CAffNet guarantees satisfaction of input-dependent affine constraints through constraint decomposition and closed-form trainable projection, while preserving neural network expressiveness.

CAffNet: Hard Constraint-Affine Neural Networks

Yang Zhao[1], Jungeun Lee[2], Jeong hwan Jeon[2], Sze Zheng Yong[1]

[1] *Mechanical and Industrial Engineering, Northeastern University;* [2] *Electrical Engineering, Ulsan National Institute of Science and Technology*

Background

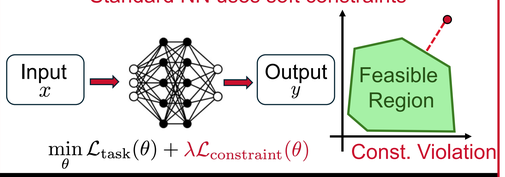
Neural networks are widely used in safety-critical systems

Autonomous Driving · Robotics · Aerospace · Power Systems · Healthcare

Need to satisfy input-dependent constraints to ensure safety

$$A(x)y \le b(x)$$

Standard NN uses soft constraints



However, soft const. do NOT guarantee safety!

Motivation: How to guarantee safety ?

Using projection-based methods is one way to ensure hard constraints.

	Existing Methods	Ours	
HardNet (Min & Azizan)	⊖ Limited # of const.	✓ Arbitrary # of const.	CAffNet
	⊖ Full-row rank const. (linearly independent)	✓ No rank limits (handles linearly dependent const.)	
	⊖ Parallel projection	✓ Trainable projection	
DC3 (Donti et al.) FSNet (Nguyen & Donti)	⊖ Asymptotic const. satisfaction	✓ Always const. satisfaction	
	⊖ Iterative computation	✓ Closed-form computation	

Methods

We address limitations with Constraint Decomposition and Trainable Projection

Constraint Decomposition

Considering combinations of choosing k constraints out of m

$$k \in [1, \min(m, n_{out})] \implies \Gamma_k := \{\gamma = (j_1, \dots, j_k) \mid 1 \le j_1 < \cdots < j_k \le m\}$$

Original Constraints

$$A = \begin{bmatrix} a_1 \\ \vdots \\ a_m \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

$$\Gamma = \bigcup_{k=1}^{\min(m,n_{out})} \Gamma_k \longrightarrow A_\gamma = \begin{bmatrix} a_{j_1} \\ \vdots \\ a_{j_k} \end{bmatrix} \quad b_\gamma = \begin{bmatrix} b_{j_1} \\ \vdots \\ b_{j_k} \end{bmatrix}$$

Sub-Constraints

Projection with Sub-Constraints

Projections

$$\mathcal{P}_\gamma(x) = f_\theta(x) - A_\gamma^\dagger(x)(A_\gamma(x) f_\theta(x) - b_\gamma(x)) + (I - A_\gamma^\dagger(x) A_\gamma(x)) w_\phi(x)$$

Null-space component

Candidates

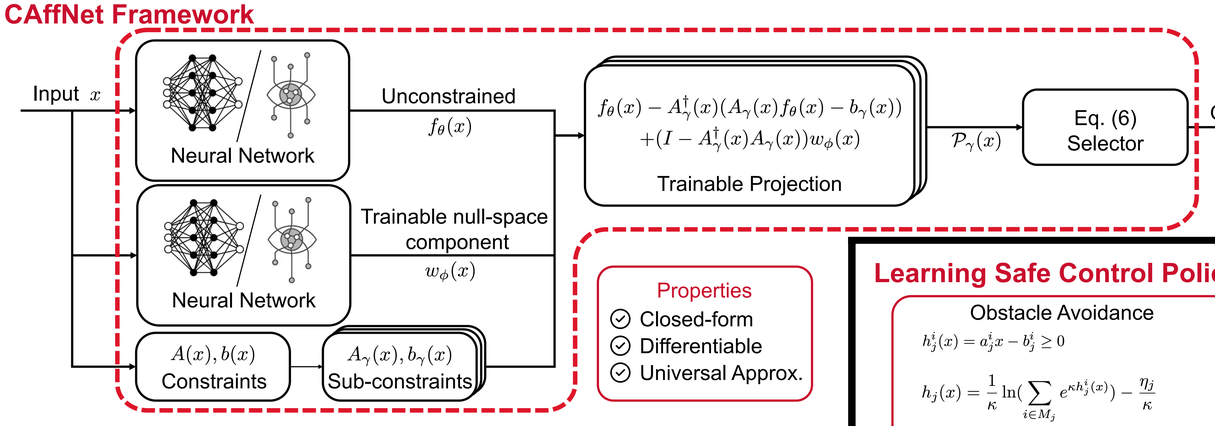
$$\mathcal{S}_\mathcal{P}(x) := \{\mathcal{P}_\gamma(x) \mid \gamma \in \Gamma,\ A(x)\mathcal{P}_\gamma(x) \le b(x)\}$$

