Bridging AI and Optimization: Methods for Hard Constraint Satisfaction

The success of deep learning is predicated on its ability to learn complex, non-linear functions from data. However, for safety-critical and physically-grounded systems, predictive accuracy is insufficient. We require ironclad guarantees that model outputs adhere to hard, non-negotiable constraints.

Deep Learning:
High Expressivity, No Guarantees

The Goal:
Principled Integration

Classical Optimization:
Formal Guarantees, Limited Expressivity

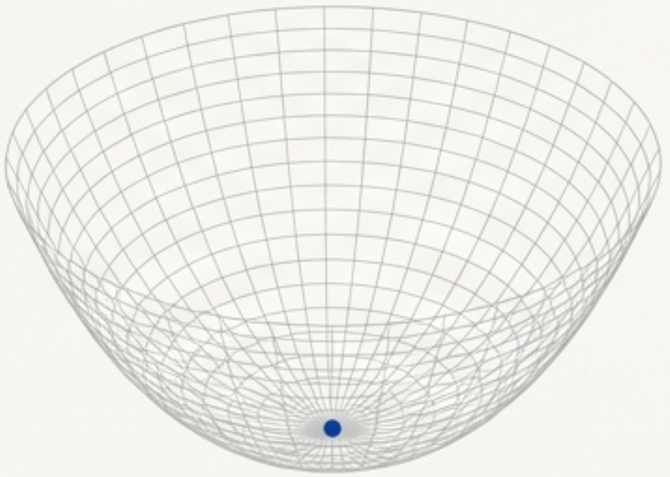
The foundational principles of classical optimization, far from being obsolete, provide the most promising pathways to imbue AI models with formal guarantees. This tutorial explores two such pathways, rooted in the concepts of duality and projection.

The Foundation: The Standard Form of a Convex Program

Recall the canonical formulation of a convex optimization problem from Boyd and Vandenberghe (Chapter 4):

$$\begin{aligned} \text{minimize} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq 0, \quad i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0, \quad i = 1, \ldots, p \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the optimization variable, the functions $f_0, \ldots, f_m$ are convex, and $h_1, \ldots, h_p$ are affine.



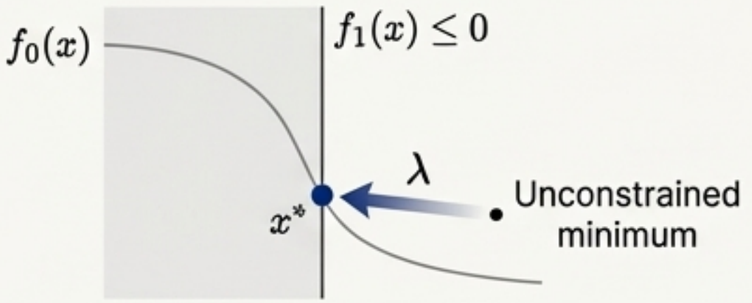
Key Insight from B&V: "The great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity." (B&V, page 17)

This distinction is central. When a problem is convex, we have access to powerful theory and reliable, efficient algorithms that guarantee finding a global optimum. This is the world of guarantees we seek to recover for AI.

Two Pillars of Classical Guarantees

Adaptation via Duality

Concept: Lagrangian Duality (B&V, Chapter 5)



Mechanism: The Lagrangian
The Lagrangian $L(x, \lambda, \nu) = f_0(x) + \sum \lambda_i f_i(x) + \sum \nu_i h_i(x)$ transforms a constrained problem into an unconstrained one by pricing constraints using dual variables $\lambda \geq 0$.

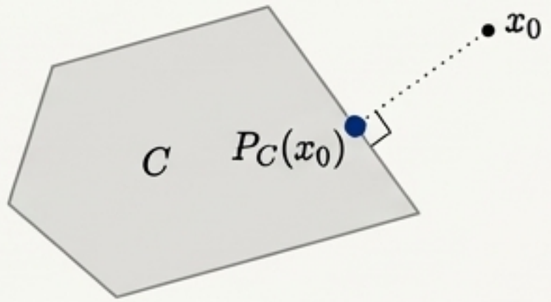
Intuition: Find an equilibrium price λ^* for violating constraints. At the optimum, the gradient vanishes:

$$\nabla f_0(x^*) + \sum \lambda_i^* \nabla f_i(x^*) + \sum \nu_i^* \nabla h_i(x^*) = 0$$

AI Analogy: Treat constraint violations as a penalty in the model's loss function.

