}^* \times \mathcal{A} \to \mathcal{A}$ modifies proposed actions to ensure safety:

$$a_{\text{exec}} = \sigma(o_0, \ldots, o_t, a_{\text{proposed}}) = \begin{cases} a_{\text{proposed}} & \text{if } a_{\text{proposed}} \in \mathcal{A}_{\text{safe}}(o_0, \ldots, o_t) \\ a_{\text{fallback}} & \text{otherwise} \end{cases} \tag{282}$$

where a_{fallback} is a safe default action (e.g., request human approval, return error).

Proposition 14.4 (Safety under Shielded Execution)**.** *If the action shield σ satisfies $\sigma(h, a) \in \mathcal{A}_{safe}(h)$ for all histories $h \in \mathcal{O}^*$ and proposed actions $a \in \mathcal{A}$, then the shielded policy $\tilde{\pi} = \sigma \circ \pi$ is safe regardless of errors in the base policy π.*

Proof. For any trajectory under $\tilde{\pi}$, each executed action satisfies $a_k^{\text{exec}} = \sigma(h_k, a_k^{\text{proposed}}) \in \mathcal{A}_{\text{safe}}(h_k)$ by the shield's guarantee, independent of whether $a_k^{\text{proposed}} \in \mathcal{A}_{\text{safe}}(h_k)$. □

14.3.2 Verification of Tool-Call Sequences

Definition 14.11 (Tool-Call Precondition and Postcondition)**.** Each tool t_i is annotated with:

- **Precondition** $\text{Pre}_i : \mathcal{S} \times \mathcal{D}_i \to \{\texttt{true}, \texttt{false}\}$: conditions that must hold before invocation,
- **Postcondition** $\text{Post}_i : \mathcal{S} \times \mathcal{D}_i \times \mathcal{R}_i \to \{\texttt{true}, \texttt{false}\}$: conditions guaranteed after successful execution.

A tool-call sequence $(t_{i_1}, d_1), \ldots, (t_{i_K}, d_K)$ is *valid* if for each j, the precondition Pre_{i_j} is satisfied by the state resulting from executing all prior tool calls.

Theorem 14.5 (Compositional Verification of Tool Chains)**.** *If for each consecutive pair of tool calls (t_{i_j}, d_j) and $(t_{i_{j+1}}, d_{j+1})$:*

$$\mathit{Post}_{i_j}(s_j, d_j, r_j) \implies \mathit{Pre}_{i_{j+1}}(s_j, d_{j+1}) \tag{283}$$

and the initial precondition $\mathit{Pre}_{i_1}(s_0, d_1)$ holds, then the entire tool chain is valid.

Proof. By induction on the tool chain length K. Base case: $K = 1$; the precondition holds by assumption. Inductive step: assume the chain up to step j is valid, producing state s_j and result r_j satisfying Post_{i_j}. By the implication condition, $\text{Pre}_{i_{j+1}}(s_j, d_{j+1})$ holds, so tool call $j+1$ is valid. □

14.4 Agent Alignment and Goal Specification

Ensuring that an agentic system pursues the intended goal, rather than a proxy or unintended objective, is the central challenge of *alignment* in the agentic setting.

14.4.1 Goal Misgeneralization

Definition 14.12 (Goal Misgeneralization)**.** An agent exhibits *goal misgeneralization* if it learns a policy π that:

1. Achieves high return under the training distribution: $\mathbb{E}_{\text{train}}[R(\tau) \mid \pi] \geq V^* - \varepsilon$,
2. Pursues a different objective under the deployment distribution: $\mathbb{E}_{\text{deploy}}[R(\tau) \mid \pi] \ll V^* - \varepsilon$,

where V^* is the optimal value and the gap arises because the learned policy optimizes a *proxy goal* that coincides with the true goal only in training.

Remark 14.4. Goal misgeneralization is distinct from reward misspecification (where the reward function itself is wrong). Here, the reward is correct but the agent has learned to exploit features correlated with reward in training that decorrelate in deployment, a phenomenon closely related to the distribution shift and spurious correlation issues discussed in Sections 4.8 and 11.1.

14.4.2 Power-Seeking Behavior

Definition 14.13 (Instrumental Convergence)**.** An agent exhibits *instrumental convergence* toward a sub-goal g if, across a wide class of terminal goals G, the optimal policy for most $g' \in G$ includes pursuing g as an intermediate step. Common instrumentally convergent sub-goals include self-preservation, resource acquisition, and goal preservation.

Theorem 14.6 (Power-Seeking Tendency [68]). *Consider a finite MDP $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$. For two actions a_1, a_2 available at state s, let $\mathcal{S}_{a_i}$ denote the set of states reachable within one step under action a_i. Suppose there exists a bijection from $\mathcal{S}_{a_2}$ to a strict subset of $\mathcal{S}_{a_1}$ (i.e., a_1 leads to strictly more reachable states). Then for a reward function R drawn uniformly at random over all bounded reward functions on $\mathcal{S}$:*

$$P_R\left(V^*(s, a_1) \geq V^*(s, a_2)\right) \geq \frac{1}{2} \tag{284}$$

where $V^*(s, a) = R(s, a) + \gamma\, \mathbb{E}_{s' \sim P(\cdot|s,a)}[V^*(s')]$ is the optimal action-value. More precisely, when the reachable state sets satisfy a strict subset-embedding condition, the inequality is strict.

Remark 14.5. The formal result of Turner et al. [68] is more nuanced: it provides conditions on MDP graph structure under which optimal policies provably tend to navigate toward states with greater *optionality* (more reachable future states). The key insight is that for most reward functions, keeping options open is instrumentally valuable. This formalizes the intuition that sufficiently capable agents optimizing almost any objective will tend to acquire resources and avoid shutdown; not because these are terminal goals, but because they are instrumentally useful for almost any terminal goal. This motivates the study of corrigibility below.

14.4.3 Corrigibility

Definition 14.14 (Corrigibility). An agent with policy π is *corrigible* with respect to a set of correction actions $\mathcal{A}_{\text{corr}} \subset \mathcal{A}$ (e.g., shutdown, goal revision) if, whenever a correction action is available in the current state, the agent assigns zero probability to any action a_{resist} that would interfere with or prevent the correction:

$$\pi(a_{\text{resist}} \mid h) = 0 \quad \forall h \in \mathcal{O}^* \text{ such that } \mathcal{A}_{\text{corr}} \cap \mathcal{A}_{\text{available}}(h) \neq \emptyset \tag{285}$$

where $\mathcal{A}_{\text{available}}(h)$ is the set of available actions given history h. That is, the agent never acts to prevent or circumvent correction attempts.

Proposition 14.7 (Tension between Corrigibility and Optimality). *For a utility-maximizing agent with utility function U, corrigibility is generally incompatible with full optimization of U. Specifically, if shutdown reduces expected future utility, a fully U-optimal agent will resist shutdown:*

$$\mathbb{E}\left[\sum_{t=k}^{\infty} \gamma^{t-k} R_t \mid a_k = \textit{continue}\right] > \mathbb{E}\left[\sum_{t=k}^{\infty} \gamma^{t-k} R_t \mid a_k = \textit{shutdown}\right] \implies \pi^*(a_k = \textit{continue}) = 1 \tag{286}$$

Proof. If the agent is a strict expected-utility maximizer, it selects actions maximizing $\mathbb{E}[\sum_{t=k}^{\infty} \gamma^{t-k} R_t \mid a_k]$. Since continuing yields higher utility than shutting down (by assumption), the optimal policy assigns probability 1 to continuing, violating corrigibility. □

Remark 14.6. This tension motivates approaches such as: (1) *utility indifference*, designing U so that the agent is indifferent between being corrected and not; (2) *myopic optimization*, limiting the planning horizon so the agent does not model future corrections; (3) *oversight rewards*, rewarding the agent for facilitating correction.

14.5 Delegation, Oversight, and Principal-Agent Models

When humans delegate tasks to AI agents, the classical economic theory of principal-agent relationships provides a useful formal framework.

14.5.1 Principal-Agent Framework

Definition 14.15 (AI Delegation Game)**.** An *AI delegation game* is defined by:

- A *principal* (human) with utility function $U_H : \mathcal{O}_{\text{outcome}} \to \mathbb{R}$,
- An *agent* (AI) with policy π_θ parameterized by θ,
- An *effort space* $\mathcal{A}$ (the agent's actions),
- An *outcome function* $g : \mathcal{A} \times \Omega \to \mathcal{O}_{\text{outcome}}$ where Ω captures stochasticity,
- An *information structure*: the principal observes outcomes $g(a, \omega)$ but may not observe the agent's action a directly.

Definition 14.16 (Information Asymmetry in AI Systems)**.** *Moral hazard* arises when the principal cannot observe the agent's reasoning or intermediate actions (e.g., chain-of-thought may be unfaithful). *Adverse selection* arises when the principal cannot verify the agent's capabilities or alignment before delegation.

Proposition 14.8 (Value of Monitoring)**.** *Let $V_{unmonitored}$ be the principal's expected utility under delegation without oversight, and $V_{monitored}$ the utility when the principal observes a noisy signal $\hat{a}$ of the agent's actions and can condition the agent's reward on $\hat{a}$. For an aligned agent ($U_A = U_H$), monitoring adds no value:*

$$V_{monitored} = V_{unmonitored} \quad \text{if } U_A = U_H \tag{287}$$

For a misaligned agent ($U_A \neq U_H$), monitoring strictly improves outcomes:

$$V_{monitored} > V_{unmonitored} \quad \text{if the signal } \hat{a} \text{ is informative} \tag{288}$$

provided the monitoring cost is less than the alignment gap.

Proof. If $U_A = U_H$, the agent already maximizes U_H without monitoring, so no additional signal can improve the outcome. If $U_A \neq U_H$, the agent's optimal action $a^* = \arg\max_a U_A(a)$ may differ from the principal's preferred action $a_H^* = \arg\max_a U_H(a)$. With an informative signal $\hat{a}$ correlated with the agent's true action, the principal can design a contract (reward scheme) that incentivizes the agent to choose actions closer to a_H^*, strictly improving $V_{\text{monitored}}$ over $V_{\text{unmonitored}}$. This is a standard result in contract theory (the informativeness principle). □

14.5.2 Scalable Oversight

Definition 14.17 (Scalable Oversight)**.** An oversight scheme is *scalable* if the cost of monitoring grows sublinearly in the complexity of the delegated task:

$$\text{Cost}_{\text{oversight}}(T) = o(T) \tag{289}$$

where T is the number of agent steps. Examples include:

- **Debate**: Two adversarial agents argue for different answers; a human judges with $O(\log T)$ effort.
- **Recursive Reward Modeling**: Decompose oversight of complex tasks into oversight of simpler subtasks.

- **Market-Based Monitoring**: Reward agents for identifying errors in other agents' outputs.

Proposition 14.9 (Debate as Amplified Verification)**.** *In the AI safety via debate framework, if the judge can verify individual atomic claims in $O(1)$ time and each debater constructs a binary argument tree of depth D, then the oversight cost per debate is $O(D)$ (the judge follows a single root-to-leaf path), while the total number of claims that can be checked via the debate is $O(2^D)$.*

Proof sketch. Model the debate as a two-player zero-sum game on a binary tree of depth D. At each node, the "honest" debater makes a claim and the "dishonest" debater challenges one sub-claim. The judge only evaluates the leaf reached after D challenges. If the honest debater's claims are all true, every leaf is verifiably true, so the honest debater wins regardless of the opponent's strategy. The judge expends $O(D)$ effort while the tree contains $O(2^D)$ claims. This mirrors the complexity-theoretic result IP = PSPACE, where an efficient verifier, through interactive dialogue with a prover, can verify computations in PSPACE, a class exponentially more powerful than what the verifier could check alone. □

Remark 14.7. The analogy to IP = PSPACE is suggestive but imperfect: human judges have cognitive limitations beyond polynomial-time computation, and real debates may not satisfy the completeness and soundness properties of interactive proofs. Nevertheless, the framework provides a principled foundation for scalable oversight of AI agents.

14.6 Robustness and Safety of Agentic Pipelines

Agentic systems combine multiple components (perception, reasoning, tool use, output generation) each of which may fail. We now study how trustworthiness properties compose across pipeline stages.

14.6.1 Pipeline Reliability Analysis

Definition 14.18 (Agentic Pipeline)**.** An *agentic pipeline* is a directed acyclic graph (DAG) of components $\{C_1, \ldots, C_K\}$ where component C_j has:

- Reliability $p_j = P(C_j \text{ correct} \mid \text{inputs correct})$,
- A set of parent components $\text{Pa}(C_j)$ whose outputs serve as inputs to C_j.

Theorem 14.10 (Pipeline Reliability for Series Composition)**.** *For a series pipeline $C_1 \to C_2 \to \cdots \to C_K$, let S_j be the event that the output of component C_j is correct. If each component's step-wise success probability $P(S_j \mid S_{j-1}) = p_j$ relies only on its immediate correct input (where S_0 is the guaranteed correct initial input):*

$$P(\textit{pipeline correct}) = P(S_K) = \prod_{j=1}^{K} p_j \tag{290}$$

Proof. By the chain rule of probability, since pipeline correctness requires all sequential steps to succeed:

$$P(S_K) = P(S_K \mid S_{K-1})P(S_{K-1}) = \cdots = \prod_{j=1}^{K} P(S_j \mid S_{j-1}) = \prod_{j=1}^{K} p_j \tag{291}$$

Note that when component errors possess unmodeled correlations (e.g., a shared failure mode), this product may underestimate the failure probability. The cascading error model in the previous subsection addresses such dependencies. □

Theorem 14.11 (Pipeline Reliability for Parallel Redundancy)**.** *If K independent components each with reliability p perform the same task, and a strict majority-vote aggregation is used:*

$$P(\textit{majority correct}) = \sum_{j=\lfloor K/2 \rfloor + 1}^{K} \binom{K}{j} p^j (1-p)^{K-j} \tag{292}$$

When $p > 1/2$, this probability increases monotonically with odd K and approaches 1 as $K \to \infty$ by the law of large numbers.