Optimal output

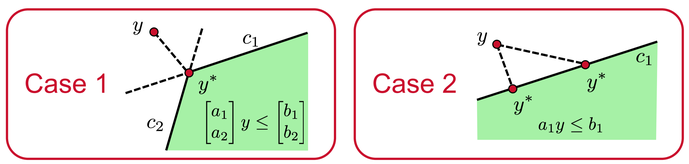
$$\mathcal{P}^*(x) = \begin{cases} f_\theta(x), & \text{if } A(x) f_\theta(x) \le b(x), \\ \arg\min_{y \in \mathcal{S}_\mathcal{P}(x)} \|y - f_\theta(x)\|_p, & \text{otherwise.} \end{cases}$$

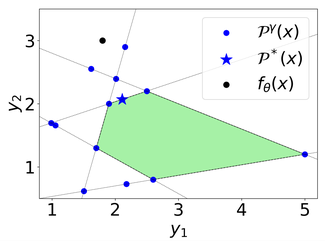
Our projection is trainable and selects the closest feasible candidate

CAffNet Framework



Properties
- ✓ Closed-form
- ✓ Differentiable
- ✓ Universal Approx.

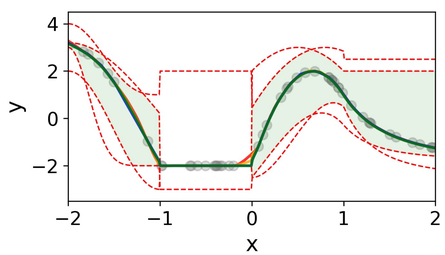


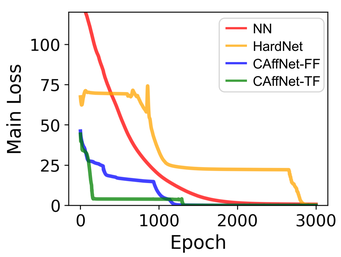


Experiments

NN and HardNet violate constraints, while CAffNet always guarantees satisfaction

Learning Nonlinear Functions





Method	MSE	Inequality violation Max	Inequality violation Mean	T_{train} (ms)	T_{test} (ms)
NN	0.0045 (0.0057)	0.0074 (0.0074)	0.0019 (0.0019)	**5.3100** (0.16)	**3.3700** (0.20)
HardNet	0.0037 (0.0063)	0.0033 (0.0037)	0.0008 (0.0009)	8.0000 (0.34)	3.7000 (0.24)
CAffNet-FF	0.0020 (0.0032)	**0.0000** (0.0000)	**0.0000** (0.0000)	11.9300 (0.55)	7.1100 (0.49)
CAffNet-TF	**0.0012** (0.0009)	**0.0000** (0.0000)	**0.0000** (0.0000)	15.5700 (0.32)	14.7100 (2.04)

- CAffNet-TF obtains the lowest MSE
- CAffNet-FF and -TF converge faster than NN and HardNet in training

Learning Optimization Solvers

$$f(x) = \arg\min_{y \in \mathbb{R}^{n_{out}}} \ \frac{1}{2} y^\top Q y + p^\top \sin(y) \quad \text{s.t.} \ \ Gy \le h, Cy = x$$

- CAffNet-FF and -FF (Lite) achieve the smallest optimality gap to baseline IPOPT, while running much faster at inference.
- Lite further reduces computation