Adaptation via Projection

Concept: Projection onto a Convex Set (B&V, Chapter 8)



Mechanism: The projection
The projection $P_C(x_0) = \operatorname{argmin}_{x \in C} \|x - x_0\|_2$ finds the point x in the feasible set C that is closest to a given point x_0.

Intuition: If a point is outside the feasible set, find the "closest" point that is inside.

AI Analogy: If a model's output is infeasible, project it onto the feasible set to "repair" it.

The AI Challenge: Constraints on Model Predictions

The Problem Setup

We have a machine learning model, typically a neural network, parameterized by weights θ. The model $y = M(z; \theta)$ maps an input z to an output y.

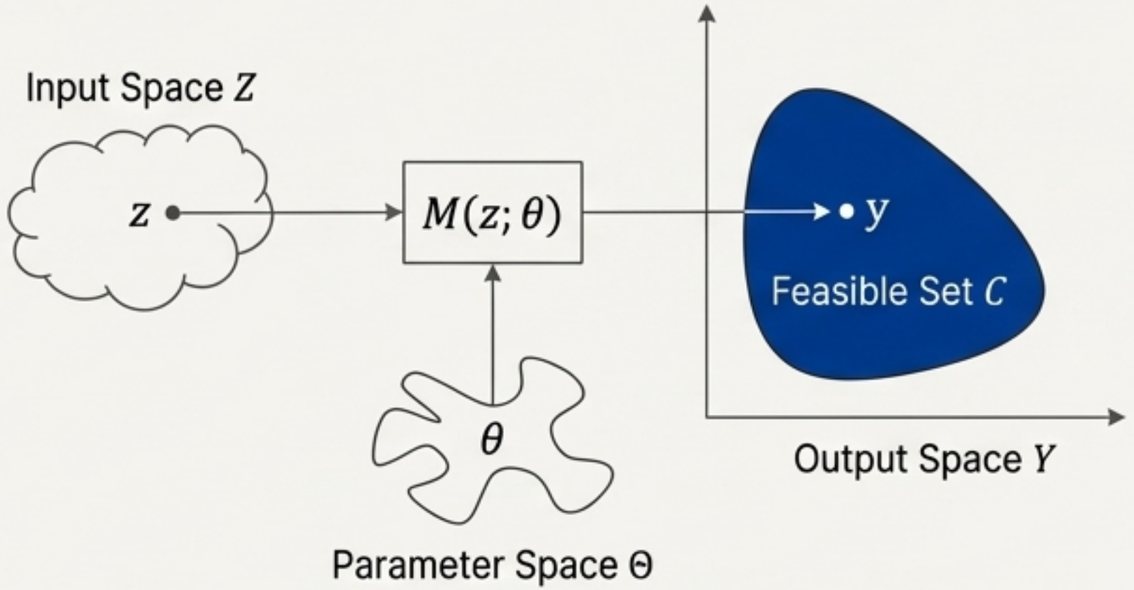
The Goal

We want to find parameters θ that minimize a task-specific loss $L_{\text{task}}(M(z; \theta), y_{\text{true}})$ while ensuring the model's output $M(z; \theta)$ satisfies a set of hard constraints for any given input z.

Formal Statement

Formally, for a set of constraints defining a feasible set C:

$$M(z; \theta) \in C, \forall z$$



Key Difficulties

1. **Non-convexity:** The loss function L_task is highly non-convex with respect to the model weights θ.
2. **Indirect Constraints:** The constraints are on the output y, not the parameters θ. The feasible set for θ is complex and input-dependent.
3. **Generalization:** The constraints must hold for all inputs, including unseen data at test time, not just the training data.

