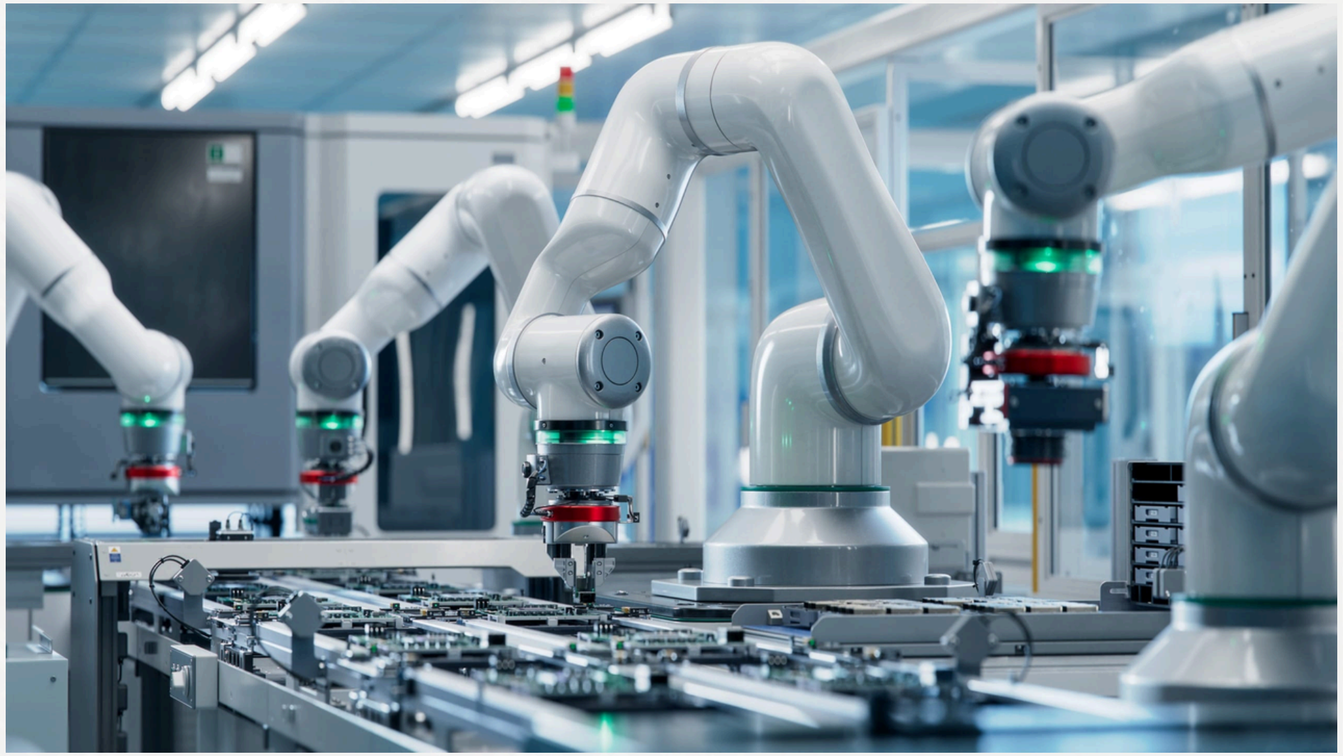
Improving Feasibility via Fast Autoencoder-Based Projections

14TH INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS

Authors: Maria Chzhen, Priya L. Donti

Importance of efficient constraint satisfaction



Goal: Efficiently improve feasibility of neural network outputs for constrained optimization and RL.