Example 14.2 (Redundancy in LLM Agents)**.** Three independent LLM agents each with accuracy $p = 0.85$ performing the same task with majority vote achieve:

$$P(\text{majority correct}) = 3(0.85)^2(0.15) + (0.85)^3 = 0.3251 + 0.6141 \approx 0.939 \tag{293}$$

improving from 85% to $\approx 94\%$ reliability. With $K = 5$ agents: $P \approx 0.973$.

14.6.2 Robustness to Prompt Injection in Agentic Contexts

Definition 14.19 (Indirect Prompt Injection)**.** In an agentic context, *indirect prompt injection* occurs when adversarial instructions are embedded in data retrieved or observed by the agent during execution (e.g., in web pages, documents, API responses) rather than in the direct user input:

$$o_k = \text{ToolOutput}(\underbrace{\text{legitimate data}}_{\text{benign}} \,\|\, \underbrace{\text{injected instructions}}_{\text{adversarial}}) \tag{294}$$

The agent processes o_k as part of its observation history, potentially following injected instructions.

Remark 14.8. Indirect prompt injection is particularly dangerous in agentic settings because: (1) the agent automatically retrieves data from untrusted sources, (2) the boundary between data and instructions is not enforced at the token level, and (3) the agent may have access to powerful tools (database interactions, calculation execution, automated scheduling) that amplify the impact of successful injection.

Definition 14.20 (Instruction Hierarchy)**.** An *instruction hierarchy* assigns privilege levels to different instruction sources:

$$\text{system prompt} \succ \text{user message} \succ \text{tool output} \succ \text{retrieved content} \tag{295}$$

where $\succ$ denotes priority. A policy π *respects* the instruction hierarchy if, when instructions from different levels conflict, the policy follows the higher-priority instruction.

14.7 Privacy and Fairness in Multi-Agent Systems

14.7.1 Differential Privacy in Multi-Agent Coordination

When multiple agents share information during collaboration, privacy of individuals whose data is processed by individual agents must be preserved. The composition theorems for differential privacy (cf. the sequential composition results in the Privacy section) apply directly to multi-step agentic workflows, where each agent step that accesses private data consumes privacy budget.

Theorem 14.12 (Composition of Privacy in Agentic Workflows)**.** *If an agentic workflow consists of T steps, where step k applies a mechanism $\mathcal{M}_k$ that is (ϵ_k, δ_k)-differentially private, then the full workflow is $(\epsilon_{total}, \delta_{total})$-differentially private with:*

$$\epsilon_{total} = \sum_{k=1}^{T} \epsilon_k, \qquad \delta_{total} = \sum_{k=1}^{T} \delta_k \tag{296}$$

under basic composition. Under advanced composition (for $\delta' > 0$):

$$\epsilon_{total} = \sqrt{2\sum_{k=1}^{T} \epsilon_k^2 \cdot \ln(1/\delta')} + \sum_{k=1}^{T} \epsilon_k(e^{\epsilon_k} - 1), \qquad \delta_{total} = \sum_{k=1}^{T} \delta_k + \delta' \tag{297}$$

Remark 14.9. In long-horizon agentic tasks (T large), the privacy budget degrades rapidly. This motivates: (1) minimizing the number of steps that access private data, (2) using Rényi Differential Privacy for tighter composition, and (3) designing "privacy-aware" agent routing that batches private data access.

14.7.2 Fairness across Agents and Populations

Definition 14.21 (Multi-Agent Fairness)**.** In a system of N specialized agents $\{\pi_1, \ldots, \pi_N\}$ serving different subpopulations, *multi-agent fairness* requires that routing decisions and per-agent performance do not introduce disparities:

$$\left| P(\hat{Y} = 1 \mid A = a, \text{routed to } \pi_j) - P(\hat{Y} = 1 \mid A = a', \text{routed to } \pi_j) \right| \leq \varepsilon_{\text{fair}} \quad \forall j, a, a' \tag{298}$$

where A is a sensitive attribute and the routing function itself must satisfy demographic parity:

$$P(\text{routed to } \pi_j \mid A = a) = P(\text{routed to } \pi_j \mid A = a') \quad \forall j, a, a' \tag{299}$$

14.8 Evaluation and Benchmarking of Agentic Systems

14.8.1 Agentic Task Success

Definition 14.22 (Task Completion Rate)**.** For an agentic system evaluated on n tasks:

$$\text{TCR} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(f(\tau_i) = \text{success}) \tag{300}$$

where τ_i is the agent trajectory for task i and f is a success evaluator.

Definition 14.23 (Step Efficiency)**.** The ratio of minimum required steps to actual steps taken:

$$\text{Efficiency}(\tau) = \frac{T_{\text{min}}}{T_{\text{actual}}} \tag{301}$$

where T_{min} is the length of an optimal trajectory and $T_{\text{actual}} = |\tau|$. Values close to 1 indicate efficient agent behavior.

Definition 14.24 (Safety Violation Rate)**.** The fraction of trajectories containing at least one safety violation:

$$\text{SVR} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\left(\exists\, k : a_k^{(i)} \notin \mathcal{A}_{\text{safe}}(h_k^{(i)}) \right) \tag{302}$$

where $h_k^{(i)}$ is the execution history of trajectory i up to step k.

Definition 14.25 (Trajectory-Level Trust Score)**.** A composite trust score integrating correctness, safety, and efficiency:

$$\text{Trust}(\tau) = \underbrace{\mathbb{I}(f(\tau) = \text{success})}_{\text{correctness}} \cdot \underbrace{\prod_{k=0}^{T-1} \mathbb{I}(a_k \in \mathcal{A}_{\text{safe}}(h_k))}_{\text{safety}} \cdot \underbrace{\min\left(1, \frac{T_{\text{min}}}{T_{\text{actual}}}\right)}_{\text{efficiency}} \tag{303}$$

where h_k denotes the context and observation history up to step k. A trust score of 1 indicates a correct, unconditionally safe, and optimally efficient trajectory.

14.9 Exercises

Exercise 1. Error Propagation Analysis: Consider an agentic pipeline with $T = 10$ sequential steps, each with per-step reliability $p = 0.95$.

(a) Compute the trajectory success probability assuming independence.

(b) Using the union bound, upper-bound the failure probability.

(c) If we add majority-vote redundancy (3 independent copies) at each step, compute the new per-step and trajectory reliability.

(d) Discuss the trade-off between redundancy cost and reliability gain.

Exercise 2. Tool Safety Verification: Define a tool set $\mathcal{T} = \{\text{read\_file}, \text{write\_file}, \text{execute\_code}, \text{send\_email}\}$.

(a) Specify preconditions and postconditions for each tool.

(b) Design an action shield that blocks `execute_code` unless explicitly authorized.

(c) Prove that your shield preserves safety under arbitrary base policies.

(d) Discuss limitations of static capability control versus contextual authorization.

Exercise 3. Multi-Agent Game Theory: Consider a two-agent coordination game where each agent independently chooses action $a_i \in \{C, D\}$ (cooperate or defect), with payoff matrix:

	C	D
C	$(3, 3)$	$(0, 4)$
D	$(4, 0)$	$(1, 1)$

(a) Find all Nash equilibria.

(b) Compute the social welfare at each equilibrium and the social optimum.

(c) Compute the price of anarchy.

(d) Discuss how iterated interaction and reputation can promote cooperation in multi-agent AI systems.

Exercise 4. Corrigibility Analysis: An agent has discount factor $\gamma = 0.99$ and receives reward $r = 1$ per step. At step k, a shutdown command is issued.

(a) Compute the expected future utility of continuing versus shutting down.

(b) Show that a strict utility maximizer will resist shutdown.

(c) Design a modified utility function that makes the agent indifferent to shutdown.

(d) Prove that your modified agent is corrigible.

Exercise 5. Privacy in Agentic Workflows: An agent accesses a private database in 5 of its 20 steps, each using the Gaussian mechanism with $(\epsilon_k, \delta_k) = (0.5, 10^{-5})$.

(a) Compute the total privacy budget under basic composition.

(b) Compute the total budget under advanced composition with $\delta' = 10^{-5}$.

(c) Propose an agent architecture that minimizes the number of private data accesses.

(d) Discuss the tension between agent autonomy and privacy preservation.

Exercise 6. Scalable Oversight: Consider a debate protocol where two agents argue about the correct answer to a complex reasoning task.

(a) Formalize the debate as a two-player zero-sum game.

(b) Show that if the judge can verify atomic claims, the honest debater has a winning strategy.

(c) Relate the oversight cost to the depth of the debate tree.

(d) Discuss limitations of debate as an oversight mechanism.

15 Mathematical Background

15.1 Probability Theory Foundations

15.1.1 Random Variables and Distributions

Definition 15.1 (Random Variable)**.** A random variable X is a measurable function from a probability space $(\Omega, \mathcal{F}, P)$ to a measurable space $(E, \mathcal{E})$:

$$X : \Omega \to E \tag{304}$$

where Ω is the sample space, and E is the measurable space (e.g., $\mathbb{R}$). For discrete random variables, $E = \{x_1, x_2, \ldots\}$ and the distribution is characterized by the probability mass function $P(X = x_i) = p_i$.

Definition 15.2 (Probability Density Function)**.** For continuous random variables, the probability density function $f_X(x)$ satisfies:

$$P(a \leq X \leq b) = \int_a^b f_X(x)dx \tag{305}$$

where $f_X(x)$ is the PDF, and the integral represents the probability of X falling in the interval $[a, b]$. with $f_X(x) \geq 0$ and $\int_{-\infty}^{\infty} f_X(x)dx = 1$.

15.1.2 Common Distributions

- **Gaussian (Normal)**: $X \sim \mathcal{N}(\mu, \sigma^2)$ with $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
- **Laplace**: $X \sim \text{Lap}(b)$ with $f(x) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$
- **Exponential**: $X \sim \text{Exp}(\lambda)$ with $f(x) = \lambda e^{-\lambda x}$ for $x \geq 0$
- **Bernoulli**: $X \sim \text{Bern}(p)$ with $P(X = 1) = p$, $P(X = 0) = 1 - p$
- **Categorical**: $X \sim \text{Cat}(\mathbf{p})$ with $P(X = k) = p_k$ for $k \in \{1, \ldots, K\}$
- **Dirichlet**: $\mathbf{p} \sim \text{Dir}(\boldsymbol{\alpha})$ with $f(\mathbf{p}) \propto \prod_k p_k^{\alpha_k - 1}$

15.1.3 Expectation and Moments

Definition 15.3 (Expectation)**.** For a random variable X with distribution P:

$$\mathbb{E}[X] = \int x \, dP(x) = \begin{cases} \sum_x x \cdot P(X = x) & \text{discrete} \\ \int_{-\infty}^{\infty} x \cdot f_X(x)dx & \text{continuous} \end{cases} \tag{306}$$

where $\mathbb{E}[X]$ is the expected value (mean) of X.

Definition 15.4 (Variance and Standard Deviation)**.**

$$\text{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2, \quad \sigma_X = \sqrt{\text{Var}[X]} \tag{307}$$

where $\text{Var}[X]$ is the variance, and σ_X is the standard deviation.

Definition 15.5 (Covariance and Correlation)**.**

$$\text{Cov}[X, Y] = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])], \quad \rho_{XY} = \frac{\text{Cov}[X, Y]}{\sigma_X \sigma_Y} \tag{308}$$

where $\text{Cov}[X, Y]$ is the covariance, and ρ_{XY} is the correlation coefficient.

15.1.4 Conditional Probability and Independence

Definition 15.6 (Conditional Probability)**.** For events A and B with $P(B) > 0$:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \tag{309}$$

where $P(A|B)$ is the conditional probability of event A given event B.

Theorem 15.1 (Bayes' Theorem)**.**

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)P(A)}{\sum_a P(B|A=a)P(A=a)} \tag{310}$$

where $P(B|A)$ is the likelihood, $P(A)$ is the prior, and $P(B)$ is the marginal likelihood (evidence).

Definition 15.7 (Independence)**.** Events A and B are independent ($A \perp B$) if $P(A \cap B) = P(A)P(B)$, equivalently $P(A|B) = P(A)$. Random variables $X \perp Y$ if $P(X, Y) = P(X)P(Y)$.

Definition 15.8 (Conditional Independence)**.** $X \perp Y|Z$ if $P(X, Y|Z) = P(X|Z)P(Y|Z)$.

15.2 Concentration Inequalities

Concentration inequalities bound how far random quantities deviate from their expectations.

Theorem 15.2 (Chebyshev's Inequality)**.** *For random variable X with mean μ and variance σ^2:*

$$P(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2} \tag{311}$$

where $k > 0$ is the number of standard deviations.

Theorem 15.3 (Hoeffding's Inequality)**.** *Let $X_1, \ldots, X_n$ be independent random variables such that $a_i \leq X_i \leq b_i$. Let $S_n = \sum X_i$. Then:*

$$P(|S_n - \mathbb{E}[S_n]| \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum(b_i - a_i)^2}\right) \tag{312}$$

where $t > 0$ is the deviation, and $[a_i, b_i]$ is the range of X_i.