Adaptation I

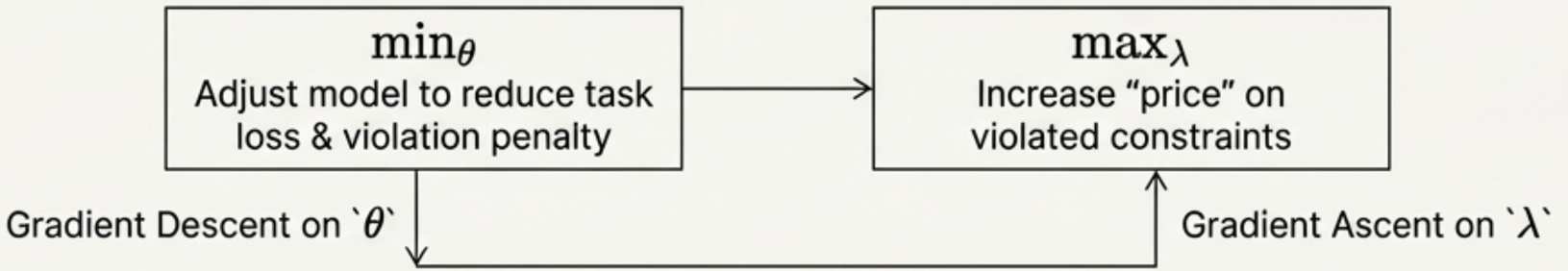
Lagrangian Methods

From Classical Duality to Penalized Loss: The classical Lagrangian $L(x, \lambda) = f_0(x) + \sum \lambda_i f_i(x)$ inspires a direct modification to the training objective of an AI model. We augment the primary task loss with a penalty term for constraint violations, weighted by learned Lagrange multipliers.

General Form of Lagrangian Loss: $L(\theta, \lambda) = L_{\text{task}}(\theta) + \sum \lambda_i \cdot C_i(M(z; \theta))$

- $L_{\text{task}}(\theta)$: The original machine learning objective (e.g., Mean Squared Error, Cross-Entropy).
- $C_i(y)$: A function measuring the violation of constraint i. For a constraint $g_i(y) \leq 0$, this could be $\max(0, g_i(y))$.
- $\lambda_i \geq 0$: The Lagrange multipliers, which are now parameters to be learned.

Training as a Saddle-Point Problem: The goal is to solve $\min_\theta \max_{\lambda \geq 0} L(\theta, \lambda)$. This is typically achieved via alternating gradient descent-ascent:



Deep Dive: Lagrangian Methods in Practice (Example: Fioretto et al.)

Problem Context: Solving constrained optimization problems like AC Optimal Power Flow (AC-OPF), where a neural network learns to predict optimal generator settings.

The model $M(\mathbf{z};\theta)$ predicts a solution $\mathbf{y}$. The constraints are $g(\mathbf{y}) \leq 0$ and $h(\mathbf{y}) = 0$. The loss is:

$$L(\theta, \lambda, \nu) = \mathcal{L}_{\text{MSE}}(M(\mathbf{z};\theta), y_{\text{true}}) + \lambda^T \max(0, g(M(\mathbf{z};\theta))) + \nu^T h(M(\mathbf{z};\theta))$$

Primal Update (on θ)

$$\theta \leftarrow \theta - \eta_\theta \cdot \nabla_\theta L(\theta, \lambda, \nu)$$

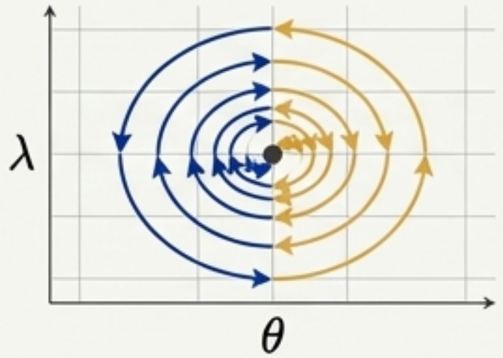
Intuition: Adjust model weights to reduce both the task loss and the constraint violations as priced by the current λ and ν.

Dual Update (on λ, ν)

$$\lambda \leftarrow \max(0, \lambda + \eta_\lambda \cdot \nabla_\lambda L(\theta, \lambda, \nu))$$

$$\nu \leftarrow \nu + \eta_\nu \cdot \nabla_\nu L(\theta, \lambda, \nu)$$

Intuition: Increase the price (λ_i or $|\nu_i|$) for constraints that are being violated. The max ensures λ remains non-negative.



Saddle-Point Optimization Trajectory

Iterative updates lead to a saddle point, balancing minimization of L w.r.t θ and maximization w.r.t λ.