Related Work

1 **Amortized optimization** (*aka* “learning to optimize”)

Paradigm designed to accelerate the process of solving optimization problems by using ML models as function approximators

- **Penalty methods:** Penalize violations in loss; no feasibility guarantees
- **Differentiable projection and/or correction:** Use exact feasibility enforcement within end-to-end training; slow & specialized
- **Post-hoc repair:** Use exact feasibility enforcement during inference only; slow, specialized, & can degrade performance

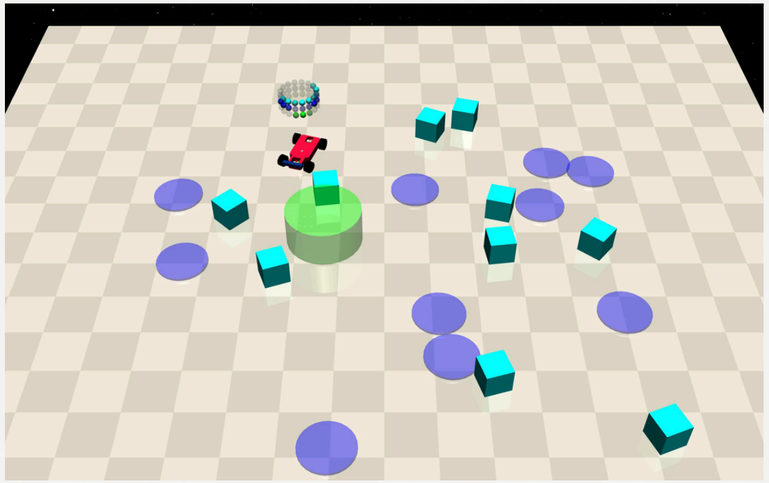
Trade-offs: Reliability vs. speed vs. generality.

Related Work

2 Safe reinforcement learning

Agent learns to make sequences of safer decisions while maximizing cumulative reward in an environment

- **Lagrangian relaxation** (penalty method)
- **Differentiable projection layers**