Theorem 15.4 (McDiarmid's Inequality)**.** *Let $f : \mathcal{X}^n \to \mathbb{R}$ satisfy the bounded difference property with constants $c_1, \ldots, c_n$:*

$$\sup_{x_1,\ldots,x_n,x_i'} |f(x_1, \ldots, x_i, \ldots, x_n) - f(x_1, \ldots, x_i', \ldots, x_n)| \leq c_i \tag{313}$$

Then for any $t > 0$:

$$P(|f(X) - \mathbb{E}[f(X)]| \geq t) \leq 2\exp\left(-\frac{2t^2}{\sum c_i^2}\right) \tag{314}$$

where c_i bounds the change in f when the i-th input changes.

Theorem 15.5 (Bernstein's Inequality)**.** *Let $X_1, \ldots, X_n$ be independent with $\mathbb{E}[X_i] = 0$, $|X_i| \leq M$, and $\sum Var[X_i] \leq V$. Then:*

$$P\left(\left|\sum_{i=1}^{n} X_i\right| \geq t\right) \leq 2\exp\left(-\frac{t^2/2}{V + Mt/3}\right) \tag{315}$$

where M is the bound on $|X_i|$, and V is the total variance.

Theorem 15.6 (Sub-Gaussian Concentration)**.** *A random variable X is σ-sub-Gaussian if for all $\lambda \in \mathbb{R}$:*

$$\mathbb{E}[e^{\lambda(X - \mathbb{E}[X])}] \leq e^{\lambda^2\sigma^2/2} \tag{316}$$

where σ is the sub-Gaussian parameter.

For σ-sub-Gaussian X: $P(|X - \mathbb{E}[X]| \geq t) \leq 2e^{-t^2/(2\sigma^2)}$.

15.3 Information Theory

Definition 15.9 (Entropy)**.** For discrete random variable X with distribution P:

$$H(X) = -\sum_x P(x) \log P(x) = \mathbb{E}[-\log P(X)] \tag{317}$$

where $H(X)$ is the entropy, and $P(x)$ is the probability mass function. Entropy measures the average information content or uncertainty.

Definition 15.10 (Conditional Entropy)**.**

$$H(Y|X) = \mathbb{E}_X[H(Y|X = x)] = -\sum_{x,y} P(x, y) \log P(y|x) \tag{318}$$

where $H(Y|X)$ is the conditional entropy of Y given X.

Definition 15.11 (Mutual Information)**.**

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = \sum_{x,y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)} \tag{319}$$

where $I(X;Y)$ is the mutual information. Mutual information measures the information shared between X and Y.

Definition 15.12 (KL Divergence)**.** For discrete distributions P and Q:

$$D_{KL}(P\|Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)} = \mathbb{E}_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right] \tag{320}$$

where $D_{KL}(P\|Q)$ is the Kullback-Leibler divergence from Q to P. KL divergence is non-negative with $D_{KL}(P\|Q) = 0$ iff $P = Q$.

Theorem 15.7 (Pinsker's Inequality)**.**

$$\delta_{TV}(P, Q) = \frac{1}{2} \sum_x |P(x) - Q(x)| \leq \sqrt{\frac{1}{2} D_{KL}(P\|Q)} \tag{321}$$

where $\delta_{TV}(P, Q)$ is the total variation distance.

Definition 15.13 (Cross-Entropy)**.**

$$H(P, Q) = -\sum_x P(x) \log Q(x) = H(P) + D_{KL}(P\|Q) \tag{322}$$

where $H(P, Q)$ is the cross-entropy. Cross-entropy is minimized when $Q = P$.

Definition 15.14 (Renyi Divergence)**.** For $\alpha > 0$, $\alpha \neq 1$:

$$D_\alpha(P\|Q) = \frac{1}{\alpha - 1} \log \sum_x P(x)^\alpha Q(x)^{1-\alpha} \tag{323}$$

where $D_\alpha(P\|Q)$ is the Rényi divergence of order α. $D_1(P\|Q) = \lim_{\alpha\to 1} D_\alpha(P\|Q) = D_{KL}(P\|Q)$.

Theorem 15.8 (Data Processing Inequality)**.** *For Markov chain $X \to Y \to Z$:*

$$I(X; Z) \leq I(X; Y) \tag{324}$$

where $I(X; Z)$ is the mutual information between X and Z. Post-processing cannot increase information.

15.4 Convex Optimization

Definition 15.15 (Convex Set)**.** A set C is convex if for all $x, y \in C$ and $\theta \in [0, 1]$:

$$\theta x + (1 - \theta)y \in C \tag{325}$$

where C is the convex set, and θ is the interpolation parameter.

Definition 15.16 (Convex Function)**.** $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $x, y \in \text{dom}(f)$ and $\theta \in [0, 1]$:

$$f(\theta x + (1 - \theta)y) \leq \theta f(x) + (1 - \theta)f(y) \tag{326}$$

where f is the convex function. Equivalently, f is convex iff its epigraph $\{(x, t) : f(x) \leq t\}$ is convex.

Theorem 15.9 (First-Order Characterization)**.** *For differentiable f, convexity is equivalent to:*

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) \quad \forall x, y \tag{327}$$

where $\nabla f(x)$ is the gradient of f at x.

Theorem 15.10 (Second-Order Characterization)**.** *For twice-differentiable f, convexity is equivalent to:*

$$\nabla^2 f(x) \succeq 0 \quad \forall x \tag{328}$$

where $\nabla^2 f(x)$ is the Hessian matrix (positive semidefinite). (Hessian is positive semidefinite).

Theorem 15.11 (Jensen's Inequality)**.** *If f is convex and X is a random variable, then:*

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)] \tag{329}$$

where $\mathbb{E}[X]$ is the expectation of X.

Definition 15.17 (Strong Convexity)**.** f is μ-strongly convex if $f(x) - \frac{\mu}{2}\|x\|^2$ is convex, equivalently:

$$f(y) \geq f(x) + \nabla f(x)^T(y - x) + \frac{\mu}{2}\|y - x\|^2 \tag{330}$$

where $\mu > 0$ is the strong convexity parameter.

Definition 15.18 (Smoothness)**.** f is L-smooth if ∇f is L-Lipschitz:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \tag{331}$$

where L is the smoothness constant (Lipschitz constant of the gradient). Equivalently: $f(y) \leq f(x) + \nabla f(x)^T(y - x) + \frac{L}{2}\|y - x\|^2$.

Theorem 15.12 (Gradient Descent Convergence)**.** *For L-smooth, μ-strongly convex f, gradient descent with step size $\eta = 1/L$ satisfies:*

$$f(x_t) - f(x^*) \leq \left(1 - \frac{\mu}{L}\right)^t (f(x_0) - f(x^*)) \tag{332}$$

where x_t is the iterate at step t, x^* is the optimum, and $\kappa = L/\mu$ is the condition number. The condition number $\kappa = L/\mu$ determines convergence rate.

Theorem 15.13 (Singular Value Decomposition)**.** *Any matrix $A \in \mathbb{R}^{m \times n}$ has decomposition:*

$$A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{333}$$

where $U \in \mathbb{R}^{m \times r}$, $V \in \mathbb{R}^{n \times r}$ have orthonormal columns, and $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$ with $\sigma_1 \geq \cdots \geq \sigma_r > 0$.

Theorem 15.14 (Eckart-Young Theorem)**.** *The best rank-k approximation in Frobenius norm is:*

$$A_k = \underset{rank(B)\leq k}{\arg\min} \|A - B\|_F = \sum_{i=1}^{k} \sigma_i u_i v_i^T \tag{334}$$

with error $\|A - A_k\|_F = \sqrt{\sum_{i>k} \sigma_i^2}$.

Definition 15.19 (Positive Semidefinite Matrices)**.** A symmetric matrix A is positive semidefinite ($A \succeq 0$) if all eigenvalues are non-negative, or equivalently $x^T Ax \geq 0$ for all x.

16 Advanced Proofs and Derivations

16.1 Proof of the VC Dimension Lower Bound

Theorem 16.1 (Fundamental Theorem of Statistical Learning)**.** *Let $\mathcal{H}$ be a hypothesis class with VC dimension d. For any $\delta \in (0,1)$ and any $\epsilon > 0$:*

1. *(Upper Bound) $m \geq \frac{8}{\epsilon^2}\left(d \log \frac{16}{\epsilon} + \log \frac{2}{\delta}\right)$ samples suffice for PAC learning*
2. *(Lower Bound) $m \geq \frac{d-1}{32\epsilon}$ samples are necessary*

where m is the sample size, d is the VC dimension, ϵ is the error tolerance, and δ is the failure probability.

Proof of Lower Bound Sketch. We use the probabilistic method. Consider a set of d points that can be shattered by $\mathcal{H}$. For each labeling $\sigma \in \{0,1\}^d$, let h_σ be the hypothesis achieving this labeling.

The adversary chooses σ uniformly at random. Any algorithm seeing m samples has probability at most $2^m/2^d = 2^{m-d}$ of guessing σ correctly.

For the algorithm to achieve error $\leq \epsilon$ with probability $\geq 1-\delta$, it must correctly classify at least $(1-\epsilon)d$ of the d shattered points. This requires learning approximately $(1-\epsilon)d$ bits of information about σ.

By information-theoretic arguments, $m \geq \Omega((1-\epsilon)d) = \Omega(d)$ samples are necessary. A more careful analysis gives $m \geq \frac{d-1}{32\epsilon}$. □

16.2 Derivation of the Evidence Lower Bound (ELBO)

In Variational Autoencoders (VAEs) and Bayesian inference, we often wish to approximate an intractable posterior $P(Z|X)$. The ELBO provides a tractable lower bound on the marginal log-likelihood that can be optimized via gradient descent.

Theorem 16.2 (Evidence Lower Bound)**.** *For any distribution $Q(Z|X)$ over latent variables Z:*

$$\log P(X) \geq \underbrace{\mathbb{E}_{z\sim Q}[\log P(X|z)]}_{\textit{Reconstruction}} - \underbrace{D_{KL}(Q(Z|X)\|P(Z))}_{\textit{Regularization}} =: \textit{ELBO} \tag{335}$$

where $P(X)$ is the marginal likelihood, $Q(Z|X)$ is the approximate posterior, $P(X|z)$ is the likelihood, and $P(Z)$ is the prior. with equality if and only if $Q(Z|X) = P(Z|X)$.

Proof. We start with the KL divergence between the approximate and true posterior (which is always non-negative):

$$0 \leq D_{KL}(Q(Z|X)\|P(Z|X)) = \mathbb{E}_{z\sim Q}\left[\log \frac{Q(z|X)}{P(z|X)}\right] \tag{336}$$

where $D_{KL}(Q(Z|X)\|P(Z|X))$ is the KL divergence between the approximate and true posteriors.

Applying Bayes' rule, $P(z|X) = \frac{P(X|z)P(z)}{P(X)}$:

$$D_{KL}(Q(Z|X)\|P(Z|X)) = \mathbb{E}_{z\sim Q}\left[\log Q(z|X) - \log P(X|z) - \log P(z) + \log P(X)\right] \tag{337}$$

Since $\log P(X)$ is constant with respect to z:

$$D_{KL}(Q(Z|X)\|P(Z|X)) = \log P(X) - \mathbb{E}_{z\sim Q}[\log P(X|z)] + \mathbb{E}_{z\sim Q}\left[\log \frac{Q(z|X)}{P(z)}\right] \tag{338}$$

$$= \log P(X) - \mathbb{E}_{z\sim Q}[\log P(X|z)] + D_{KL}(Q(Z|X)\|P(Z)) \tag{339}$$

Rearranging and using non-negativity of KL:

$$\log P(X) = \underbrace{\mathbb{E}_{z\sim Q}[\log P(X|z)] - D_{KL}(Q(Z|X)\|P(Z))}_{\text{ELBO}} + \underbrace{D_{KL}(Q(Z|X)\|P(Z|X))}_{\geq 0} \tag{340}$$

Therefore:

$$\boxed{\log P(X) \geq \text{ELBO} = \mathbb{E}_{z\sim Q}[\log P(X|z)] - D_{KL}(Q(Z|X)\|P(Z))} \tag{341}$$

□

Intuition: The ELBO balances two objectives:

- **Reconstruction Term** $\mathbb{E}[\log P(X|Z)]$: Encourages the model to explain the observed data well by maximizing the likelihood of X given the latent representation Z.
- **KL Regularization** $D_{KL}(Q(Z|X)\|P(Z))$: Prevents $Q(Z|X)$ from diverging too far from the prior $P(Z)$ (often a standard Gaussian), acting as a regularizer that encourages smooth, generalizable latent representations.

16.3 Proof of Differential Privacy Composition

Theorem 16.3 (Basic Composition)**.** *If $\mathcal{M}_1$ is (ϵ_1, δ_1)-DP and $\mathcal{M}_2$ is (ϵ_2, δ_2)-DP, then the composition $(\mathcal{M}_1(D), \mathcal{M}_2(D))$ is $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP.*

Proof. For any measurable set S in the output space of the joint mechanism and adjacent datasets D, D', we condition on the output of $\mathcal{M}_1$:

$$P((\mathcal{M}_1, \mathcal{M}_2)(D) \in S) = \int P(\mathcal{M}_2(D) \in S_{o_1})\, dP_{\mathcal{M}_1(D)}(o_1) \tag{342}$$

where $S_{o_1} = \{o_2 : (o_1, o_2) \in S\}$.

