SOC-ICNN: From Polyhedral to Conic Geometry for Learning Convex Surrogate Functions

Kang Liu
School of Future Technology
Xi'an Jiaotong University
Xi'an, China
kanyo@foxmail.com

Jianchen Hu*
School of Future Technology
School of Automation Science and Engineering
Xi'an Jiaotong University
Xi'an, China
horace89@xjtu.edu.cn

Wei Peng
School of Automation Science and Engineering
Xi'an Jiaotong University
Xi'an, China
pengwei@stu.xjtu.edu.cn

Abstract

Finite-width ReLU-based Input Convex Neural Networks (ICNNs) admit exact linear programming (LP) value-function representations, but their outputs are restricted to convex piecewise-linear functions. We propose SOC-ICNN for surrogate learning in which convexity with respect to designated decision variables is known or required. The architecture extends the LP representation to second-order cone programming (SOCP) by augmenting a ReLU-ICNN backbone with positive semidefinite quadratic and Euclidean norm primitives. We prove that SOC-ICNN strictly enlarges the representable function class and preserves the asymptotic order of forward-pass complexity when the total structural width grows at most proportionally to the backbone width. Experiments demonstrate improved convex function approximation and competitive downstream decision quality. Additional robust regression and optimal transport experiments show that the benefit of the norm branch depends on the target geometry. The code is available at https://github.com/Kanyooo/SOC-ICNN.

1 Introduction

Input Convex Neural Networks (ICNNs) rigorously guarantee the convexity of their outputs with respect to specified inputs. Unlike generic black-box models, a convex surrogate supports global optimization over convex feasible sets, subject to numerical solution accuracy, which is a critical advantage in downstream applications such as structured prediction, inverse optimization, optimal transport, and parametric decision-making [Amos et al., 2017, Makkuva et al., 2020, Rosemberg et al., 2024].

A standard paradigm is to learn a parameterized convex surrogate $f_\Theta(\boldsymbol{x})$ and subsequently solve the downstream task:

$$\min_{\boldsymbol{x}\in\mathcal{X}} f_\Theta(\boldsymbol{x}), \tag{1}$$

where $\mathcal{X}$ is a convex feasible set and Θ denotes the network parameters. For the widely used ReLU-ICNN, we recall the known linear programming (LP) interpretation of inference [Amos et al., 2017]

*Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

and state the corresponding value-function representation in our notation. In this view, the network output identically equals the optimal value of a parameterized LP. This perspective clarifies both their structural strength (the natural representation of continuous piecewise-linear (CPWL) convex polyhedra) and their fundamental intrinsic limitation: they cannot natively encode smooth curvature or non-polyhedral conic geometries, such as Euclidean norms. Quantitatively, approximating strongly convex targets requires a number of affine pieces that grows as a power of inverse accuracy with an exponent proportional to dimension. This is a piece-complexity statement; converting it into a network-parameter lower bound requires additional architectural assumptions. Recent theoretical work further substantiates this by proving lower bounds on the depth complexity of ICNNs and characterizing the exact conditions under which a ReLU network can be convex [Gagneux et al., 2025, Bakaev et al., 2025].

Recent research has sought to improve ICNN expressivity predominantly along two lines. The first line enriches local nonlinearities, for instance, by employing smoother activations (e.g., Softplus) or learnable univariate basis functions, as seen in Input Convex Kolmogorov-Arnold Networks (ICKAN) [Deschatre and Warin, 2025]. The second line focuses on improving trainability through tailored initialization schemes or architectural adaptations [Hoedt and Klambauer, 2023]. These approaches improve flexibility or trainability, but do not by themselves provide the explicit quadratic and Euclidean-norm primitives considered here. Changing the activation also requires a separate analysis of any exact optimization representation; the ReLU-based LP lift does not automatically carry over.

Another relevant trajectory builds upon differentiable convex optimization layers, where a convex program is explicitly embedded as a neural network layer and differentiated via its solution map [Agrawal et al., 2019, Besançon et al., 2024]. More recently, Learning Parametric Convex Functions (PCF) has been proposed to fit disciplined-programming-compatible convex families directly from data [Schaller et al., 2025]. Optimization layers use a solver to obtain decisions; this is a different operation from evaluating a learned objective. Solver-compatible convex models also provide a complementary route to downstream optimization. Furthermore, they often require the optimization template or disciplined structure to be specified *a priori*.