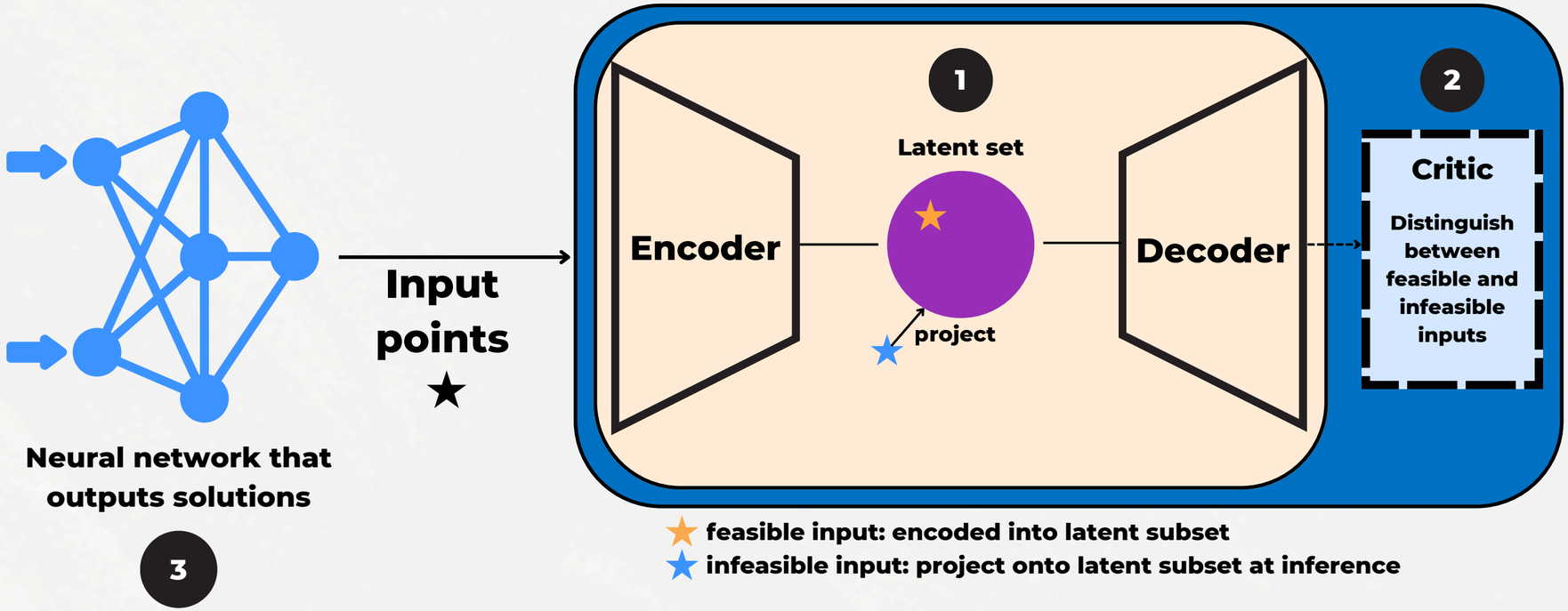


Proposed Approach

Develop **autoencoder with structured latent space** to rapidly enforce constraints

- Idea: Create a convex part of the latent space that decodes to the feasible set, via two-phase autoencoder training

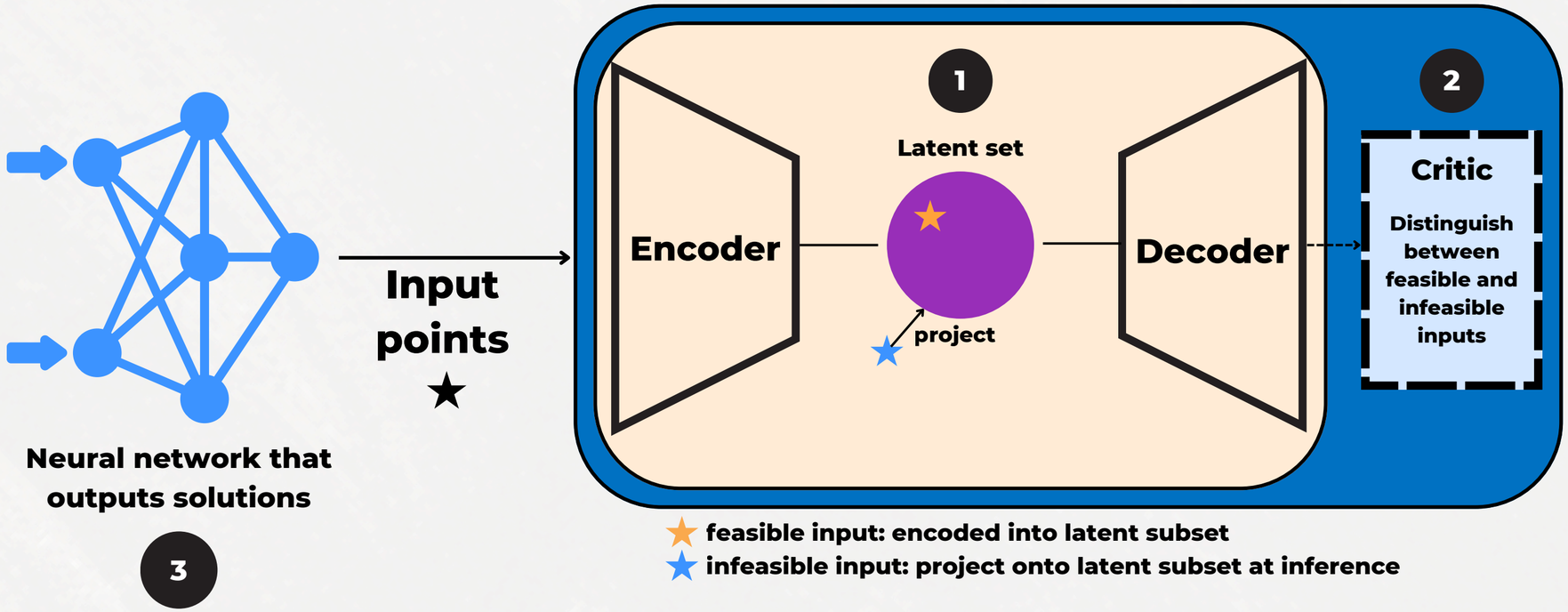
Allows for feasibility correction of NN outputs via fast projection in the latent space



Proposed Approach

Two autoencoder training phases

1. Constraint set reconstruction
2. Latent space structuring (via adversarial training)



Two-Phase Autoencoder Training

1

Feasible set reconstruction

- Train on feasible points only, learn to reconstruct feasible points (MSE loss)