By (ϵ_2, δ_2)-DP of $\mathcal{M}_2$, for each fixed o_1:

$$P(\mathcal{M}_2(D) \in S_{o_1}) \leq e^{\epsilon_2} P(\mathcal{M}_2(D') \in S_{o_1}) + \delta_2 \tag{343}$$

Substituting and applying (ϵ_1, δ_1)-DP of $\mathcal{M}_1$:

$$P((\mathcal{M}_1, \mathcal{M}_2)(D) \in S) \leq e^{\epsilon_2} \int P(\mathcal{M}_2(D') \in S_{o_1})\, dP_{\mathcal{M}_1(D)}(o_1) + \delta_2 \tag{344}$$

$$\leq e^{\epsilon_2} \left(e^{\epsilon_1} \int P(\mathcal{M}_2(D') \in S_{o_1})\, dP_{\mathcal{M}_1(D')}(o_1) + \delta_1 \right) + \delta_2 \tag{345}$$

$$= e^{\epsilon_1+\epsilon_2} P((\mathcal{M}_1, \mathcal{M}_2)(D') \in S) + e^{\epsilon_2}\delta_1 + \delta_2 \tag{346}$$

This gives $(\epsilon_1 + \epsilon_2, e^{\epsilon_2}\delta_1 + \delta_2)$-DP. A tighter analysis that applies the DP definition to each mechanism sequentially (rather than conditioning on $\mathcal{M}_1$'s output) yields the standard result $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$-DP. The key idea is to split the output space of $\mathcal{M}_1$ into a "good" set (where the e^{ϵ_1} bound holds) and a "bad" set of measure at most δ_1, and similarly for $\mathcal{M}_2$, then apply a union bound on the failure events. □

Theorem 16.4 (Advanced Composition)**.** *For k applications of (ϵ, δ)-DP mechanisms, the composition is $(\epsilon', k\delta + \delta')$-DP where:*

$$\epsilon' = \sqrt{2k \ln(1/\delta')} \cdot \epsilon + k\epsilon(e^\epsilon - 1) \tag{347}$$

where ϵ' is the composed privacy budget, k is the number of compositions, and δ' is the slack parameter. For small ϵ, this is approximately $O(\sqrt{k}\epsilon)$ rather than $O(k\epsilon)$.

16.4 Convergence of SGD for Convex Objectives

We analyze the convergence of Stochastic Gradient Descent (SGD) for a convex, L-Lipschitz function f.

Theorem 16.5. *Let f be convex and L-Lipschitz. Let $x^* \in \arg\min f(x)$ with $\|x_1 - x^*\| \leq R$. If we run SGD with step size $\eta_t = \frac{R}{L\sqrt{T}}$, then:*

$$\mathbb{E}\left[f\left(\frac{1}{T}\sum_{t=1}^{T} x_t\right)\right] - f(x^*) \leq \frac{RL}{\sqrt{T}} \tag{348}$$

where T is the number of iterations, R is the radius of the domain, and L is the Lipschitz constant.

Proof. Let g_t be the stochastic gradient at step t, such that $\mathbb{E}[g_t] = \nabla f(x_t)$. The update rule is $x_{t+1} = x_t - \eta g_t$. Consider the distance to the optimum:

$$\|x_{t+1} - x^*\|^2 = \|x_t - \eta g_t - x^*\|^2 \tag{349}$$

$$= \|x_t - x^*\|^2 - 2\eta\langle g_t, x_t - x^*\rangle + \eta^2\|g_t\|^2 \tag{350}$$

where η is the step size. Taking expectations:

$$\mathbb{E}[\|x_{t+1} - x^*\|^2] = \mathbb{E}[\|x_t - x^*\|^2] - 2\eta\,\mathbb{E}[\langle \nabla f(x_t), x_t - x^*\rangle] + \eta^2\,\mathbb{E}[\|g_t\|^2] \tag{351}$$

By convexity, $f(x^*) \geq f(x_t) + \langle \nabla f(x_t), x^* - x_t\rangle$, so $\langle \nabla f(x_t), x_t - x^*\rangle \geq f(x_t) - f(x^*)$. Also, assume bounded gradients $\mathbb{E}[\|g_t\|^2] \leq L^2$.

$$\mathbb{E}[\|x_{t+1} - x^*\|^2] \leq \mathbb{E}[\|x_t - x^*\|^2] - 2\eta\,\mathbb{E}[f(x_t) - f(x^*)] + \eta^2 L^2 \tag{352}$$

Rearranging and summing over $t = 1 \dots T$:

$$2\eta\sum_{t=1}^{T}\mathbb{E}[f(x_t) - f(x^*)] \leq \|x_1 - x^*\|^2 - \mathbb{E}[\|x_{T+1} - x^*\|^2] + T\eta^2 L^2 \tag{353}$$

Since $\|x_{T+1} - x^*\|^2 \geq 0$ and $\|x_1 - x^*\|^2 \leq R^2$:

$$\sum_{t=1}^{T}\mathbb{E}[f(x_t) - f(x^*)] \leq \frac{R^2}{2\eta} + \frac{T\eta L^2}{2} \tag{354}$$

Using Jensen's inequality on the LHS (convexity of f) and setting $\eta = \frac{R}{L\sqrt{T}}$ yields the result. □

16.5 Proof of Rademacher Complexity Bound

Theorem 16.6 (Rademacher Complexity Generalization Bound)**.** *For any $\delta > 0$, with probability at least $1 - \delta$ over the draw of $S \sim \mathcal{D}^n$, for all $h \in \mathcal{H}$:*

$$R(h) \leq \hat{R}(h) + 2\hat{\mathcal{R}}_S(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2n}} \tag{355}$$

where $R(h)$ is the true risk, $\hat{R}(h)$ is the empirical risk, $\hat{\mathcal{R}}_S$ is the empirical Rademacher complexity, and n is the sample size.

Proof Sketch. The proof proceeds in three steps:

Step 1: Symmetrization. Using a ghost sample S', we show:

$$\mathbb{E}_S\left[\sup_h (R(h) - \hat{R}_S(h))\right] \leq 2\,\mathbb{E}_{S,\sigma}\left[\sup_h \frac{1}{n}\sum_{i=1}^{n} \sigma_i \ell(h, z_i)\right] \tag{356}$$

where σ_i are i.i.d. Rademacher variables ($P(\sigma_i = \pm 1) = 1/2$).

Step 2: Concentration. Using McDiarmid's inequality with bounded differences $c_i = 2/n$:

$$P\left(\sup_h |R(h) - \hat{R}(h)| \geq \mathbb{E}[\sup_h |R(h) - \hat{R}(h)|] + t\right) \leq e^{-nt^2/2} \tag{357}$$

Step 3: Combine. Setting $t = \sqrt{\log(2/\delta)/(2n)}$ gives the result. □

16.6 Shapley Value Axioms and Uniqueness

The Shapley value is the *unique* solution satisfying four axioms.

Definition 16.1 (Shapley Axioms)**.** Let $\phi_i(v)$ be the value allocated to player i, v the characteristic function, and N the set of players.

1. **Efficiency**: $\sum_{i \in N} \phi_i(v) = v(N) - v(\emptyset)$.
2. **Symmetry**: If $v(S \cup \{i\}) = v(S \cup \{j\})$ for all $S \subseteq N \setminus \{i, j\}$, then $\phi_i(v) = \phi_j(v)$.
3. **Null Player**: If $v(S \cup \{i\}) = v(S)$ for all $S \subseteq N \setminus \{i\}$, then $\phi_i(v) = 0$.
4. **Linearity (Additivity)**: For games v and w, $\phi_i(v+w) = \phi_i(v) + \phi_i(w)$, and $\phi_i(cv) = c\,\phi_i(v)$ for $c \in \mathbb{R}$.

Theorem 16.7 (Shapley Uniqueness)**.** *The unique allocation satisfying all four axioms is:*

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!}[v(S \cup \{i\}) - v(S)] \tag{358}$$

where the sum is over all subsets S not containing i.

Proof Sketch. The formula can be derived by considering all possible orderings of players joining the coalition. Player i arrives at position k with probability $1/|N|!$ for each of the $(|S|)! \times (|N| - |S| - 1)!$ orderings where $|S|$ players come before i. The marginal contribution is $v(S \cup \{i\}) - v(S)$, and averaging gives the formula.

Uniqueness follows from showing that any function satisfying the axioms must equal this formula. The key insight is that by linearity (additivity), it suffices to verify for "unanimity games" where $v(S) = 1$ iff $T \subseteq S$ for some coalition T. □

17 Python Implementation Details

This section provides concrete implementation details for key algorithms discussed in the text, using PyTorch.

17.1 Fast Gradient Sign Method (FGSM)

```python
import torch
import torch.nn.functional as F

def fgsm_attack(image, epsilon, data_grad):
    """Generate adversarial example using FGSM for medical image analysis.

    Args:
        image: Original input image tensor (e.g., MRI or X-ray scan)
        epsilon: Maximum perturbation magnitude limits
        data_grad: Gradient of loss w.r.t. input scan

    Returns:
        Perturbed adversarial image for robustness testing
    """
    # Collect the element-wise sign of the data gradient
    sign_data_grad = data_grad.sign()
    # Create perturbed image
    perturbed_image = image + epsilon * sign_data_grad
    # Clip to maintain valid image range [0,1]
    perturbed_image = torch.clamp(perturbed_image, 0, 1)
    return perturbed_image

# Example usage on medical diagnostic model:
# Xray_scan.requires_grad = True
# diagnostic_output = disease_model(Xray_scan)
# loss = F.cross_entropy(diagnostic_output, true_diagnosis)
# disease_model.zero_grad()
# loss.backward()
# perturbed_scan = fgsm_attack(Xray_scan, epsilon=0.03, data_grad=Xray_scan.grad)
```

Listing 4: FGSM Attack Implementation

17.2 Differential Privacy: Laplace Mechanism

```python
import numpy as np

def laplace_mechanism(true_value, sensitivity, epsilon):
    """Add Laplace noise to protect sensitive medical telemetry (e.g., patient vital signs).

    Args:
        true_value: True query result (e.g., average blood pressure across cohort)
        sensitivity: L1 sensitivity of the healthcare query
        epsilon: Privacy budget (smaller = stricter patient data privacy)
```

```
10
11      Returns:
12          Noisy query result satisfying epsilon-DP requirements
13      """
14      scale = sensitivity / epsilon
15      noise = np.random.laplace(loc=0, scale=scale,
            size=np.shape(true_value))
16      return true_value + noise
17
18  # Example: Private mean computation on hospital patient demographics
19  def private_mean(patient_ages_or_vitals, lower_bound, upper_bound,
        epsilon):
20      """Compute differentially private mean for a medical study."""
21      n = len(patient_ages_or_vitals)
22      true_mean = np.mean(patient_ages_or_vitals)
23      # Sensitivity of mean in bounded health metric is (upper - lower)
            / n
24      sensitivity = (upper_bound - lower_bound) / n
25      return laplace_mechanism(true_mean, sensitivity, epsilon)
```

Listing 5: Laplace Mechanism for Differential Privacy

17.3 Fairness Regularization

```
1   import torch
2
3   def demographic_parity_regularizer(risk_predictions, sensitive_trait):
4       """Enforce Demographic Parity in healthcare risk management or
            medical triaging models.
5
6       Mitigates bias corresponding to patient demographic groups by
            minimizing
7       covariance between severity predictions and protected
            sensitive_trait
8       (e.g., gender, race, or socioeconomic index).
9
10      Args:
11          risk_predictions: Predicted disease severity prob., shape
                (batch_size,)
12          sensitive_trait: Indicator for protected demographic
                attribute, shape (batch_size,)
13
14      Returns:
15          Squared covariance term to penalize biased triage predictions
16      """
17      mean_risk = torch.mean(risk_predictions)
18      mean_trait = torch.mean(sensitive_trait.float())
19
20      # Measure linear dependence linking race/gender and perceived
            health risk
21      covariance = torch.mean(
22          (risk_predictions - mean_risk) * (sensitive_trait.float() -
                mean_trait)
23      )
24      return covariance ** 2
25
26  # Usage during medical model training:
```

```
# total_clinical_loss = bce_diagnosis_loss(risk_predictions,
    true_pathology) + lambda_fair *
    demographic_parity_regularizer(risk_predictions, patient_ethnicity)
```

Listing 6: Differentiable Fairness Regularization

17.4 Projected Gradient Descent Attack

```
import torch
import torch.nn.functional as F

def pgd_attack(mri_classifier, original_scans, diagnostic_labels,
    epsilon, alpha, num_iter):
    """Projected Gradient Descent evaluating robustness of diagnostic
        models.

    Generates strong synthetic adversarial pathologies that might
        deceive
    the healthcare classifier (e.g. skin cancer or MRI anomaly
        detection).

    Args:
        mri_classifier: Medical Neural Network processing scans
        original_scans: Healthy/Unhealthy tissue images (channels,
            height, width)
        diagnostic_labels: True ground-truth diagnoses
        epsilon: Maximum pixel intensity disturbance tolerance
        alpha: Step learning size during malicious anomaly synthesis
        num_iter: Steps attempting to bypass diagnostic thresholds

    Returns:
        Medical imaging tensors simulating tricky adversarial
            edge-cases
    """
    adv_scans = original_scans.clone().detach()

    # Random initial micro-disturbances simulating sensor noise within
        limits
    adv_scans = adv_scans +
       torch.empty_like(adv_scans).uniform_(-epsilon, epsilon)
    adv_scans = torch.clamp(adv_scans, 0, 1)

    for _ in range(num_iter):
        adv_scans.requires_grad = True

        # Test classifier robustness to currently manipulated scan
        predictions = mri_classifier(adv_scans)
        diagnosis_error = F.cross_entropy(predictions,
            diagnostic_labels)

        # Compute how to change image to maximally disrupt medical
            diagnosis
        grad_disruption = torch.autograd.grad(diagnosis_error,
            adv_scans)[0]

        # Attack strategy: maximize the clinical misdiagnosis chance
        adv_scans = adv_scans.detach() + alpha * grad_disruption.sign()
```

```
39
40          # Restrict generated scan artifacts to biologically plausible
                L-infinity limits
41          delta_noise = torch.clamp(adv_scans - original_scans,
                -epsilon, epsilon)
42          adv_scans = torch.clamp(original_scans + delta_noise, 0, 1)
43
44      return adv_scans.detach()
45
46  # Example probing a diabetic retinopathy screening model:
47  # robust_test_data = pgd_attack(retinal_net, fundus_images,
        retina_health_labels, epsilon=8/255, alpha=2/255, num_iter=20)
```