In this paper, we pursue a structural route that complements activation design and solver-based modeling. Since ReLU-ICNNs admit exact LP value-function representations, a natural question arises: *Can we elevate the architecture to a richer convex optimization class while strictly preserving both the transparent optimization interpretation and rapid forward inference?* Among mathematical programming extensions beyond LP, Second-Order Cone Programming (SOCP) provides a natural candidate. It elegantly unifies quadratic curvature and norm-based geometry, which together cover a vast class of smooth and structured convex objectives. Motivated by this, we propose the SOC-ICNN, which augments the polyhedral ReLU backbone with two explicit geometric primitives: (1) **quadratic branch** to inject positive semi-definite (PSD) curvature: $\frac{\alpha}{2}\|B\boldsymbol{x} + \boldsymbol{e}\|_2^2$. (2) **conic branch** to capture Euclidean norm geometry: $\lambda\|A\boldsymbol{x} + \boldsymbol{d}\|_2$.

Because both primitives are convex, their non-negative linear combination strictly preserves convexity. Crucially, each primitive admits an SOCP epigraph lift. Consequently, the entire forward pass of the network can be mathematically formulated as the optimal value of a finite-dimensional SOCP. The proposed SOC-ICNN therefore constitutes an architectural upgrade from LP to SOCP, introducing non-CPWL primitives and retaining a rigorous optimization-theoretic interpretation. The primary contributions of this paper are twofold:

1. **Architectural generalization:** We propose SOC-ICNN, systematically elevating the underlying mathematical structure of ICNNs from LP to SOCP value functions by explicitly injecting curvature and conic geometric primitives, with the same lifting principle extending naturally to convolutional and finite-horizon recurrent architectures.
2. **Theoretical guarantees:** We formally prove that the SOC-ICNN strictly expands the representational function class of classical ReLU-ICNNs, while preserving the asymptotic order of forward computation when the total structural width is proportional to the backbone width.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 details the architectural design of SOC-ICNN. Section 4 provides a rigorous theoretical analysis. Section 5 presents empirical evaluations, followed by conclusions in Section 6. The appendix provides complete

proofs, full experimental results, and additional convolutional and finite-horizon recurrent SOC-ICNN extensions with downstream tests.

2 Related Work

ICNNs and their expressive extensions. ICNNs were introduced in [Amos et al., 2017] as neural architectures with built-in input convexity, including both fully input-convex and partially input-convex constructions, thereby enabling optimization-aware prediction and inference. Since then, ICNNs have been used as structured convex surrogates in applications such as optimal transport [Makkuva et al., 2020], value-function learning for optimal power flow [Rosemberg et al., 2024], two-stage stochastic programming [Liu et al., 2025], and power system contingency screening with provable reliability guarantees [Christianson et al., 2025].

A central limitation of finite-width ReLU-ICNNs, however, is that their representable class is tied to polyhedral, piecewise-linear convex functions. Existing work has mainly addressed this limitation by enriching the parameterization or extending the backbone within the ICNN framework. One line adapts ICNNs to sequential and control-oriented settings, including input-convex recurrent models for optimal control [Chen et al., 2019], input-convex LSTM architectures for real-time optimization [Wang et al., 2025], and recurrent variants that further incorporate Lipschitz constraints for robustness and efficient process modeling [Wang et al., 2024]. Another line enriches the internal parameterization of input-convex models. In particular, ICKAN [Deschatre and Warin, 2025] introduces learnable univariate basis functions into input-convex architectures, improving empirical flexibility beyond standard ReLU-based constructions. Concurrently, theoretical investigations have characterized the necessary and sufficient conditions under which a ReLU network is convex [Gagneux et al., 2025], and have established lower bounds on the depth complexity of ICNNs, highlighting fundamental expressivity limitations [Bakaev et al., 2025].