Lagrangian Methods: Where They Succeed and Fail

✓ Conditions for Success

- **Well-behaved Saddle-Point Dynamics:** The method is most effective when the non-convex **min-max** optimization is stable and converges to a meaningful saddle point.
- **Differentiable Constraints:** The constraint violation functions C_i must be differentiable (or sub-differentiable) to allow for gradient-based updates.
- **Sufficient Model Capacity:** The underlying model $M(z;\theta)$ must have enough capacity to jointly minimize the task loss and satisfy the constraints.
- **Effective as a 'Soft' Regularizer:** Excellent for problems where small constraint violations are tolerable and the goal is to guide the model towards feasible regions.

✗ Causes of Failure

- **No Hard Guarantees:** This approach penalizes violations but does not eliminate them. At inference time, there is no guarantee that the model output will be feasible. Violations, though hopefully small, can still occur.
- **Optimization Instability:** The alternating gradient descent-ascent can be unstable, prone to oscillations, or slow to converge, especially in highly non-convex landscapes. Finding the right learning rates (η_θ, η_λ) can be challenging.
- **Poor Local Minima:** The process may converge to a state where both task performance and feasibility are poor, representing a poor local minimum of the saddle-point problem.

Empirical Results: Lagrangian Methods for AC-OPF

Comparison of constraint violation and objective cost on a power grid benchmark.

Method	Avg. Constraint Violation (p.u.)	Objective Cost ($)
Unconstrained NN	1.2e-1	15,230
Penalty Method	5.8e-3	15,980
Fioretto et al.	**9.1e-5**	**15,450**
Traditional Solver	0.0	15,410

- The **Lagrangian** method significantly **reduces constraint violations** by orders of magnitude compared to unconstrained or simple penalty-based training.
- It achieves **near-optimal performance** on the primary **objective** (cost), demonstrating an effective trade-off.
- While not perfectly feasible like a traditional solver, it provides a **fast** approximation that is "close enough" for many applications.

Source: Conceptual result based on Fioretto et al., "Lagrangian Duality for Constrained Deep Learning", NeurIP

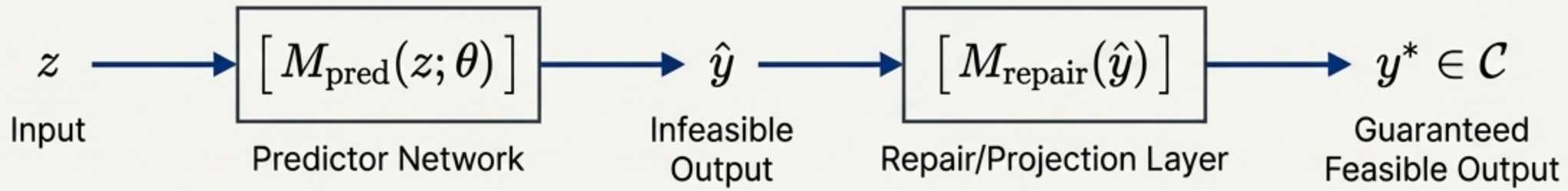
Adaptation II

Repair Methods via Projection

From Classical Projection to Output Repair: Recall the classical projection operator $P_C(\mathbf{x}_0)$ which finds the closest feasible point in a set $\mathcal{C}$. We cannot project the high-dimensional weights θ of a neural network.

The Architectural Idea: Instead, we project the *output* of the network.

1. Train a standard, unconstrained model $M_{\text{pred}}(z;\theta)$ to produce a tentative, potentially infeasible prediction $\hat{y}$.
2. Add a subsequent, non-trainable 'repair' layer M_{repair} that projects $\hat{y}$ onto the feasible set $\mathcal{C}$.



Key Advantage: This decouples the complex learning task (handled by M_{pred}) from the strict requirement of satisfying hard constraints (enforced by M_{repair}).

Deep Dive: Repair Architectures (Example: HardNet)

Problem Context: Learning to solve Sudoku, where the output must satisfy the rules of a valid grid.