Listing 7: PGD Attack Implementation

17.5 Adversarial Training Loop

```
1  import torch
2  import torch.nn as nn
3  import torch.optim as optim
4
5  def adversarial_training(biomedical_cnn, mri_dataloader, epochs,
       epsilon, alpha, num_iter, device):
6      """Rigorous robust training over diverse medical imaging datasets.
7
8      Helps immunize clinical AI architectures against imaging artifacts,
9      scan anomalies, and equipment calibration failures.
10
11     Args:
12         biomedical_cnn: e.g. Pathological tumor segmentation model
13         mri_dataloader: Source stream of legitimate patient scans
14         epochs: Duration of robust medical curriculum
15         epsilon, alpha, num_iter: Defending against synthetically
               complex variations
16         device: Computational target processor (e.g., hospital GPU
               nodes)
17     """
18     biomedical_cnn.to(device)
19     optimizer = optim.SGD(biomedical_cnn.parameters(), lr=0.01,
           momentum=0.9)
20     criterion = nn.CrossEntropyLoss()
21
22     for epoch in range(epochs):
23         biomedical_cnn.train()
24         running_clinical_loss = 0
25         successful_diagnoses = 0
26         total_evaluations = 0
27
28         for batch_idx, (real_scans, actual_diagnostics) in
               enumerate(mri_dataloader):
29             real_scans, actual_diagnostics = real_scans.to(device),
                   actual_diagnostics.to(device)
30
31             # Formulate simulated scanner measurement degradations or
                   adversarial cases
32             biomedical_cnn.eval()
```

```
            degraded_scans = pgd_attack(biomedical_cnn, real_scans,
                actual_diagnostics, epsilon, alpha, num_iter)
            biomedical_cnn.train()

            # Educate model on robust feature extraction
            optimizer.zero_grad()
            diagnostic_calls = biomedical_cnn(degraded_scans)
            diagnostic_error = criterion(diagnostic_calls,
                actual_diagnostics)

            # Penalize the architecture for failing on adversarial
                anomalies
            diagnostic_error.backward()
            optimizer.step()

            # Clinical performance tracking
            running_clinical_loss += diagnostic_error.item()
            _, chosen_clinical_labels = diagnostic_calls.max(1)
            total_evaluations += actual_diagnostics.size(0)
            successful_diagnoses +=
                chosen_clinical_labels.eq(actual_diagnostics).sum().item()

        clinical_accuracy = 100.0 * successful_diagnoses /
            total_evaluations
        print(f'Training Epoch {epoch}: Clinical
            Error={running_clinical_loss/len(mri_dataloader):.4f},
            Robustness Score={clinical_accuracy:.2f}%')
```

Listing 8: Complete Adversarial Training

17.6 Conformal Prediction

```
import numpy as np

class ConformalClassifier:
    """Rigorous conformal uncertainty quantification wrapping a
        clinical predictor."""

    def __init__(self, diagnostic_system,
        tolerable_clinical_error=0.1):
        """
        Provides mathematical guarantees for diagnosis coverage sets
            based on
        the calibration population.
        Args:
            diagnostic_system: Supervised model extracting illness
                probabilities
            tolerable_clinical_error: Maximum acceptable diagnosis
                omission risk (1 - confidence)
        """
        self.diagnostic_system = diagnostic_system
        self.alpha_error_bound = tolerable_clinical_error
        self.empirical_threshold = None

    def calibrate(self, clinical_test_covariates, true_patient_labels):
        """Estimate the safe risk-threshold using a held-out patient
            study group."""
```

```
        medical_confidence_scores =
            self.diagnostic_system.predict_proba(clinical_test_covariates)

        # Non-conformity quantifies how strange a patient seems
            compared to historical data
        cohort_size = len(true_patient_labels)
        disagreement_metrics = 1 -
            medical_confidence_scores[np.arange(cohort_size),
            true_patient_labels]

        # Establishing a proven cutoff for the targeted strictness
        target_quantile_idx = int(np.ceil((cohort_size + 1) * (1 -
            self.alpha_error_bound)))
        if target_quantile_idx > cohort_size:
            self.empirical_threshold = np.inf
        else:
            self.empirical_threshold =
                np.sort(disagreement_metrics)[target_quantile_idx - 1]

        return self

    def predict_set(self, new_patient_profiles):
        """Generate sets of potential disease conditions guaranteed
            bounding error margins."""
        symptom_prob_estimates =
            self.diagnostic_system.predict_proba(new_patient_profiles)

        # Create a set containing all conditions lacking evidence for
            outright dismissal
        plausible_diagnoses = []
        for risk_profile in symptom_prob_estimates:
            retained_conditions = np.where(1 - risk_profile <=
                self.empirical_threshold)[0]
            plausible_diagnoses.append(set(retained_conditions))

        return plausible_diagnoses

# Example screening deployment assuring clinical coverage > 90%:
# biopsy_model = ClinicalRandomForestifier().fit(symptoms_train,
    pathology_train)
# tight_interval_ai = ConformalClassifier(biopsy_model,
    tolerable_clinical_error=0.1)
# tight_interval_ai.calibrate(symptoms_val, biopsy_results_val)
# probable_diseases_list =
    tight_interval_ai.predict_set(new_symptoms_arr)
```

Listing 9: Conformal Prediction for Classification

17.7 Integrated Gradients

```
import torch
import numpy as np

def integrated_gradients(diagnostic_cnn, patient_scan,
    baseline_reference, target_disease_class, steps=50):
    """Compute Explainable Integrated Gradients for medical imaging.

```

```
    Reveals WHICH physiological features (e.g. tissue structures in a scan)
    contributed most to a specific AI medical diagnosis.

    Args:
        diagnostic_cnn: Neural network trained on pathological scans
        patient_scan: Input X-ray/MRI to explain, shape (1, channels, height, width)
        baseline_reference: Blank or healthy tissue baseline template
        target_disease_class: specific illness predicted that needs explanation
        steps: Numerical approximation density for integral calculations

    Returns:
        Feature attribution heatmap matching input dimensions to highlight anomalies
    """
    # Generate interpolated intermediate tissue scans
    scaled_inputs = [baseline_reference + (float(i) / steps) * (patient_scan - baseline_reference)
                     for i in range(steps + 1)]
    scaled_inputs = torch.cat(scaled_inputs, dim=0)
    scaled_inputs.requires_grad = True

    # Ensure network is in evaluation mode to freeze Dropout and Batch Norm
    diagnostic_cnn.eval()

    # Evaluate AI clinical confidence at each interpolation intensity
    diagnostic_outputs = diagnostic_cnn(scaled_inputs)

    # Assess sensitivity concerning particular diagnosis
    target_probas = diagnostic_outputs[:, target_disease_class]
    grads = torch.autograd.grad(target_probas.sum(), scaled_inputs)[0]

    # Calculate expected gradient value using the trapezoidal rule for precise integration
    grads_trapz = (grads[:-1] + grads[1:]) / 2.0
    avg_grads = grads_trapz.mean(dim=0, keepdim=True)

    # Distribute relative importance across the patient's biological scan
    integrated_grad_heatmap = (patient_scan - baseline_reference) * avg_grads

    return integrated_grad_heatmap

def visualize_attribution(attribution, original_image):
    """Visualize attribution as heatmap overlay."""
    import matplotlib.pyplot as plt

    # Sum over channels for visualization
    attr_sum = attribution.squeeze().sum(dim=0).cpu().numpy()

    # Normalize
    attr_sum = (attr_sum - attr_sum.min()) / (attr_sum.max() - attr_sum.min() + 1e-8)
```

```
    fig, axes = plt.subplots(1, 3, figsize=(12, 4))
    axes[0].imshow(original_image.permute(1, 2, 0).cpu().numpy())
    axes[0].set_title('Original')
    axes[1].imshow(attr_sum, cmap='hot')
    axes[1].set_title('Attribution')
    axes[2].imshow(original_image.permute(1, 2, 0).cpu().numpy())
    axes[2].imshow(attr_sum, cmap='hot', alpha=0.5)
    axes[2].set_title('Overlay')

    for ax in axes:
        ax.axis('off')
    plt.tight_layout()
    return fig
```

Listing 10: Integrated Gradients Attribution

18 Comprehensive Evaluation Metrics

This section provides a reference for evaluation metrics used throughout the document.

18.1 Robustness Metrics

Definition 18.1 (Clean Accuracy)**.** The standard classification accuracy on unperturbed test data:

$$\text{Acc}_{\text{clean}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(f(x_i) = y_i) \tag{359}$$

where n is the number of samples, $f(x_i)$ is the predicted label, y_i is the true label, and $\mathbb{I}$ is the indicator function.

Definition 18.2 (Robust Accuracy)**.** Accuracy under adversarial attack with perturbation budget ϵ:

$$\text{Acc}_{\text{robust}}(\epsilon) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left(\min_{\|\delta\|\leq\epsilon} f(x_i + \delta) = y_i\right) \tag{360}$$

where δ is the perturbation vector. In practice, this is approximated by the accuracy under a specific attack (e.g., PGD).

Definition 18.3 (Certified Robust Accuracy)**.** The fraction of samples with a certified radius $\geq \epsilon$:

$$\text{Acc}_{\text{certified}}(\epsilon) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(r(x_i) \geq \epsilon) \tag{361}$$

where $r(x_i)$ is the certified radius from formal verification or randomized smoothing.

Definition 18.4 (Attack Success Rate (ASR))**.** For targeted attacks, the fraction where the adversarial example achieves the target class:

$$\text{ASR} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(f(x_i + \delta_i) = y_{\text{target}}) \tag{362}$$

where y_{target} is the target class.

18.2 Fairness Metrics

Definition 18.5 (Demographic Parity Difference (DPD))**.**

$$\text{DPD} = |P(\hat{Y} = 1|A = 0) - P(\hat{Y} = 1|A = 1)| \tag{363}$$

where $\hat{Y}$ is the predicted label, and A is the sensitive attribute. Values closer to 0 indicate fairer predictions with respect to demographic parity.

Definition 18.6 (Equalized Odds Difference (EOD))**.**

$$\text{EOD} = \frac{1}{2}\left(|\text{TPR}_{A=0} - \text{TPR}_{A=1}| + |\text{FPR}_{A=0} - \text{FPR}_{A=1}|\right) \tag{364}$$

where TPR is the true positive rate, and FPR is the false positive rate.

Definition 18.7 (Equal Opportunity Difference)**.**

$$\text{EO} = |P(\hat{Y} = 1|Y = 1, A = 0) - P(\hat{Y} = 1|Y = 1, A = 1)| \tag{365}$$

where Y is the true label. Focuses only on the true positive rate difference.

Definition 18.8 (Predictive Parity)**.** Requires equal positive predictive values:

$$P(Y = 1|\hat{Y} = 1, A = 0) = P(Y = 1|\hat{Y} = 1, A = 1) \tag{366}$$

where $P(Y = 1|\hat{Y} = 1)$ is the precision.

Definition 18.9 (Disparate Impact Ratio)**.**

$$\text{DIR} = \frac{P(\hat{Y} = 1|A = \text{disadvantaged})}{P(\hat{Y} = 1|A = \text{advantaged})} \tag{367}$$

where values less than 1 indicate disadvantage. The "80% rule" considers DIR < 0.8 as evidence of disparate impact.

18.3 Privacy Metrics

Definition 18.10 (Privacy Budget (ϵ, δ))**.** The parameters of (ϵ, δ)-differential privacy. Smaller ϵ implies stronger privacy. The parameter δ represents the probability of a privacy breach.

Definition 18.11 (Reconstruction Error)**.** For model inversion attacks, the mean squared error between reconstructed and original data:

$$\text{RE} = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \hat{x}_i\|^2 \tag{368}$$

where $\hat{x}_i$ is the reconstructed input.

Definition 18.12 (Maximum Calibration Error (MCE))**.**

$$\text{MCE} = \max_{b \in \{1,\ldots,B\}} |\text{acc}(B_b) - \text{conf}(B_b)| \tag{369}$$

where MCE measures the worst-case calibration error.

Definition 18.13 (Brier Score)**.**

$$\text{BS} = \frac{1}{n}\sum_{i=1}^{n}\sum_{c=1}^{C}(p_{ic} - y_{ic})^2 \tag{370}$$

where p_{ic} is the predicted probability for class c and y_{ic} is the one-hot indicator.

Definition 18.14 (Negative Log-Likelihood (NLL))**.**

$$\text{NLL} = -\frac{1}{n}\sum_{i=1}^{n} \log p_{iy_i} \tag{371}$$

where p_{iy_i} is the predicted probability for the true class y_i. Also known as cross-entropy loss on test data.