While these approaches enlarge the practical approximation power of ICNNs, they do not explicitly upgrade the underlying optimization-theoretic structure beyond the LP/polyhedral regime. In contrast, our goal is not merely to enrich the local nonlinearity or change the network backbone, but to lift the value-function class itself, moving from LP value functions to SOCP value functions.

Parametric convex models and differentiable convex programs. Another related line of work builds optimization-aware models by embedding or learning convex programs directly. Early work such as OptNet [Amos and Kolter, 2017] introduced differentiable optimization layers based on quadratic programs. Differentiable convex optimization layers then generalized this idea to disciplined convex programs and cone programs, including SOCPs [Agrawal et al., 2019], and DiffOpt [Besançon et al., 2024] further develops this perspective through differentiable optimization via model transformations. More recently, BPQP [Pan et al., 2024] introduces a differentiable convex optimization framework that reformulates the backward pass as a decoupled quadratic program for enhanced efficiency, and comprehensive surveys have synthesized developments at the intersection of optimization theory and deep learning [Katyal, 2024]. Learning Parametric Convex Functions (PCF) has been proposed to fit disciplined-programming-compatible convex families directly from data [Schaller et al., 2025], and very recent work explores constructing convex surrogates for parametric nonconvex optimization via compositions of convex and monotonic terms [Wang et al., 2026].

These methods typically begin from a prescribed optimization template and rely on differentiating through the corresponding solution map or learning a disciplined-programming-compatible expression tied to a solver layer. In contrast, we construct a direct feed-forward architecture whose forward pass itself admits an exact SOCP value-function interpretation, thereby lifting ICNNs from polyhedral LP geometry to conic SOCP geometry while retaining fast inference.

3 Method

This section recalls the known LP representation of ReLU-ICNNs and introduces SOC-ICNN. We then rigorously expose the fundamental representational bottleneck of this LP class so that we elevate the backbone to SOCP and finally prove its exact SOCP formulation.

3.1 ReLU-ICNN as an LP value-function backbone

Given an input $\boldsymbol{x} \in \mathbb{R}^{d_0}$, a standard ReLU-ICNN is defined by $\boldsymbol{z}_\ell = \sigma(W_\ell \boldsymbol{x} + U_\ell \boldsymbol{z}_{\ell-1} + \boldsymbol{b}_\ell)$, $\ell = 1, \dots, L$, with $\boldsymbol{z}_0 = \mathbf{0}$, elementwise ReLU $\sigma(t) = \max\{t, 0\}$, and nonnegativity constraints $U_\ell \geq 0$, $\boldsymbol{c} \geq 0$. Its output is $f_{\mathrm{ReLU}}(\boldsymbol{x}) = \boldsymbol{c}^\top \boldsymbol{z}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0$. We recall the LP interpretation of ReLU-ICNN inference from Amos et al. [2017], using our notation for the subsequent SOCP lift.

Lemma 1 (Known LP value-function representation)**.** *For any input $\boldsymbol{x}$, $f_{\mathrm{ReLU}}(\boldsymbol{x})$ is exactly equal to the optimal value of*

$$\begin{aligned} f_{\mathrm{ReLU}}(\boldsymbol{x}) = \min_{\{\boldsymbol{z}_\ell\}_{\ell=1}^{L}} \quad & \boldsymbol{c}^\top \boldsymbol{z}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0 \\ \text{s.t.} \quad & \boldsymbol{z}_\ell \geq W_\ell \boldsymbol{x} + U_\ell \boldsymbol{z}_{\ell-1} + \boldsymbol{b}_\ell, \quad \boldsymbol{z}_0 = \mathbf{0}, \quad \boldsymbol{z}_\ell \geq \mathbf{0}, \quad \ell = 1, \dots, L. \end{aligned} \tag{2}$$

The forward recursion yields componentwise minimal feasible hidden states; nonnegative hidden and output weights then imply equality of the LP optimum and the forward value. Appendix A.1 records the derivation for completeness.