2

Latent space structuring

- Train the autoencoder + a feasible/infeasible discriminator *D* on feasible points, infeasible points, and random samples from the convex latent subset
- Discriminator and autoencoder have distinct loss functions:

$$\mathcal{L}_{\text{disc}}(y, x, c) = -\left(c \log D_\xi(y, x) + (1 - c) \log\left(1 - D_\xi(y, x)\right)\right)$$

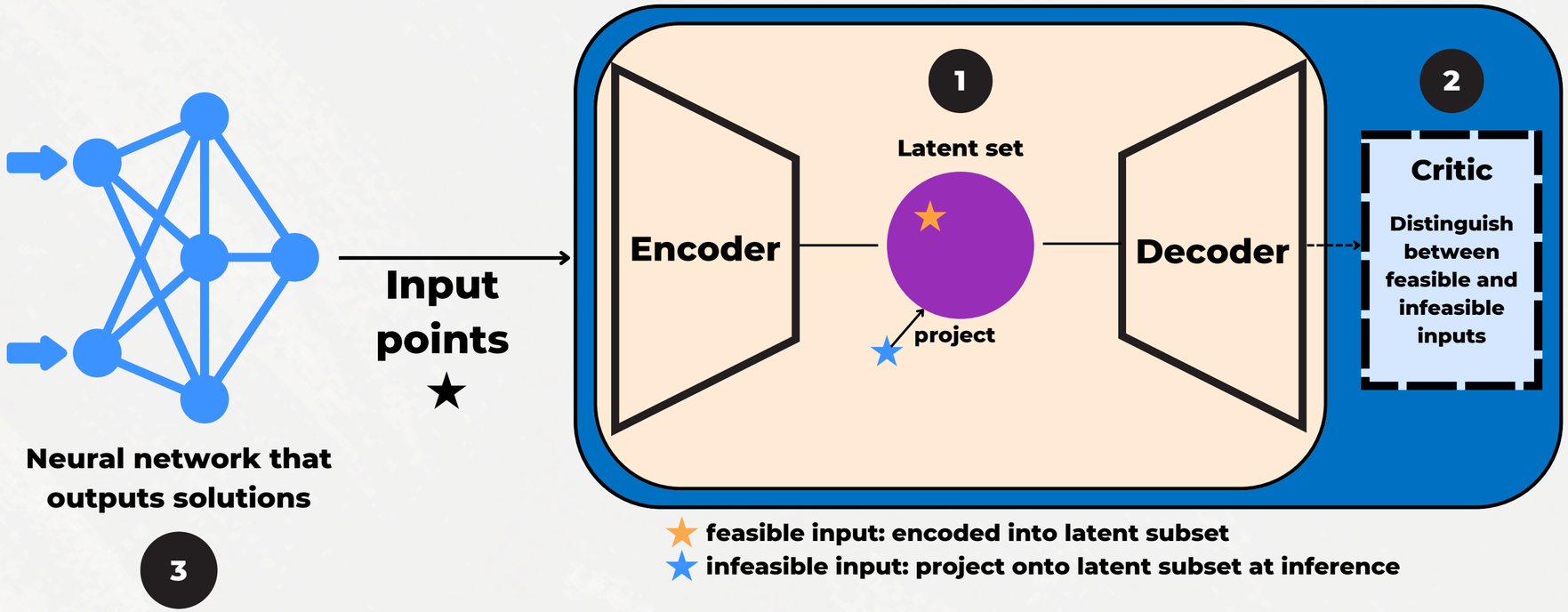
$$\mathcal{L}_{\text{struc}} = \frac{1}{N}\sum_{i=1}^{N}\left[\lambda_{\text{recon}}\mathcal{L}_{\text{recon}}(y^{(i)}, x^{(i)})\right] + \frac{1}{N}\sum_{i=1}^{N}\left[\lambda_{\text{hinge}}\mathcal{L}_{\text{hinge}}(y^{(i)}, x^{(i)}, c^{(i)})\right]$$
$$+ \frac{1}{M}\sum_{j=1}^{M}\left[\lambda_{\text{latent}}\mathcal{L}_{\text{latent}}(z^{(j)})\right] + \lambda_{\text{geom}}\mathcal{L}_{\text{geom}}(\{z^{(1)}, \ldots, z^{(M)}\}),$$