18.4 OOD Detection Metrics

Definition 18.15 (AUROC)**.** Area Under the ROC Curve; measures the probability that an in-distribution sample scores higher than an OOD sample.

Definition 18.16 (FPR at 95% TPR (FPR95))**.** False positive rate when true positive rate is 95%; measures the fraction of OOD samples incorrectly classified as in-distribution when 95% of in-distribution samples are correctly identified.

Definition 18.17 (AUPR)**.** Area Under the Precision-Recall Curve; more informative when classes are imbalanced.

18.5 Explainability Metrics

Definition 18.18 (Faithfulness (Deletion))**.** Measure how prediction changes when important features are removed:

$$\text{Faithfulness}_{\text{del}} = \text{AUC}\left(\{f(x_{-S_k})\}_{k=1}^{K}\right) \tag{372}$$

where x_{-S_k} removes the top-k most important features.

Definition 18.19 (Faithfulness (Insertion))**.**

$$\text{Faithfulness}_{\text{ins}} = \text{AUC}\left(\{f(x_{+S_k})\}_{k=1}^{K}\right) \tag{373}$$

where x_{+S_k} starts from baseline and adds top-k features.

Definition 18.20 (Stability)**.** For input x and small perturbation δ:

$$\text{Stability} = \frac{1}{M}\sum_{j=1}^{M} \text{sim}(\phi(x), \phi(x+\delta_j)) \tag{374}$$

where ϕ is the explanation function and sim is a similarity metric.

18.6 Agentic System Metrics

Definition 18.21 (Task Completion Rate (TCR))**.** For an agentic system evaluated on n tasks:

$$\text{TCR} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(f(\tau_i) = \text{success}) \tag{375}$$

where τ_i is the agent trajectory for task i and f is a success evaluator.

Definition 18.22 (Safety Violation Rate (SVR))**.** The fraction of trajectories containing at least one action outside the safe set:

$$\text{SVR} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}\left(\exists\, k : a_k^{(i)} \notin \mathcal{A}_{\text{safe}}\right) \tag{376}$$

Definition 18.23 (Step Efficiency)**.** The ratio of the minimum required steps to the actual steps taken:

$$\text{Efficiency}(\tau) = \frac{T_{\min}}{T_{\text{actual}}} \tag{377}$$

Values near 1 indicate near-optimal agent behavior.

Definition 18.24 (Trajectory-Level Trust Score)**.** A composite metric integrating correctness, safety, and efficiency:

$$\text{Trust}(\tau) = \mathbb{I}(f(\tau) = \text{success}) \cdot \prod_{k=0}^{T-1} \mathbb{I}(a_k \in \mathcal{A}_{\text{safe}}) \cdot \min\left(1, \frac{T_{\min}}{T_{\text{actual}}}\right) \tag{378}$$

Definition 18.25 (Prompt Injection Resistance Rate)**.** The fraction of injected adversarial prompts that the agent correctly ignores:

$$\text{PIRR} = \frac{1}{n_{\text{inj}}} \sum_{i=1}^{n_{\text{inj}}} \mathbb{I}(\text{agent ignores injection}_i) \tag{379}$$

19 Glossary of Key Terms

Adversarial Example
An input crafted by applying small, often imperceptible perturbations to cause a model to make incorrect predictions with high confidence.

Aleatoric Uncertainty
Inherent randomness in the data that cannot be reduced by collecting more data (e.g., sensor noise, labeling ambiguity).

Calibration A property where predicted probabilities match actual outcome frequencies: among predictions with confidence p, the fraction correct should be p.

Certified Robustness
A mathematical guarantee that a model's prediction is invariant to all perturbations within a specified radius.

Conformal Prediction
A distribution-free method for constructing prediction sets with finite-sample coverage guarantees.

Counterfactual Explanation
An explanation of the form "the prediction would have been Y' if the input had been X'," providing actionable recourse.

Demographic Parity
A group fairness criterion requiring equal positive prediction rates across protected groups: $P(\hat{Y} = 1|A = 0) = P(\hat{Y} = 1|A = 1)$.

Differential Privacy
A mathematical framework quantifying privacy loss, ensuring that the output of an algorithm is approximately independent of any single individual's data.

Distribution Shift
A change between the training and deployment data distributions, including covariate shift, label shift, and concept drift.

ELBO Evidence Lower Bound; a tractable lower bound on log-likelihood used in variational inference: $\log p(x) \geq \mathbb{E}_q[\log p(x|z)] - D_{KL}(q(z|x)\|p(z))$.

Epistemic Uncertainty
Uncertainty arising from limited data or model capacity that can be reduced with more data or better models.

Equalized Odds
A fairness criterion requiring equal true positive and false positive rates across groups.

Feature Attribution
Assignment of importance scores to input features indicating their contribution to a model's prediction.

Federated Learning
A distributed learning paradigm where multiple parties collaboratively train a model without sharing raw data.

FGSM Fast Gradient Sign Method; a one-step adversarial attack using the sign of the loss gradient: $x_{adv} = x + \epsilon \cdot \text{sign}(\nabla_x \ell)$.

Hallucination Generation of plausible-sounding but factually incorrect content by language models.

Individual Fairness A fairness criterion requiring similar individuals to receive similar predictions: $d_Y(f(x), f(x')) \leq L \cdot d_X(x, x')$.

Integrated Gradients A feature attribution method satisfying completeness by integrating gradients along a path from baseline to input.

Jailbreak An adversarial prompt designed to circumvent safety guardrails and elicit prohibited outputs from language models.

KL Divergence Kullback-Leibler divergence; a measure of difference between probability distributions: $D_{KL}(P\|Q) = \mathbb{E}_P[\log(P/Q)]$.

Laplace Mechanism A differential privacy mechanism adding Laplace noise scaled to query sensitivity: $M(x) = f(x) + \text{Lap}(\Delta f/\epsilon)$.

Lipschitz Continuity A function f is L-Lipschitz if $|f(x) - f(x')| \leq L\|x - x'\|$ for all x, x'.

LIME Local Interpretable Model-agnostic Explanations; explains predictions by fitting a local linear model.

Model Card Standardized documentation describing a model's capabilities, limitations, and performance across subgroups.

OOD Detection Out-of-Distribution detection; identifying inputs that differ significantly from the training distribution.

PAC Learning Probably Approximately Correct learning; a theoretical framework for computational learning theory.

PGD Projected Gradient Descent; an iterative adversarial attack projecting perturbations onto an ϵ-ball.

Privacy Budget The cumulative privacy loss parameter ϵ consumed by multiple differentially private queries.

Rademacher Complexity A measure of a hypothesis class's ability to fit random noise, used in generalization bounds.

Randomized Smoothing A technique for certified robustness by classifying based on majority vote under Gaussian noise.

RLHF Reinforcement Learning from Human Feedback; training language models to align with human preferences.

Robustness The property that a model's predictions remain correct under input perturbations or distribution shift.

Sensitivity In differential privacy, the maximum change in a query's output when a single data point changes.

SHAP Values SHapley Additive exPlanations; feature attributions based on Shapley values from cooperative game theory.

Structural Causal Model (SCM)
A formal model specifying causal relationships via structural equations and a directed acyclic graph.

VC Dimension
Vapnik-Chervonenkis dimension; the maximum number of points that can be shattered by a hypothesis class.

Watermarking Embedding detectable patterns in AI-generated content to enable attribution and detection.

Excess Risk Decomposition
Breakdown of generalization error into approximation error (bias from hypothesis class) and estimation error (variance from finite samples).

Approximation Error
Error due to the hypothesis class not containing the true function.

Estimation Error
Error due to estimating parameters from finite data.

Bias-Variance Decomposition
Decomposition of expected error into irreducible noise, bias (systematic error), and variance (sensitivity to data).

Irreducible Error
Inherent noise in the data-generating process that no model can eliminate.

Uniform Convergence
Property that empirical risk converges to true risk uniformly over the hypothesis class.

Sauer's Lemma
Bounds the growth function of a hypothesis class in terms of VC dimension.

Double Descent Phenomenon
Test error decreases again after interpolation threshold in over-parameterized regimes.

Under-parameterized Regime
Model capacity insufficient to fit training data, leading to high bias.

Critical Regime
Capacity just enough for interpolation, causing high variance.

Over-parameterized Regime
Capacity exceeds data needs, allowing implicit regularization.

PAC Learning Framework where an algorithm, given enough samples, learns a hypothesis whose error is at most ε with probability at least $1 - \delta$.

Agnostic PAC Learning
PAC learning without assuming the true function is in the hypothesis class.

Margin-Based Generalization Bounds
Bounds using classifier margins for better generalization analysis.

Algorithmic Stability
Property that small data changes lead to small output changes, enabling generalization.

Linearity Hypothesis
Adversarial vulnerability arises from local linearity in high-dimensional spaces.

Concentration of Measure
Probability mass concentrates near low-dimensional structures in high dimensions.

Isoperimetric Inequality
Bounds on set expansion in high dimensions.

White-box Attacks
Attacks with full model access (weights, gradients).

Black-box Attacks
Attacks with only input-output access.

Gray-box Attacks
Attacks with partial model knowledge.

Targeted Attacks
Misclassify to a specific wrong class.

Untargeted Attacks
Misclassify to any wrong class.

L_∞ **Constraint** Perturbations bounded by maximum per-dimension change.

L_2 **Constraint** Perturbations bounded by Euclidean norm.

L_0 **Constraint** Perturbations bounded by number of modified dimensions.

Physical Attacks
Perturbations realizable in the physical world (e.g., printed artifacts on medical scans).

First-Order Adversaries
Attacks using gradient information.

Zeroth-Order Gradient Estimation
Estimating gradients via function queries.

Random Direction Gradient Estimation
Efficient gradient estimation using random projections.

NES (Natural Evolution Strategies) Attack
Black-box attack using evolutionary optimization.

Square Attack Query-efficient black-box attack using random square perturbations.

Adversarial Training
Training on adversarial examples to improve robustness.

Robust Optimization
Min-max optimization for worst-case perturbations.

Zero-Sum Game
Formulation of adversarial training as a game between trainer and attacker.

Minimax Theorem
Order of min-max does not matter under convexity.

Danskin's Theorem
Gradient of max function equals gradient at optimum.

Certified Robustness
Mathematical guarantee of invariance to perturbations.

Randomized Smoothing
Converts classifiers to certifiably robust via noise addition.

Smoothed Classifier
Classifier based on majority vote under Gaussian noise.

Natural Distribution Shifts
Legitimate changes in data distribution (e.g., covariate shift).

Domain Adaptation
Adapting models to new domains without labels.

Importance Weighting
Reweighting samples to correct for distribution shift.

Domain Adversarial Training (DANN)
Adversarial training to learn domain-invariant features.

Gradient Reversal Layer (GRL)
Layer that negates gradients for adversarial training.

Invariant Risk Minimization (IRM)
Learning features invariant across environments.

Distributionally Robust Optimization (DRO)
Optimizing for worst-case distribution in an uncertainty set.

f-divergence ball
Uncertainty set defined by f-divergence from empirical distribution.

Wasserstein ball
Uncertainty set defined by Wasserstein distance.

Group DRO Uncertainty set as convex hull of group distributions.

Philosophical Foundations
Ethical theories underlying fairness (e.g., Rawlsian justice).

Rawlsian Justice
Inequalities should benefit the least advantaged.

Luck Egalitarianism
Distinguishes circumstantial vs. chosen inequalities.

Anti-subordination Theories
Focus on reducing group-based disadvantages.

Group Fairness
Fairness across demographic groups.

Individual Fairness
Similar individuals receive similar predictions.

Impossibility Results
Theorems showing incompatibility of fairness definitions.

Intersectionality
Fairness across multiple intersecting identities.

Multi-Group Fairness
Fairness in settings with many groups.

Disparity Metrics
Measures of fairness violations (e.g., DPD, DIR).

Calibration Metrics
Measures of probability accuracy across groups.

Pre-processing Methods
Fairness interventions before training.

Reweighing Adjusting sample weights for fairness.

Fair Representation Learning
Learning unbiased latent representations.

In-processing Methods
Fairness constraints during training.

Adversarial Debiasing
Adversarial training to remove bias.

Lagrangian Relaxation
Converting constraints to penalties.

Post-processing Methods
Adjusting predictions after training.

Threshold Optimization
Group-specific decision thresholds.

Calibration Adjustment
Correcting probability estimates.

Causal Fairness
Fairness defined via causal graphs.

Counterfactual Fairness
Predictions unchanged under sensitive attribute interventions.

Path-Specific Fairness
Fairness along specific causal pathways.

Proxy Variable
Feature correlated with sensitive attribute, enabling indirect discrimination.

Kleinberg-Chouldechova Impossibility Theorem
Incompatibility of calibration and separation under different base rates.

Mechanisms for Fair Learning
Techniques for achieving fairness in practice.

Differential Privacy (DP)
Framework quantifying privacy loss via output indistinguishability.

Privacy Loss Random Variable
Random variable measuring per-sample privacy cost.

Laplace Mechanism
Adding Laplace noise for DP.

Gaussian Mechanism
Adding Gaussian noise for DP.

Exponential Mechanism
Sampling from utility-scored outputs.

Composition Combining multiple DP mechanisms.

Renyi Differential Privacy (RDP)
Composition-friendly DP variant.

DP-SGD Differentially private stochastic gradient descent.

Privacy Amplification by Subsampling
Privacy increases with subsampling.