Lemma 1 exposes the exact optimization class realized by finite-width ReLU-ICNNs: they are LP value-function models. This observation is not merely interpretive; it identifies the source of both their strength and their bottleneck. To break the bottleneck while preserving a value-function interpretation, the natural route is to enlarge the underlying optimization class rather than only modify local nonlinearities.

3.2 Fundamental Limitations of LP Value Functions

Because the ReLU-ICNN forward pass is the value of the parametric LP in (2), its output is confined to the CPWL class. The following proposition makes this restriction explicit.

Proposition 1 (CPWL structure)**.** *Any finite-depth, finite-width ReLU-ICNN represents a function $f_{\mathrm{ReLU}} : \mathbb{R}^{d_0} \to \mathbb{R}$ that can be written exactly as a finite max-affine convex function*

$$f_{\mathrm{ReLU}}(\boldsymbol{x}) = \max_{j \in \mathcal{J}} \{\boldsymbol{a}_j^\top \boldsymbol{x} + \beta_j\},$$

where $\mathcal{J}$ is finite. In particular, f_{ReLU} is continuous, convex, and piecewise linear.

Proof sketch. Dualizing (2) yields a linear maximization problem over a bounded polyhedron, with boundedness induced by the nonnegative dual chain $0 \leq \boldsymbol{\nu}_L \leq \boldsymbol{c}$ and $0 \leq \boldsymbol{\nu}_\ell \leq U_{\ell+1}^\top \boldsymbol{\nu}_{\ell+1}$. A linear objective over a polyhedron attains its maximum at an extreme point, and finitely many extreme points yield a finite max-affine representation. Details are provided in Appendix A.3. □

Regional flat limitation. Proposition 1 implies a rigid polyhedral subgradient structure. For any input $\boldsymbol{x}$ and any $\boldsymbol{g} \in \partial f_{\mathrm{ReLU}}(\boldsymbol{x})$,

$$\boldsymbol{g} = \boldsymbol{v} + \sum_{\ell=1}^{L} W_\ell^\top \boldsymbol{\nu}_\ell, \tag{3}$$

where $\{\boldsymbol{\nu}_\ell\}$ are confined to the polyhedral constraints in (11). Thus the gradient mapping $\boldsymbol{x} \mapsto \partial f_{\mathrm{ReLU}}(\boldsymbol{x})$ is piecewise constant: the Jacobian is zero almost everywhere, and the model has no native mechanism for smooth curvature on each linear region.

This regional flatness becomes a severe quantitative bottleneck when the target has genuine curvature. The next result shows that CPWL models must spend a large number of affine pieces merely to approximate strongly convex geometry.

Proposition 2 (CPWL lower bound for strongly convex targets)**.** *Let $\Omega \subset \mathbb{R}^{d_0}$ be compact and convex with nonzero volume, and let $f : \Omega \to \mathbb{R}$ be μ-strongly convex on Ω. If a CPWL function $g(\boldsymbol{x}) = \max_{1 \leq i \leq N} \{\boldsymbol{a}_i^\top \boldsymbol{x} + b_i\}$ composed of N affine pieces satisfies $\|f - g\|_{L_\infty(\Omega)} \leq \varepsilon$, then*

$$N \geq \frac{\mathrm{vol}_{d_0}(\Omega)}{\omega_{d_0} \, 2^{d_0}} \left(\frac{\mu}{\varepsilon}\right)^{d_0/2}, \tag{4}$$

where $\omega_{d_0} = \mathrm{vol}_{d_0}(B_2^{d_0})$.

Proof sketch. Let $A_i = \{\boldsymbol{x} \in \Omega : g(\boldsymbol{x}) = \boldsymbol{a}_i^\top \boldsymbol{x} + b_i\}$ be the active region of the i-th affine piece and refine these sets into a measurable partition P_i. For any $\boldsymbol{x}, \boldsymbol{y} \in A_i$, strong convexity at the midpoint together with $\|f - g\|_{L_\infty(\Omega)} \le \varepsilon$ yields $\|\boldsymbol{x} - \boldsymbol{y}\|_2 \le 4\sqrt{\varepsilon/\mu}$. Hence each P_i has diameter at most $4\sqrt{\varepsilon/\mu}$. The isodiametric inequality gives $\mathrm{vol}(P_i) \le \omega_{d_0}(2\sqrt{\varepsilon/\mu})^{d_0}$. Summing over all pieces and rearranging yields (4). A complete proof is provided in Appendix A.4. □