The Two-Stage Process:

1. **Prediction Stage:** A standard neural network M_{pred} takes the input z (e.g., a Sudoku puzzle) and produces a tentative output $\hat{y}$ (e.g., a continuous relaxation of the solution grid).
$$\hat{y} = M_{\text{pred}}(z; \theta)$$
2. **Repair Stage:** The output $\hat{y}$ is projected onto the feasible set C. This projection is formulated as a convex optimization problem.
$$y^* = M_{\text{repair}}(\hat{y}) = \underset{y \in C}{\text{argmin}} \, \|y - \hat{y}\|_2^2$$

End-to-End Training:

For the entire model to be trainable with backpropagation, the repair layer M_{repair} must be differentiable.

- Many convex optimization problems (LPs, QPs, SOCPs) have differentiable solutions. The gradients $\partial y^* / \partial \hat{y}$ can be computed efficiently, often by solving the KKT conditions of the projection problem.
- The loss $L_{\text{task}}(y^*, y_{\text{true}})$ is backpropagated through the repair layer to update the weights θ of the predictor network. The predictor learns to produce outputs $\hat{y}$ that are not only close to the final solution but also 'easy' to repair (i.e., close to the feasible set).

Repair Methods: Where They Succeed and Fail

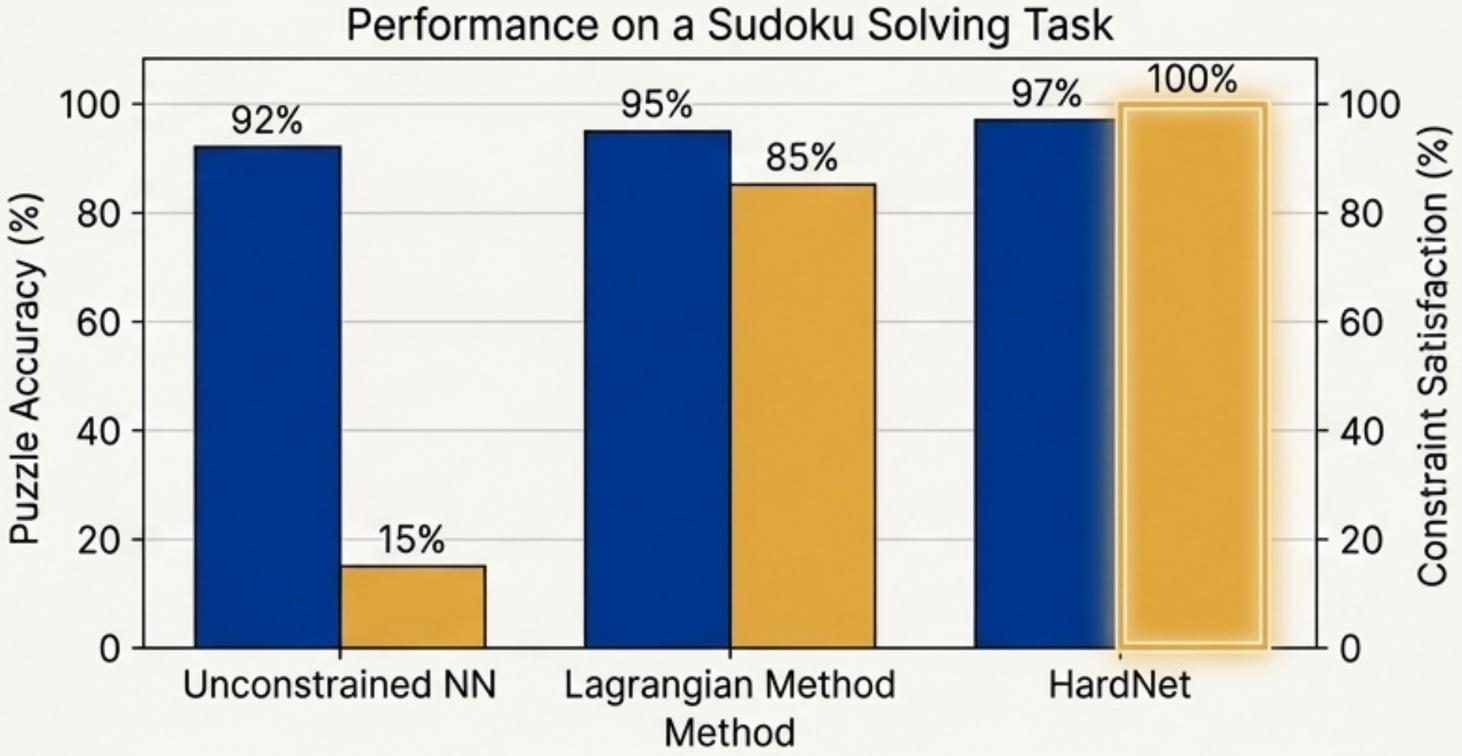
✓ Conditions for Success

- **Hard Guarantees at Inference:** By construction, the final output y^* is *always* feasible. This is the primary advantage and makes it suitable for safety-critical applications.
- **Efficiently Projectable Sets:** The method is viable when the feasible set C is convex and the projection $\text{argmin}_{y \in C} \|y - \hat{y}\|^2$ can be solved very efficiently (e.g., projection onto a simplex, a box, a polyhedron via LP/QP, or an SOCP cone).
- **Decoupling of Concerns:** The predictor network can focus on learning the complex patterns from data, while the convex projection layer handles the logic of feasibility.

✗ Causes of Failure or Difficulty

- **Projection Complexity:** If the feasible set C is non-convex or the projection problem is itself NP-hard, this method is not applicable. The complexity of the projection layer can be a computational bottleneck at inference time.
- **Performance Degradation:** If the unconstrained prediction $\hat{y}$ is very far from the feasible set, the projection y^* might be a poor solution to the original task, even if it's feasible. The 'repair' can significantly distort the network's learned solution.