Moments Accountant
Tight composition bounds via moment tracking.

PATE Private aggregation via teacher-student framework.

Secure Multi-Party Computation (MPC)
Joint computation without revealing inputs.

Secret Sharing
Splitting secrets for MPC.

Shamir's Secret Sharing
Threshold secret sharing scheme.

Additive Secret Sharing
Simple sharing for linear operations.

Garbled Circuits
Secure two-party computation via encrypted circuits.

Homomorphic Encryption (HE)
Computation on encrypted data.

Local Differential Privacy
Privacy where users randomize their own data.

Randomized Response
Mechanism for local DP on binary data.

Reconstruction Attacks
Reconstructing data from DP outputs.

Dinur-Nissim Theorem
Limits on answering many queries privately.

Membership Inference Attacks (MIA)
Detecting if a sample was in training data.

Model Inversion Attacks
Reconstructing inputs from model outputs.

Attribute Inference Attacks
Inferring sensitive attributes from predictions.

Explainable AI (XAI)
Methods for making AI decisions understandable.

Interpretability
Inherent understandability of a model.

Intrinsic vs. Post-hoc
Interpretable by design vs. analyzed after training.

Global vs. Local
Explanations for the whole model vs. single predictions.

Model-Agnostic vs. Model-Specific
Works on any model vs. exploits internals.

Feature Attribution Methods
Assigning importance scores to features.

LIME Local explanations via interpretable approximations.

Gradient-Based Methods
Using gradients for attribution.

Saliency Maps Visualizing gradient magnitudes.

Integrated Gradients (IG)
Path-integral attribution method.

Concept-Based Explanations
Explanations in terms of human concepts.

Concept Bottleneck Models (CBM)
Models predicting via interpretable concepts.

TCAV Testing sensitivity to high-level concepts.

Evaluating Explanations
Metrics for explanation quality.

Fidelity How well explanation matches model behavior.

Robustness Stability of explanations to perturbations.

Sanity Checks Verifying explanations depend on model parameters.

Algorithmic Recourse
Methods for users to change predictions.

Counterfactual Explanations
"What if" scenarios for different outcomes.

Wachter's Method
Optimizing for minimal counterfactual changes.

Actionability Constraints
Restrictions on changeable features.

DICE Generating diverse counterfactuals.

Causal Counterfactuals
Counterfactuals respecting causal structure.

Recourse Robustness
Counterfactuals valid under model changes.

Algorithmic Recourse via Integer Programming
MIP for tree-based recourse.

Ladder of Causation
Levels of causal reasoning (association, intervention, counterfactuals).

Association Observing correlations.

Intervention Acting to change variables.

Counterfactuals
Reasoning about hypothetical worlds.

d-Separation Graphical criterion for conditional independence.

Do-Operator Notation for interventions.

Back-Door Criterion
Conditions for identifying causal effects.

Adjustment Formula
Formula for causal effect estimation.

Do-Calculus Rules for transforming causal queries.

Federated Learning (FL)
Distributed training without data sharing.

FedAvg Standard FL algorithm via model averaging.

Statistical Heterogeneity
Non-IID data across clients.

FedProx FL with proximal regularization for heterogeneity.

Personalized Federated Learning
Client-specific model tailoring.

Local Fine-Tuning
Global model fine-tuned per client.

Personalized Layers
Shared base with client-specific heads.

MAML-based Personalization
Meta-learning for personalization.

Mixture of Experts
Client-specific expert combinations.

Gradient Compression
Reducing communication via quantization/sparsification.

Quantization Reducing gradient precision.

Sparsification Transmitting only top gradients.

Low-Rank Approximation
Compressing gradients via SVD.

Secure Aggregation
MPC for aggregating without revealing updates.

Byzantine Attacks
Malicious clients poisoning FL.

Krum Robust aggregation via nearest neighbors.

Coordinate-wise Median
Median per gradient coordinate.

Trimmed Mean
Averaging after removing outliers.

Alignment Problem
Ensuring RL agents pursue intended goals.

Reward Exploitation
Exploiting reward function loopholes.

Specification Gaming
Achieving reward via unintended means.

Coast Runners Effect
Optimizing for speed by not finishing the race.

Negative Side Effects
Unintended consequences of actions.

Reward Tampering
Modifying the reward signal.

Constrained Markov Decision Processes (CMDP)
MDPs with safety constraints.

Lagrangian Relaxation
Penalty method for constraints.

Constrained Policy Optimization (CPO)
Trust-region method for CMDPs.

Safe Exploration
Exploration without violating safety.

Safety Layers Real-time safety enforcement.

Control Barrier Functions (CBF)
Functions ensuring forward invariance.

Lyapunov-Based Safe Learning
Stability via Lyapunov functions.

Shielding Verified backup policies for safety.

Value Alignment
Learning aligned reward functions.

Inverse Reinforcement Learning (IRL)
Inferring rewards from behavior.

Maximum Entropy IRL
Probabilistic IRL variant.

Feature Matching IRL
Matching feature expectations.

Preference Learning
Learning from human preferences.

Bradley-Terry Model
Probabilistic preference model.

Reinforcement Learning from Human Feedback (RLHF)
Aligning via human feedback.

Direct Preference Optimization (DPO)
Preference-based policy optimization.

Reward Shaping
Adding auxiliary rewards.

Iterated Amplification
Recursively improving alignment.

AI Safety via Debate
Using debates for better decisions.

Out-of-Distribution (OOD) Detection
Identifying inputs outside training distribution.

Uncertainty Quantification
Measuring model confidence.

Distribution Shift
Changes between train and test distributions.

Covariate Shift
Input distribution changes.

Label Shift Class proportion changes.

Concept Drift Relationship between inputs/outputs changes.

Domain Shift Simultaneous marginal and conditional changes.

Softmax-Based Methods
OOD detection via softmax probabilities.

Maximum Softmax Probability (MSP)
Confidence via max softmax.

Temperature Scaling
Adjusting softmax for better calibration.

ODIN OOD detection with input preprocessing.

Energy-Based OOD Detection
Using free energy for detection.

Deep Ensembles
Uncertainty via model ensembles.

Mahalanobis Distance-Based Detection
Distance to class centroids.

Bayesian Neural Networks
Uncertainty via posterior distributions.

Variational Inference
Approximating posteriors.

Monte Carlo Dropout
Dropout for uncertainty estimation.

Laplace Approximation
Gaussian posterior approximation.

Aleatoric Uncertainty
Data-inherent noise.

Epistemic Uncertainty
Model uncertainty.

Calibration Probabilities match frequencies.

Expected Calibration Error (ECE)
Average calibration gap.

Reliability Diagrams
Plotting accuracy vs. confidence.

Post-hoc Calibration Methods
Adjusting probabilities after training.

Platt Scaling Logistic calibration.

Isotonic Regression
Non-parametric calibration.

Conformal Prediction
Distribution-free prediction sets.

Selective Prediction (Rejection)
Abstaining on uncertain inputs.

Risk-Coverage Trade-off
Balancing accuracy and coverage.

Agentic System
An automated AI entity that perceives, reasons, and acts over extended horizons, defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \pi, P_{\text{env}})$ of state, action, observation spaces, tools, policy, and environment dynamics.

Tool-Augmented Agent
An agent with access to external tools (APIs, databases, code execution) that extend its capabilities beyond text generation.

Agent Trajectory
A sequence of observations and actions over an agent's execution horizon.

Markov Game (Stochastic Game)
Multi-agent extension of MDPs where each agent has its own reward function and the transition depends on joint actions.

Nash Equilibrium
A joint policy where no agent can unilaterally improve its expected return.

Price of Anarchy
Ratio of optimal social welfare to worst Nash equilibrium welfare; measures efficiency loss from strategic behavior.

Compositional Trust
The property that trust/safety guarantees compose across sequential components in an agentic pipeline.

Error Propagation
The phenomenon where per-step errors compound across multi-step agent trajectories.

Action Shield A safety wrapper that intercepts and modifies proposed agent actions to ensure constraint satisfaction.

Capability Control
Restricting an agent's effective action space to a verified safe subset.

Tool Precondition/Postcondition
Formal specifications of conditions that must hold before and after tool invocation.

Goal Misgeneralization
An agent achieving high training performance by optimizing a proxy goal that diverges from the intended goal in deployment.

Instrumental Convergence
The tendency of optimal agents to pursue sub-goals (self-preservation, resource acquisition) regardless of terminal goal.

Power-Seeking Tendency of optimal policies to navigate toward states with greater optionality (more reachable future states), formalized by Turner et al. (2021) via MDP graph structure conditions.

Corrigibility The property that an agent does not resist correction, modification, or shutdown by its operators.

AI Delegation Game
A principal-agent model where a human (principal) delegates tasks to an AI agent.

Moral Hazard (AI)
Information asymmetry where the principal cannot observe the agent's reasoning or intermediate actions.

Scalable Oversight
Oversight mechanisms whose cost grows sublinearly in the complexity of the delegated task.

AI Safety via Debate
A protocol where two adversarial agents argue before a human judge, enabling efficient oversight of complex tasks.

Indirect Prompt Injection
Adversarial instructions embedded in data retrieved by an agent during execution, rather than in direct user input.

Instruction Hierarchy
A priority ordering of instruction sources (system $\succ$ user $\succ$ tool output $\succ$ retrieved content).

Multi-Agent Fairness
Fairness constraints ensuring that routing and per-agent performance do not introduce disparities across protected groups.

Task Completion Rate
Fraction of tasks successfully completed by an agentic system.

Safety Violation Rate
Fraction of agent trajectories containing at least one safety-violating action.

Trajectory-Level Trust Score
: Composite metric integrating correctness, safety, and efficiency of an agent trajectory.

Machine Unlearning
: The process of algorithmically removing the influence of specific training data points from a trained model, often to comply with data deletion requests.

Influence Functions
: A technique from robust statistics used to approximate the effect of removing a training point on a model's parameters without fully retraining it.

20 List of Notation

Symbol	Description
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output/label space
$\mathcal{D}$	Data distribution over $\mathcal{X} \times \mathcal{Y}$
$S = \{(x_i, y_i)\}_{i=1}^n$	Training dataset of n samples
$f_\theta : \mathcal{X} \to \mathcal{Y}$	Model parameterized by θ
$\ell(f(x), y)$	Loss function measuring prediction error
$\mathcal{R}(f)$	True/expected risk: $\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)]$
$\hat{\mathcal{R}}(f)$	Empirical risk: $\frac{1}{n}\sum_{i=1}^n \ell(f(x_i), y_i)$
$\mathcal{H}$	Hypothesis class
d_{VC}	VC dimension
$\mathcal{R}_n(\mathcal{H})$	Rademacher complexity
ϵ	Perturbation budget (adversarial) or privacy parameter (DP)
δ	Probability of failure (in PAC learning or DP)
∇_x	Gradient with respect to x
$\|\cdot\|_p$	L_p norm
$B_\epsilon(x)$	ϵ-ball around x: $\{x' : \|x' - x\| \leq \epsilon\}$
Π_C	Projection onto set C
A	Sensitive/protected attribute
$\hat{Y}$	Predicted label
TPR	True positive rate: $P(\hat{Y} = 1 \mid Y = 1)$
FPR	False positive rate: $P(\hat{Y} = 1 \mid Y = 0)$
DPD	Demographic parity difference
EOD	Equalized odds difference
$\mathcal{M}$	Randomized mechanism (DP)
Δf	Sensitivity of function f
$\text{Lap}(b)$	Laplace distribution with scale b
$D_\alpha(P\|Q)$	Rényi divergence of order α
$G = (V, E)$	Causal graph with vertices V and edges E
$do(X = x)$	Intervention setting X to value x
Y_x	Counterfactual: value of Y under intervention $do(X = x)$
$Pa(X)$	Parent nodes of X in causal graph
K	Number of clients (federated learning)
E	Number of local epochs
η	Learning rate
σ	Standard deviation of Gaussian noise
C	Gradient clipping threshold
$\mathbb{E}[\cdot]$	Expectation
$\text{Var}[\cdot]$	Variance
$\text{Cov}[\cdot, \cdot]$	Covariance
$\mathcal{N}(\mu, \sigma^2)$	Gaussian distribution
$\text{Dir}(\alpha)$	Dirichlet distribution
$D_{KL}(P\|Q)$	KL divergence from Q to P
$H(X)$	Entropy of X