Leveraging the trained feasibility mapping

Training: Train NN + (fixed-weight) autoencoder end-to-end to solve the original problem

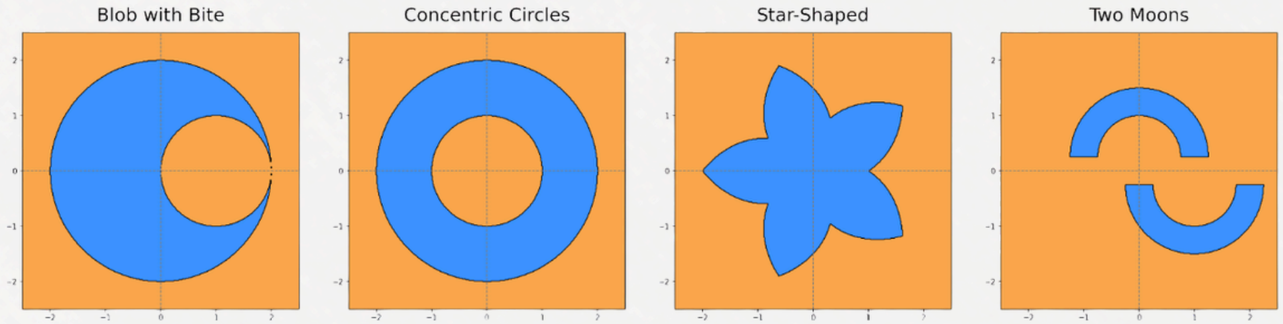
- Autoencoder encodes the predicted solutions from the base NN.
- If encoded solution is outside of latent subset, cheaply project before decoding.

Inference: Use the final NN + autoencoder

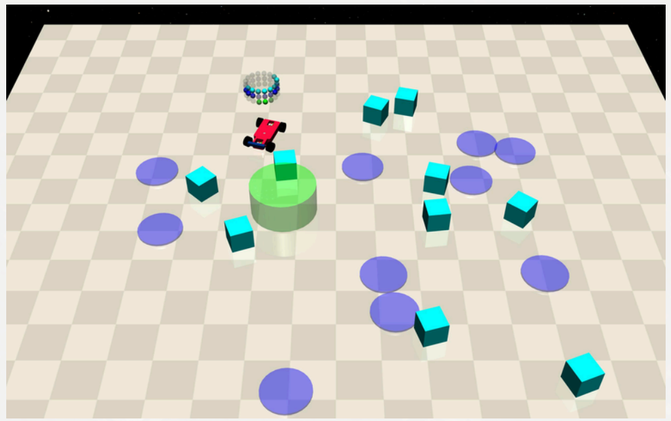


Experiments

Constrained Optimization: Input-independent nonconvex constraint sets



Reinforcement Learning: Input-dependent nonconvex constraint sets



Constrained Optimization

Linear:

$$\min_y a^\top y \quad \text{s.t. } y \in \mathcal{C}$$

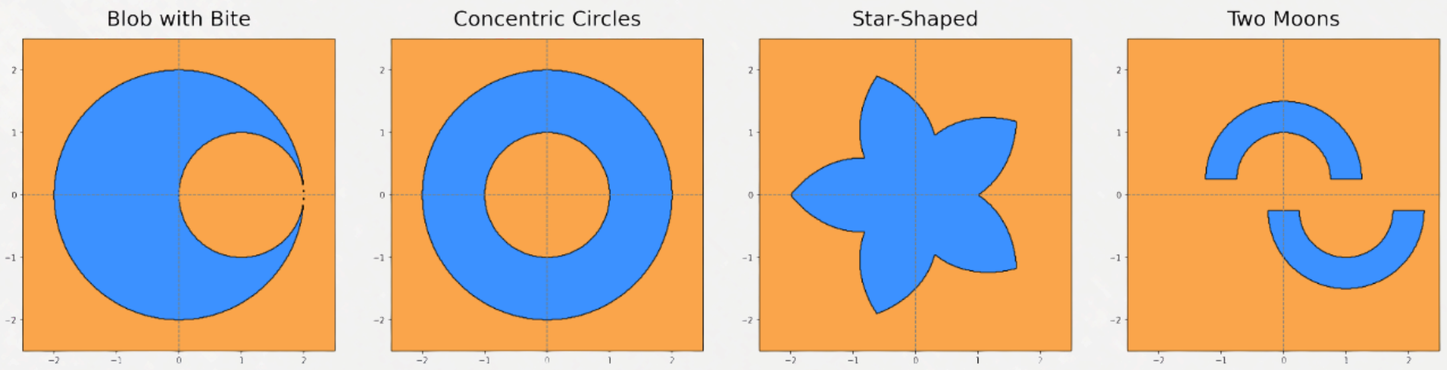
Quadratic:

$$\min_y y^\top Q y + a^\top y \quad \text{s.t. } y \in \mathcal{C}$$

Distance minimization:

$$\min_y \|y - t\|^2 \quad \text{s.t. } y \in \mathcal{C},$$

Tested on four classes of 2-dimensional nonconvex constraint families (below), as well as 3-, 5-, and 10-dimensional hyperspherical shell constraints



Constrained Optimization

Results: Quadratic objective, Blob with Bite

	↓ Optim. gap (%)	↑ Feas (%)	↓ Time (ms)
Projected Grad.	0.60 ± 0.88	100 ± 0.0	38.73 ± 28.68
FSNet	1.15 ± 1.67	99.5 ± 7.3	2.66 ± 11.09
Penalty NN	0.50 ± 0.76	97.1 ± 16.9	0.19 ± 0.06
FAB (ours)	0.53 ± 0.71	100 ± 0.0	0.69 ± 1.44

Across experiments, FAB achieves near-perfect feasibility and sub-millisecond inference of ~0.5ms, while enabling reasonable optimality gap

Constrained Optimization

Results: Quadratic objective, Blob with Bite

	↓ Optim. gap (%)	↑ Feas (%)	↓ Time (ms)
Projected Grad.	0.60 ± 0.88	100 ± 0.0	38.73 ± 28.68
FSNet	1.15 ± 1.67	99.5 ± 7.3	2.66 ± 11.09
Penalty NN	0.50 ± 0.76	97.1 ± 16.9	0.19 ± 0.06
FAB (ours)	0.53 ± 0.71	100 ± 0.0	0.69 ± 1.44

Across experiments, FAB achieves near-perfect feasibility and sub-millisecond inference of ~0.5ms, while enabling reasonable optimality gap

Constrained Optimization (Ctd.)

Results: Quadratic objective, Two Moons

	↓ Optim. gap (%)	↑ Feas (%)	↓ Time (ms)
Projected Grad.	1.18 ± 1.58	80.5 ± 39.6	79.007 ± 91.65
FSNet	1.13 ± 1.45	98.7 ± 11.5	4.42 ± 2.77
Penalty NN	0.61 ± 1.06	91.1 ± 28.4	0.20 ± 1.02
FAB (ours)	0.85 ± 1.16	96.8 ± 17.6	0.55 ± 0.40