- **Implementation Overhead:** Implementing a differentiable convex optimization layer is more complex than adding a simple penalty term to a loss function.

Empirical Results: Repair Methods for Structured Prediction



- The Repair Method (HardNet) achieves 100% constraint satisfaction on the test set, by design.
- Methods based on soft penalties (Lagrangian) fail to satisfy constraints on a non-trivial fraction of instances.
- The end-to-end training of the predictor and repair layer allows HardNet to maintain high task accuracy, demonstrating that the projection does not need to compromise performance.

Source: Conceptual result based on Amos et al., "OptNet: Differentiable Optimization as a Layer in Neural Networks", ICM

Synthesis: Two Philosophies for Enforcing Constraints

Feature	Lagrangian Methods	Repair Methods
Core Principle	Duality (Pricing Constraints)	Projection (Correcting Outputs)
Mechanism	Modify the **loss function** during training.	Modify the model **architecture** with a terminal layer.
Guarantee Level	**Soft**: Encourages feasibility, no test-time guarantee.	**Hard**: Guarantees feasibility by construction at test time.
When to Use	When small violations are acceptable; as a regularizer.	When constraints are non-negotiable (safety-critical).
Primary Challenge	Unstable training dynamics, convergence issues.	Computational cost and complexity of the projection layer.
Example Papers	Fioretto et al.	HardNet et al.

The choice between these methods reflects a fundamental design decision: Do we guide the learning process towards feasibility (**Lagrangian** in blue #0033A0), or do we enforce it absolutely after the fact (**Repair** in ochre #E8A838)?

A Roadmap of the Field

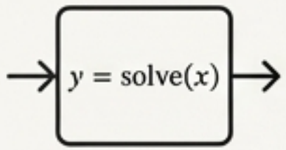
This tutorial has focused on two dominant paradigms rooted in classical optimization. A broader taxonomy of methods for handling constraints in AI (as surveyed by Van Henterick et al.) includes:

Relaxation-Based (e.g., Lagrangian Methods)

Core Idea: Make it "expensive" to violate constraints.

Projection-Based (e.g., Repair Methods)

Core Idea: Correct any violations post-prediction.



Implicit-Layer Models

Core Idea: Define layers by properties, not forward passes. Repair methods are a specific type.

Search-Based Methods

Core Idea: Combine ML with combinatorial search to explore the feasible space.

Key Trend: A clear shift from 'soft' penalty-based methods toward architectures that provide 'hard' guarantees by embedding optimization or logical reasoning directly into the model.