Method	Obj. value	Inequality violation Max	Inequality violation Mean	Inequality violation Num (%)	Eq viol. Max	Eq viol. Mean	Eq viol. Num (%)	T_{train} (ms)	T_{test} (ms)
IPOPT	**-0.3494** (0.4995)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	- (-)	19799.8000 (295.16)
NN	-0.1194 (0.5824)	0.0018 (0.0015)	0.0004 (0.0003)	0.31% (0.21%)	0.1136 (0.0035)	0.0685 (0.0023)	100.00% (0.00%)	**4.2700** (0.09)	**1.2300** (0.01)
HardNet	-0.3000 (0.4975)	0.0027 (0.0026)	0.0006 (0.0006)	3.24% (2.91%)	0.0435 (0.0123)	0.0258 (0.0071)	99.99% (0.01%)	7.3800 (0.20)	2.1100 (0.03)
CAffNet-FF	**-0.3418** (0.5028)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	1325.3800 (26.28)	1298.6700 (26.28)
CAffNet-FF (Lite)	**-0.3418** (0.5028)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	737.1600 (20.54)	713.6900 (20.73)
CAffNet-TF	-0.1576 (0.3653)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	1327.7900 (24.21)	1299.9500 (24.62)

Learning Safe Control Policies

Obstacle Avoidance

$$h_j^i(x) = a_j^i x - b_j^i \ge 0$$

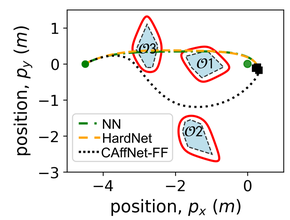
$$h_j(x) = \frac{1}{\kappa} \ln\Big(\sum_{i \in M_j} e^{\kappa h_j^i(x)}\Big) - \frac{\eta_j}{\kappa}$$

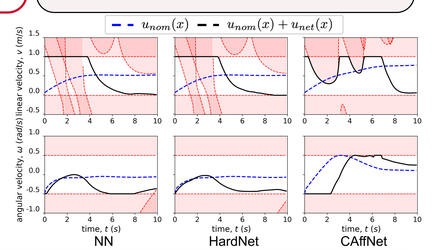
$$L_f h_j(x) + L_g h_j(x) u(x) \ge -\alpha(h_j(x))$$

State Constraint $h_x(x) = b_x - A_x x \ge 0$

Input Constraint $A_u(x) u(x) \le b(x)$

- NN and HardNet hit obstacles
- CAffNet guarantees safety





Method	Cost	Inequality violation Max	Inequality violation Mean	Inequality violation Num (%)	T_{train} (ms)	T_{test} (ms)
NN	$\mathbf{4.1411 \times 10^5}$ (6.7984×10^4)	0.1348 (0.0083)	0.0106 (0.0006)	2.60% (0.03%)	**2895.1300** (263.98)	**47.1700** (4.17)
HardNet	4.5701×10^5 (4.2737×10^4)	0.1317 (0.0066)	0.0104 (0.0005)	2.60% (0.03%)	4885.6000 (231.74)	855.5100 (81.92)
CAffNet-FF	7.2060×10^5 (6.9090×10^4)	**0.0000** (0.0000)	**0.0000** (0.0000)	**0.00%** (0.00%)	12207.9300 (306.59)	1885.1000 (45.96)

Discussion

$w_{\phi,\gamma}(x)$ Constraint-Specific vs. $w_\phi(x)$ Shared Null-Space Component

- ✓ Flexible Expressive
- ⊖ Compute-heavy Memory-intensive

Scalability Improvements

- Combinatorial explosion → CAffNet-Lite
- Hardware limits → Batched processing
- $A^\dagger$ Pseudoinverse → GPU acceleration Precomputation

Conclusion

Constraint Decomposition	Trainable Projection	Hard Guarantees
Decompose arbitrary numbers of input-dependent affine constraints into manageable sub-constraints	Learns the best feasible projection within the constraint-affine space	Guarantees satisfaction for all inputs while preserving expressive power

References

Min & Azizan. (2024). HardNet: Hard-constrained neural networks with universal approximation guarantees. arXiv:2410.10807.

Donti et al. (2021). DC3: A learning method for optimization with hard constraints. ICLR.

Nguyen & Donti. (2025). FSNet: Feasibility-seeking neural network for constrained optimization with guarantees. NeurIPS.

Learn More!

LinkedIn