This lower bound formalizes the curse of polyhedral approximation: approximating smooth strongly convex geometry to accuracy ε requires $N = \Omega(\varepsilon^{-d_0/2})$ affine pieces. For ReLU-ICNNs, this means that curvature must be simulated indirectly by proliferating linear regions. This is precisely the structural bottleneck that SOC-ICNN removes. Instead of spending polyhedral pieces to imitate curvature, we lift the value-function class from LP to SOCP and inject quadratic and conic primitives directly into the architecture.

3.3 SOC-ICNN Architecture

The fundamental limitation of LP value functions is their purely polyhedral geometry. To overcome this, we augment the standard ReLU-ICNN backbone with two explicit structural primitives. From a neural network perspective, this results in a topology where the input $\boldsymbol{x}$ is processed through three parallel computational branches before being aggregated. The same construction also extends to convolutional and finite-horizon recurrent architectures by replacing dense affine maps with structured convolutional or unrolled recurrent linear operators; see Appendices C and D.

Quadratic primitive. For each $h = 1, \dots, H$, we introduce a quadratic branch defined as

$$\frac{\alpha_h}{2} \|B_h \boldsymbol{x} + \boldsymbol{e}_h\|_2^2, \qquad \alpha_h \ge 0, \tag{5}$$

where $B_h \in \mathbb{R}^{m_h \times d_0}$ and $\boldsymbol{e}_h \in \mathbb{R}^{m_h}$. Architecturally, this branch applies a learnable affine transformation to the input, routes the resulting vector through a squared ℓ_2-norm pooling operator, and scales it by a non-negative weight α_h, which provides an explicit PSD curvature primitive.

Conic primitive. For each $g = 1, \dots, G$, we introduce a conic branch defined as:

$$\lambda_g \|A_g \boldsymbol{x} + \boldsymbol{d}_g\|_2, \qquad \lambda_g \ge 0, \tag{6}$$

where $A_g \in \mathbb{R}^{k_g \times d_0}$ and $\boldsymbol{d}_g \in \mathbb{R}^{k_g}$. Similarly, this branch projects the input via an affine layer, passes it through an un-squared standard ℓ_2-norm operator, and scales the output by λ_g. This term acts as an explicit Euclidean norm primitive.

Combining the outputs of these two parallel branches with the main ReLU backbone yields the unified SOC-ICNN forward pass:

$$f_{\mathrm{SOC}}(\boldsymbol{x}) = f_{\mathrm{ReLU}}(\boldsymbol{x}) + \sum_{h=1}^{H} \frac{\alpha_h}{2} \|B_h \boldsymbol{x} + \boldsymbol{e}_h\|_2^2 + \sum_{g=1}^{G} \lambda_g \|A_g \boldsymbol{x} + \boldsymbol{d}_g\|_2 . \tag{7}$$

The three terms in (7) play distinct and complementary geometric roles. The ReLU backbone captures the polyhedral (piecewise-linear) structure, the quadratic branch injects explicit smooth curvature, and the norm branch captures second-order conic geometry. Thus, the SOC-ICNN extends classical ReLU-ICNNs from purely polyhedral value functions into a unified class of structured convex functions encompassing piecewise-linear, quadratic, and conic components. The overall parallel topology is summarized in Figure 1.

Structural budgets. Quadratic atoms can be merged: with $\bar{B} = \mathrm{stack}_h(\sqrt{\alpha_h} B_h)$ and $\bar{e} = \mathrm{stack}_h(\sqrt{\alpha_h} e_h)$, their sum is $\frac{1}{2}\|\bar{B}x + \bar{e}\|_2^2$. Thus $r_Q = \sum_h m_h$ controls the attainable Hessian rank, at most $\min(d_0, r_Q)$; $H = 1$ is sufficient for representation. Norm atoms generally cannot be merged, so both $r_C = \sum_g k_g$ and the allocation $(k_1, \dots, k_G)$ matter. Known groups can guide this allocation; otherwise it is a validation hyperparameter. Appendix E gives controlled allocation tests, and Appendix F discusses training stability.