The Next Frontier: From Adaptation to Intrinsic Structure

The Story So Far

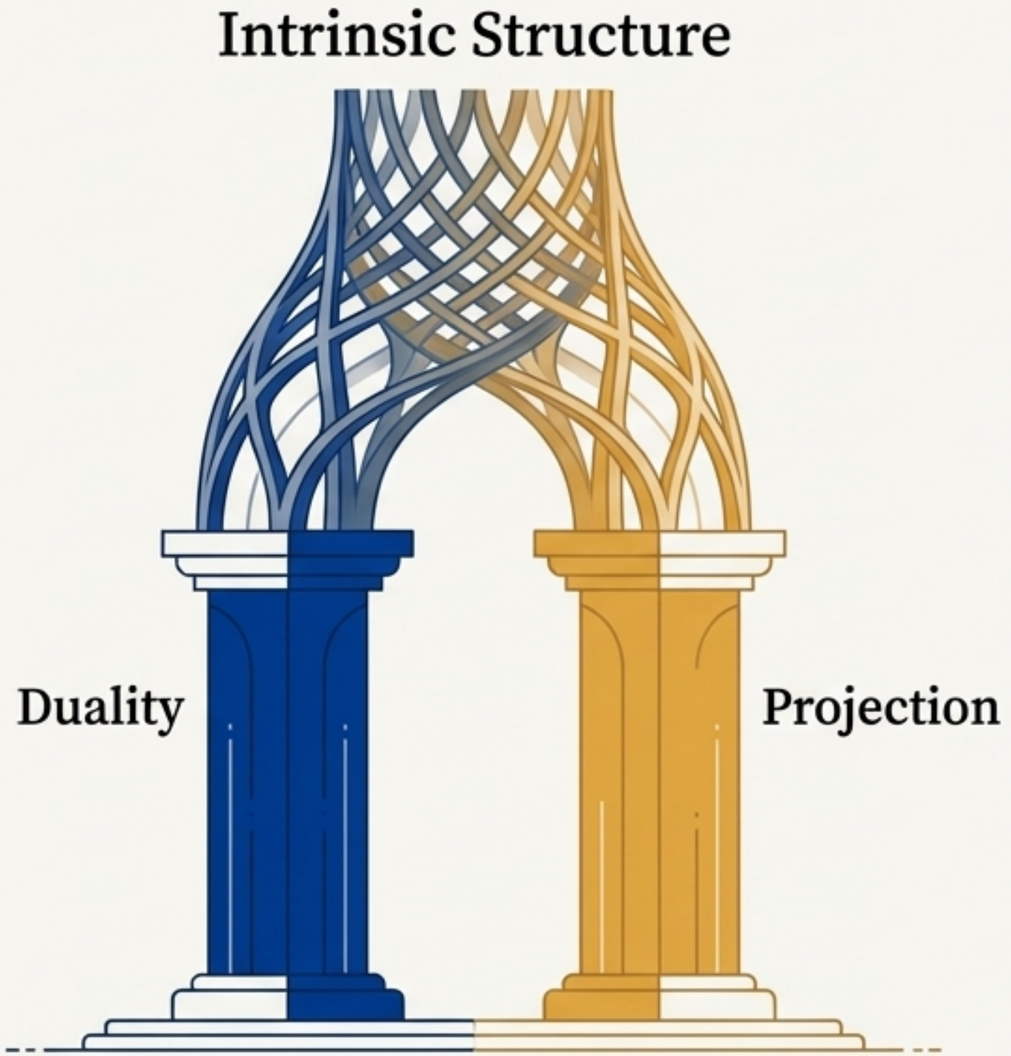
We have successfully adapted the classical principles of **duality** and **projection** from the convex world to manage constraints in non-convex deep learning models.

- **Lagrangian Methods** use duality to *guide* learning.
- **Repair Methods** use projection to *enforce* feasibility.

Future Research Directions

The Deeper Challenge: Both approaches treat the expressive neural network and the rigid constraint structure as separate entities to be reconciled. Can we design model architectures that are *intrinsically* constraint-aware?

- **Structured Architectures:** Can we design neural networks whose inherent symmetries and computational structure guarantee satisfaction of certain constraints (e.g., conservation laws)?
- **Differentiable Reasoning**: Moving beyond simple convex projections to embedding more complex symbolic or logical reasoning modules within networks.
- **A Unified Theory:** Developing a framework that merges statistical learning with formal guarantees from the ground up.



The goal is not just to build models that *obey* the rules, but models that *understand* them.