Results are robust even for complex, disjoint sets

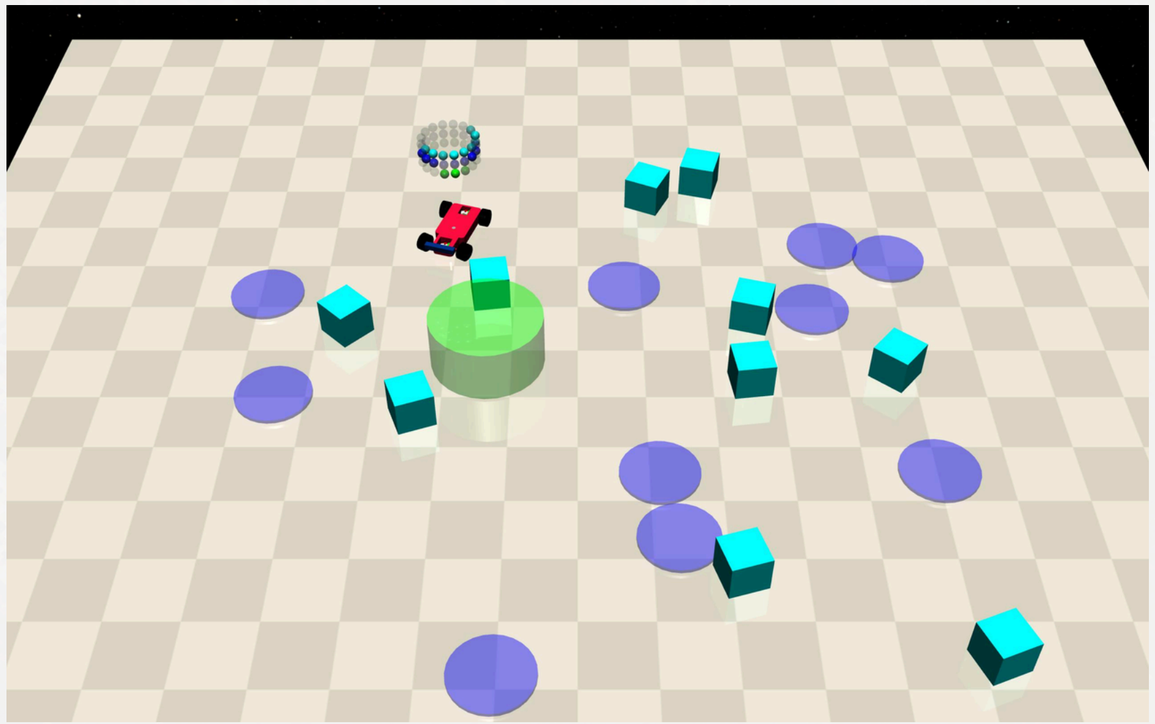
Further ablations, robustness tests, and variants in paper:

- Omitting autoencoder training phases or loss function components
- Incomplete coverage of the constraint set in the autoencoder training data
- Using different decoder depths/widths, or multiple decoders

Reinforcement Learning

Safety Gymnasium benchmarks (SafetyPointGoal2-v0, SafetyPointPush2-v0)

Constraints are *state-dependent* (autoencoder is a conditional autoencoder)



Reinforcement Learning

FAB achieves lowest cost (fewest violations), fastest inference

Rewards could definitely be higher (“cost of feasibility”)

Algorithm	Reward Mean ↑	Reward Std ↓	Cost Mean ↓	Cost Std ↓	Time Mean (s) ↓
PPO_FAB	-1.12	**1.11**	**24.32**	**37.84**	2.75
PPO	13.26	14.05	167.46	87.06	2.47
PPO_LAG	2.24	5.10	54.10	64.50	**2.40**
TRPO	**15.58**	10.31	164.14	88.43	2.59
TRPO_LAG	2.37	8.46	89.04	187.67	2.78

Results: SafetyPointGoal2-v0

Limitations and Future Directions

Limitations:

- No hard feasibility guarantees.
- Relies on representative constraint-set data.
- Instability of adversarial training (requires hyperparameter tuning).
- So far, only tested on small-scale settings.

Future directions:

- Improve sample efficiency, distributional robustness, interpretability.
- Explore alternative feasibility enforcement architectures (e.g., input-convex networks, operator learning).
- Formal verification of learned feasibility mapping.
- Co-training autoencoder with base networks for end-to-end learning.

Conclusion

FAB: Fast, data-driven feasibility mapping via autoencoder.

Benefits: Near-perfect feasibility, sub-millisecond inference, robust to nonconvex/disjoint sets.

Positioning: Middle ground between penalty methods and exact enforcement. Practical for latency-sensitive applications.

Link to code

Link to paper