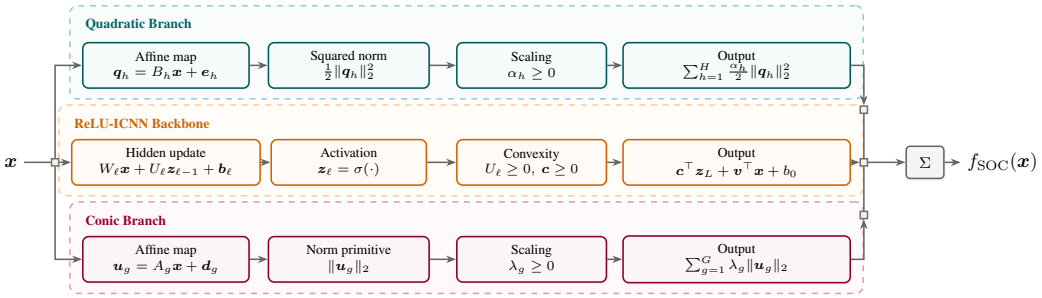

Figure 1: Unified SOC-ICNN architecture. The input $\boldsymbol{x}$ is processed by three parallel branches. The quadratic branch applies an affine map followed by a squared ℓ_2 primitive and nonnegative scaling, the ReLU-ICNN backbone captures polyhedral piecewise-linear structure under standard ICNN convexity constraints, and the conic branch applies an affine map followed by an ℓ_2-norm primitive and nonnegative scaling. Summing the three branch outputs yields the final convex surrogate $f_{\mathrm{SOC}}(\boldsymbol{x})$, which admits an exact SOCP value-function interpretation.

3.4 SOCP lifting and value-function interpretation

Both structural branches admit standard SOCP epigraph lifts. For the quadratic branch, let $\mathcal{Q}_r^{m+2} := \{(\xi, \eta, \boldsymbol{w}) \in \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}^m : 2\xi\eta \geq \|\boldsymbol{w}\|_2^2\}$ denote the rotated second-order cone. Introducing $\boldsymbol{q}_h = B_h\boldsymbol{x} + \boldsymbol{e}_h$ and $(s_h, 1, \boldsymbol{q}_h) \in \mathcal{Q}_r^{m_h+2}$ gives $s_h \geq \frac{1}{2}\|B_h\boldsymbol{x} + \boldsymbol{e}_h\|_2^2$, which can be chosen tight at an optimum for $\alpha_h \geq 0$ (and must be tight if $\alpha_h > 0$). For the norm branch, let $\mathcal{Q}^{k+1} := \{(\boldsymbol{u}, t) \in \mathbb{R}^k \times \mathbb{R} : \|\boldsymbol{u}\|_2 \leq t\}$ denote the standard second-order cone. Introducing $\boldsymbol{u}_g = A_g\boldsymbol{x} + \boldsymbol{d}_g$ and $(\boldsymbol{u}_g, t_g) \in \mathcal{Q}^{k_g+1}$ gives $t_g \geq \|A_g\boldsymbol{x} + \boldsymbol{d}_g\|_2$, which likewise admits a tight optimum for $\lambda_g \geq 0$. We now state the main structural result: the entire SOC-ICNN forward pass is exactly an SOCP value function.

Theorem 1 (SOC-ICNN as an SOCP value function)**.** *For every input $\boldsymbol{x}$, the unified architecture in* (7) *is equal to the optimal value of the following SOCP:*