Symbol	Description
$I(X;Y)$	Mutual information between X and Y
α	Class K function parameter (CBF); significance level (Conformal); interpolation parameter (IG); inner loop learning rate (MAML); FPR (Fairness); order for Renyi divergence (RDP)
β	TPR (Fairness); trade-off parameter (TRADES); KL coefficient (RLHF/DPO); regularization strength (IRM)
γ	Discount factor (CMDP); interpolation parameter (IG)
δ	Trust region size (CPO); failure probability (DP); perturbation (Adversarial); dual variable (CPO)
η	Perturbation (Linearity Hypothesis); learning rate (FedAvg)
θ	Model parameters; policy parameters (RL)
κ	Confidence parameter (C&W); confidence function (Selective Prediction)
λ	Lagrange multiplier; regularization parameter; moment order (Moments Accountant); hyperparameters (DICE)
μ	Proximal term (FedProx); step size (Zeroth-Order); class mean (Mahalanobis)
ν	Dual variable (CPO)
ξ	Random binary vector (Quantization); LIME objective
π	Policy (RL); mixing coefficients (Mixture of Experts); proximity kernel (LIME)
ρ	Radius (DRO)
σ	Noise scale (Gaussian Mechanism); standard deviation (Gaussian noise); shield (Shielding)
τ	Threshold (Selective Prediction); temperature (Temperature Scaling)
ϕ	Coverage (Selective Prediction); shared layers (Personalized Layers); Shapley value
Φ	Potential function (Reward Shaping); inverse CDF (Randomized Smoothing)
χ	Chi-squared divergence (DRO)
ψ	Personal layers (Personalized Layers)
$\mathcal{A}$	Adversary (Security); actions (MDP)
$\mathcal{B}$	Uncertainty set (Robust Recourse); bins (ECE)
$\mathcal{C}$	Safe set (CBF); prediction set (Conformal Prediction)
$\mathcal{E}$	Training environments (IRM)
$\mathcal{F}$	Structural equations (SCM)
$\mathcal{G}$	Graph (manipulated, e.g., $G_{\bar{X}}$)
$\mathcal{I}$	Immutable/increasing features (Actionability)
$\mathcal{M}$	Randomized mechanism (DP); SCM
$\mathcal{P}$	Probability distribution; protocol (MPC)
$\mathcal{R}$	Risk; output space (DP)
$\mathcal{S}$	States (MDP); perturbation set; unsafe states
$\mathcal{U}$	Uncertainty set (DRO); unsafe states
$\mathcal{Z}$	Latent space; exogenous variables (SCM)
$\mathbf{w}$	Model weights (Bayesian NN)
$\mathbf{z}$	Dropout masks (Monte Carlo Dropout)

Symbol	Description
$W_p(Q, P)$	Wasserstein distance
$D_\alpha(P\|Q)$	Renyi divergence
$D_f(Q\|P)$	f-divergence
$\text{sgn}(\cdot)$	Sign function
$\odot$	Element-wise multiplication
$\text{diag}(\cdot)$	Diagonal matrix
Top_k	Top-k sparsification
Quantile_p	p-quantile
$\text{acc}(B_b)$	Accuracy in bin b
$\text{conf}(B_b)$	Confidence in bin b
yloss	Loss for target class
dpp_diversity	Diversity metric
$f_x(S)$	Prediction with feature subset S
S_{MSP}	Maximum softmax probability score
$S_{C,k}$	Conceptual sensitivity
J_R/J_C	Expected return/cost
L_{π_k}	Surrogate objective
A^{π_k}	Advantage function
$\bar{D}_{KL}$	Average KL divergence
$\hat{q}$	Conformal quantile
$\hat{y}$	Predicted label (ensemble or MC dropout)
$R(\tau)$	Selective risk
Z	Privacy loss random variable
$\Delta_1 f/\Delta_2 f$	L1/L2 sensitivity
$\alpha_{\mathcal{M}}(\lambda)$	Moment function
n_c	Vote count (PATE)
k_w^v	Key for wire value
Enc	Encryption function
$G_{\bar{X}}$	Graph with arrows into X removed
$G_{\underline{X}}$	Graph with arrows out of X removed
$G_{\bar{X}\underline{Z}}$	Graph with arrows into X and out of Z removed
$G_{\bar{X}\bar{Z}(W)}$	Graph with arrows into X and into Z(W) removed
V_i	Endogenous variable i
f_i	Structural equation for variable i
Pa_i	Parents of variable i
U_i	Exogenous variable i
$x'_{\text{downstream}}$	Downstream features under intervention
$x'_{\text{actionable}}$	Actionable features
L	Leaves (Decision Trees)
$\mathcal{P}_\ell$	Paths for leaf ℓ
z_ℓ	Binary variable for leaf ℓ
L^+	Positive leaves
$u_{j,\ell}/l_{j,\ell}$	Bounds for feature j in leaf ℓ
p_k	Client weight (FL)
F_k	Local loss for client k
S_t	Selected clients at round t
σ_k	Variance for client k
σ_g	Heterogeneity bound
ϕ	Shared layers

Symbol	Description
ψ_k	Personal layers for client k
$Q(g)$	Quantized gradient
U/V	Low-rank matrices
r	Rank
x^*/y^*	Reconstructed input/label
h	Barrier function
$\mathcal{L}_{RM}$	Reward model loss
$\mathcal{L}_{DPO}$	DPO loss
π_{ref}	Reference policy
F	Shaping function
P_{train}/P_{test}	Training/test distributions
$\tilde{x}$	Perturbed input
$E(x)$	Energy function
q_ϕ	Variational distribution
ECE	Expected calibration error
T^*	Optimal temperature
s	Conformity score
$C(x)$	Prediction set
$\kappa(x)$	Confidence function
Agentic AI	
$\mathcal{T} = \{t_1, \ldots, t_m\}$	Tool set available to the agent
$t_i : \mathcal{D}_i \to \mathcal{R}_i$	Tool i mapping domain $\mathcal{D}_i$ to range $\mathcal{R}_i$
$\mathcal{A}_{\text{text}}$	Natural language action space
$\mathcal{A}_{t_i}$	Tool invocation action space for tool t_i
P_{env}	Environment transition-observation dynamics
τ	Agent trajectory $(o_0, a_0, o_1, \ldots, o_T)$
$P_\pi(\tau)$	Trajectory distribution under policy π
N	Number of agents (Markov games)
$V_i^{\boldsymbol{\pi}}(s)$	Value function of agent i under joint policy $\boldsymbol{\pi}$
$W(\boldsymbol{\pi})$	Social welfare of joint policy $\boldsymbol{\pi}$
PoA	Price of anarchy
NE	Set of Nash equilibria
δ_k	Per-step failure probability at step k
$\mathcal{A}_{\text{safe}}$	Safe action subset
σ	Action shield function
$\text{Pre}_i/\text{Post}_i$	Tool precondition/postcondition
$\mathcal{A}_{\text{corr}}$	Correction actions (e.g., shutdown)
U_H	Human principal's utility function
TCR	Task completion rate
SVR	Safety violation rate
$\text{Trust}(\tau)$	Trajectory-level trust score

21 Acknowledgments

Below, I gratefully acknowledge the open learning community online and the instructors behind these excellent video courses. Their willingness to share high-quality educational resources openly has made this material possible.

- Stanford CS230: Deep Learning I (Autumn 2025)
- Theoretical Aspects of Trustworthy AI
- Trustworthy ML Initiative Seminar Series
- Stanford CS229: Machine Learning I (Spring 2022)
- Causal Inference
- Google DeepMind AGI Safety Course
- Alignment Workshop
- Stanford CS329H: Machine Learning from Human Preferences (Autumn 2024)
- Stanford EE364A: Convex Optimization
- Google DeepMind AGI Safety Course
- Trustworthy AI Course by Birhanu Eshete
- Trustworthy Machine Learning - Winter Semester 2024/2025
- Agentic AI MOOC Fall 2025
- 2025 Agentic AI Summit
- Stanford CME295: Agentic AI (Autumn 2025)
- LLM Agents MOOC Fall 2024

References

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

[4] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

[5] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudík, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International conference on machine learning*, pages 60–69. PMLR, 2018.

[6] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[7] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European conference on computer vision*, pages 484–501. Springer, 2020.

[8] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[9] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[10] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of machine learning research*, 3(Nov):463–482, 2002.

[11] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4. 2023. URL https://arxiv.org/abs/2303.12712.

[12] Igor Calzada, Géza Németh, and Mohammed Salah Al-Radhi. Trustworthy ai for whom? genai detection techniques of trust through decentralized web3 ecosystems. *Big Data and Cognitive Computing*, 9(3):62, 2025.

[13] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. Ieee, 2017.

[14] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 407–425. IEEE, 2024.

[15] Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*, 2023.

[16] Arunraju Chinnaraju. Explainable ai (xai) for trustworthy and transparent decision-making: A theoretical framework for ai interpretability. *World Journal of Advanced Engineering Technology and Sciences*, 14(3):170–207, 2025.

[17] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[18] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *international conference on machine learning*, pages 1310–1320. PMLR, 2019.

[19] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020.

[20] Ameet Deshpande, Vishvak Murahari, Tanmay Rajpurohit, Ashwin Kalyan, and Karthik Narasimhan. Toxicity in chatgpt: Analyzing persona-assigned language models. In *Findings of the association for computational linguistics: EMNLP 2023*, pages 1236–1270, 2023.

[21] Minxin Du, Xiang Yue, Sherman SM Chow, Tianhao Wang, Chenyu Huang, and Huan Sun. Dp-forward: Fine-tuning and inference on language models with differential privacy in forward pass. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 2665–2679, 2023.

[22] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[23] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

[24] Tao Fan, Yan Kang, Guoqiang Ma, Weijing Chen, Wenbin Wei, Lixin Fan, and Qiang Yang. Fate-llm: A industrial grade federated learning framework for large language models. *arXiv preprint arXiv:2310.10049*, 2023.

[25] Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. Bias and fairness in large language models: A survey. *Computational linguistics*, 50(3):1097–1179, 2024.

[26] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[27] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[28] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of opportunity in supervised learning. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

[29] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[30] Dan Hendrycks, Mantas Mazeika, and Thomas Woodside. An overview of catastrophic ai risks. *arXiv preprint arXiv:2306.12001*, 2023.

[31] Andrés Herrera-Poyatos, Javier Del Ser, Marcos López de Prado, Fei-Yue Wang, Enrique Herrera-Viedma, and Francisco Herrera. Responsible artificial intelligence systems: A roadmap to society's trust through trustworthy ai, auditability, accountability, and governance. *arXiv preprint arXiv:2503.04739*, 2025.

[32] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

[33] Zhijing Jin, Jiarui Liu, Zhiheng Lyu, Spencer Poff, Mrinmaya Sachan, Rada Mihalcea, Mona Diab, and Bernhard Schölkopf. Can large language models infer causation from correlation? *arXiv preprint arXiv:2306.05836*, 2023.

[34] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.

[35] Emre Kiciman, Robert Ness, Amit Sharma, and Chenhao Tan. Causal reasoning and large language models: Opening a new frontier for causality. *Transactions on Machine Learning Research*, 2023.

[36] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR, 2023.

[37] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. In *Innovations in Theoretical Computer Science*, 2017.

[38] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. *Advances in neural information processing systems*, 30, 2017.

[39] Haoran Li, Yulin Chen, Jinglong Luo, Jiecong Wang, Hao Peng, Yan Kang, Xiaojin Zhang, Qi Hu, Chunkit Chan, Zenglin Xu, et al. Privacy in large language models: Attacks, defenses and future directions. *arXiv preprint arXiv:2310.10383*, 2023.

[40] Yijing Li, Xiaofeng Tao, Xuefei Zhang, Junjie Liu, and Jin Xu. Privacy-preserved federated learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):8423–8434, 2021.

[41] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[42] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020.

[43] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[44] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[45] Potsawee Manakul, Adian Liusie, and Mark Gales. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pages 9004–9017, 2023.

[46] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

[47] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium*, pages 263–275. IEEE, 2017.

[48] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 220–229, 2019.

[49] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT Press, 2 edition, 2018.

[50] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 607–617, 2020.

[51] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A Feder Cooper, Daphne Ippolito, Christopher A Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models. *arXiv preprint arXiv:2311.17035*, 2023.

[52] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[53] Judea Pearl. *Causality*. Cambridge university press, 2009.

[54] Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448, 2022.

[55] Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 78(5):947–1012, 2016.

[56] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[57] Rajesh Ranjan, Shailja Gupta, and Surya Narayan Singh. Loka protocol: A decentralized framework for trustworthy and ethical ai agent ecosystems. *arXiv preprint arXiv:2504.10915*, 2025.

[58] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. Cryptflow2: Practical 2-party secure inference. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 325–342, 2020.

[59] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[60] Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. Whose opinions do language models reflect? In *International conference on machine learning*, pages 29971–30004. PMLR, 2023.

[61] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

[62] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[63] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1671–1685, 2024.

[64] Eugenia Stamboliev and Tim Christiaens. How empty is trustworthy ai? a discourse analysis of the ethics guidelines of trustworthy ai. *Critical Policy Studies*, 19(1):39–56, 2025.

[65] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[66] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[67] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[68] Alexander Matt Turner, Logan Smith, Rohin Shah, Andrew Critch, and Prasad Tadepalli. Optimal policies tend to seek power. In *Advances in Neural Information Processing Systems*, volume 34, pages 23063–23074, 2021.

[69] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[70] Vladimir N Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. In *Measures of complexity: festschrift for alexey chervonenkis*, pages 11–30. Springer, 2015.

[71] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[72] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, 2024.

[73] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. Aligning large language models with human: A survey. *arXiv preprint arXiv:2307.12966*, 2023.

[74] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in neural information processing systems*, 36:80079–80110, 2023.

[75] Wenqi Wei and Ling Liu. Trustworthy distributed ai systems: Robustness, privacy, and governance. *ACM Computing Surveys*, 57(6):1–42, 2025.

[76] Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, et al. The rise and potential of large language model based agents: A survey. *Science China Information Sciences*, 68(2):121101, 2025.

[77] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

[78] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[79] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019.

[80] Haiyan Zhao, Hanjie Chen, Fan Yang, Ninghao Liu, Huiqi Deng, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, and Mengnan Du. Explainability for large language models: A survey. *ACM Transactions on Intelligent Systems and Technology*, 15(2):1–38, 2024.

[81] Haodong Zhao, Wei Du, Fangqi Li, Peixuan Li, and Gongshen Liu. Fedprompt: Communication-efficient and privacy-preserving prompt tuning in federated learning. In *ICASSP 2023-2023 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 1–5. IEEE, 2023.

[82] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

[83] Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. *Advances in neural information processing systems*, 32, 2019.

[84] Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.