6.1.6 Training Strategies and Practical Considerations

Training L2O policies is itself a challenging optimization task. Policies must learn to balance optimality, feasibility, and generalization while avoiding collapse into trivial or infeasible solutions. The following strategies are commonly employed to improve stability and performance in practice.

Optimization Techniques. Practical success often hinges on stabilizing the learning dynamics of π_θ through careful optimizer design and training heuristics.

- *Curriculum learning:* Start training on simpler or smaller instances, gradually increasing difficulty (problem size, constraint tightness, horizon length).
- *Warm-start initialization:* Initialize policies using outputs of classical solvers or heuristics to avoid poor local minima early in training.
- *Adaptive weighting:* Dynamically adjust penalties on constraint violations (λ_1, λ_2) to ensure that feasibility is prioritized without stalling progress on the objective.
- *Gradient stabilization:* Use clipping, normalization, or variance reduction to handle unstable gradients that arise from constraint layers and unrolled optimizers.

Constraint Handling. A central design choice is how to enforce feasibility during training and deployment, balancing strict guarantees with flexibility and tractability.

- *Hard layers:* Projections, proximal maps, and equality-satisfying parameterizations guarantee feasibility for simple sets (e.g., bounds, linear constraints).
- *Soft penalties:* Augmented Lagrangians or barrier methods enforce feasibility approximately, trading guarantees for flexibility.
- *Hybrid schemes:* Combine hard enforcement for simple constraints with soft terms for complex couplings (common in nonlinear or mixed-integer problems).

Data and Supervision. The training regime is shaped by the availability and cost of solver-generated labels, which determine whether supervised, self-supervised, or hybrid approaches are most effective.

- *Supervised training:* Beneficial when high-quality solver outputs are available; pretraining on labeled solutions accelerates convergence.
- *Self-supervised training:* Useful when solvers are expensive or infeasible; requires careful design of the loss to prevent trivial solutions.
- *Semi-supervised regimes:* Combine a small labeled dataset with many unlabeled problem instances, leveraging solver labels for calibration and self-supervision for scalability.

Generalization and Robustness. Ensuring that learned optimizers transfer reliably beyond the training distribution is critical for real-world deployment in safety- or cost-critical domains.

- *Distributional robustness:* Training on diverse instances and introducing adversarial perturbations improves out-of-distribution performance.
- *Regularization:* Enforce smoothness, stability, or Lipschitz constraints on π_θ to promote generalization and avoid overfitting to the training distribution.
- *Validation metrics:* Evaluate both objective suboptimality and feasibility violation, since low loss does not always imply constraint satisfaction.

Key Takeaways

L2O offers dramatic speedups and seamless integration into constrained optimization pipelines, but reliable deployment requires careful training. Effective strategies include solver-informed pretraining, constraint-aware architectures, and robust optimization techniques. In practice, the choice of approach depends on the availability of solver labels, the complexity of constraints, and the need for certified feasibility, with hybrid designs—combining supervised initialization, self-supervised fine-tuning, and feasibility mechanisms—often providing the best balance between performance and reliability.