$$
\begin{aligned}
f_{\mathrm{SOC}}(\boldsymbol{x}) = \min_{\eta} \quad & \boldsymbol{c}^\top \boldsymbol{z}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0 + \sum_{h=1}^{H} \alpha_h s_h + \sum_{g=1}^{G} \lambda_g t_g \\
\text{s.t.} \quad & \boldsymbol{z}_\ell \geq W_\ell \boldsymbol{x} + U_\ell \boldsymbol{z}_{\ell-1} + \boldsymbol{b}_\ell, \quad \boldsymbol{z}_0 = \boldsymbol{0}, \quad \boldsymbol{z}_\ell \geq \boldsymbol{0}, \quad \ell = 1, \ldots, L, \\
& \boldsymbol{q}_h = B_h\boldsymbol{x} + \boldsymbol{e}_h, \quad (s_h, 1, \boldsymbol{q}_h) \in \mathcal{Q}_r^{m_h+2}, \quad h = 1, \ldots, H, \\
& \boldsymbol{u}_g = A_g\boldsymbol{x} + \boldsymbol{d}_g, \quad (\boldsymbol{u}_g, t_g) \in \mathcal{Q}^{k_g+1}, \quad g = 1, \ldots, G,
\end{aligned}
\tag{8}
$$

where $\eta = \{\boldsymbol{z}_\ell, s_h, \boldsymbol{q}_h, t_g, \boldsymbol{u}_g\}$.

Proof sketch. Let $V(\boldsymbol{x})$ denote the optimal value of (8). For the upper bound, choose the ReLU forward activations, set $\boldsymbol{q}_h = B_h\boldsymbol{x} + \boldsymbol{e}_h$, $s_h = \frac{1}{2}\|\boldsymbol{q}_h\|_2^2$, $\boldsymbol{u}_g = A_g\boldsymbol{x} + \boldsymbol{d}_g$, and $t_g = \|\boldsymbol{u}_g\|_2$. This point is feasible and attains exactly the value in (7), hence $V(\boldsymbol{x}) \leq f_{\mathrm{SOC}}(\boldsymbol{x})$. For the lower bound, any feasible point satisfies the ReLU LP lower bound from Lemma 1 and the conic epigraph inequalities $s_h \geq \frac{1}{2}\|B_h\boldsymbol{x}+\boldsymbol{e}_h\|_2^2$, $t_g \geq \|A_g\boldsymbol{x}+\boldsymbol{d}_g\|_2$. Multiplying by $\alpha_h, \lambda_g \geq 0$ and summing gives $V(\boldsymbol{x}) \geq f_{\mathrm{SOC}}(\boldsymbol{x})$. Thus $V(\boldsymbol{x}) = f_{\mathrm{SOC}}(\boldsymbol{x})$. A complete derivation is given in Appendix A.2. □

Since the ReLU backbone is convex in $\boldsymbol{x}$, squared Euclidean norms and Euclidean norms composed with affine maps are convex, and nonnegative linear combinations preserve convexity, f_{SOC} remains globally convex with respect to the input.

Scope and downstream use. Theorem 1 gives an exact SOCP representation for every SOC-ICNN; it does not characterize all SOCP value functions. Once network parameters are fixed, the lift can be embedded in downstream optimization when the remaining objective and constraints are SOCP-representable. Forward evaluation itself requires no solver. These guarantees concern convexity in decision variables, not convexity of training in network parameters or recovery of an unknown system's original conic coefficients.

4 Theoretical Analysis

This section establishes the theoretical core of SOC-ICNN. After exposing the CPWL bottleneck of ReLU-ICNNs in Section 3.2, we show that SOC-ICNN breaks this bottleneck by lifting the underlying value-function class. The analysis proceeds in three steps: exact structural representation, structural absorption for residual approximation, and forward-complexity preservation.

4.1 Exact Structured Representations and Strict Extension

Having identified the CPWL bottleneck, we now show that the quadratic and conic branches do not merely improve empirical flexibility; they strictly enlarge the representable convex function class. Define the structured conic family

$$\mathcal{G}_{\mathrm{SOC}}(r,G)=\big\{\boldsymbol{a}^\top\boldsymbol{x}+b+\tfrac{1}{2}\boldsymbol{x}^\top Q\boldsymbol{x}+\sum_{g=1}^{G}\lambda_g\|A_g\boldsymbol{x}+\boldsymbol{d}_g\|_2 \mid Q\succeq 0,\ \mathrm{rank}(Q)\le r,\ \lambda_g\ge 0\big\}.$$

This class contains affine terms, PSD quadratic curvature, and Euclidean norm primitives.

Proposition 3 (Exact representation and strict extension)**.** *For any target function $f\in\mathcal{G}_{\mathrm{SOC}}(r,G)$, there exists a finite-width SOC-ICNN such that*

$$f_{\mathrm{SOC}}(\boldsymbol{x})\equiv f(\boldsymbol{x}),\qquad \forall\boldsymbol{x}\in\mathbb{R}^{d_0}.$$

Moreover, if $\mathcal{F}_{\mathrm{ReLU}}$ and $\mathcal{F}_{\mathrm{SOC}}$ denote the function classes represented by finite-width ReLU-ICNNs and SOC-ICNNs, respectively, then

$$\mathcal{F}_{\mathrm{ReLU}}\subsetneq\mathcal{F}_{\mathrm{SOC}}.$$

In particular, every non-CPWL member of $\mathcal{G}_{\mathrm{SOC}}(r,G)$, such as $f(\boldsymbol{x})=\frac{1}{2}\|\boldsymbol{x}\|_2^2$, belongs to $\mathcal{F}_{\mathrm{SOC}}\setminus\mathcal{F}_{\mathrm{ReLU}}$.

Proof sketch. Since $Q\succeq 0$ and $\mathrm{rank}(Q)\le r$, there exists B such that $Q=B^\top B$, hence $\frac{1}{2}\boldsymbol{x}^\top Q\boldsymbol{x}=\frac{1}{2}\|B\boldsymbol{x}\|_2^2$, which is exactly represented by one quadratic branch. Each norm term $\lambda_g\|A_g\boldsymbol{x}+\boldsymbol{d}_g\|_2$ is exactly represented by one conic branch. The affine residual $\boldsymbol{a}^\top\boldsymbol{x}+b$ is represented by the affine passthrough term $\boldsymbol{v}^\top\boldsymbol{x}+b_0$ in the backbone. Thus exact representation holds. Strict inclusion follows because finite-width ReLU-ICNNs are CPWL by Proposition 1, whereas $\mathcal{G}_{\mathrm{SOC}}(r,G)$ contains non-CPWL curved functions. A detailed proof is in Appendix A.5. □

4.2 Structural Absorption and Piece Efficiency

Exact representation already separates SOC-ICNN from ReLU-ICNN on purely quadratic or conic targets. The sharper point is structural absorption: when the target contains dominant curvature plus a residual term, SOC-ICNN absorbs the curvature exactly and leaves only the residual to the ReLU backbone. Consider

$$f(\boldsymbol{x})=q(\boldsymbol{x})+h(\boldsymbol{x}),\tag{9}$$

where $q\in\mathcal{G}_{\mathrm{SOC}}(r,G)$ captures the quadratic and conic structure, and h is a convex residual with $\tilde{L}$-Lipschitz gradient on a compact domain Ω, i.e.,

$$\|\nabla h(\boldsymbol{x})-\nabla h(\boldsymbol{y})\|_2\le\tilde{L}\|\boldsymbol{x}-\boldsymbol{y}\|_2,\qquad\forall\boldsymbol{x},\boldsymbol{y}\in\Omega.$$

This is a conditional decomposition: subtracting a convex structured term from a convex target need not leave a convex residual. The bound concerns affine pieces, not a fixed-depth parameter budget.

Proposition 4 (Constructive structural absorption bound)**.** *Assume that the target f admits the decomposition in* (9)*. Then for every affine-piece budget $N\ge 1$, there exists an SOC-ICNN ϕ_N whose ReLU backbone uses at most N affine pieces and satisfies*

$$\|f-\phi_N\|_{L_\infty(\Omega)}\le C_\Omega\tilde{L}\,N^{-2/d_0},\tag{10}$$

where $C_\Omega>0$ depends only on the geometry of Ω.

Proof sketch. By Proposition 3, the structured component q is represented exactly by the quadratic and conic branches. For the residual h, construct a δ-net on Ω with $\delta \asymp N^{-1/d_0}$, and place tangent hyperplanes of h at the net points. Convexity and the $\tilde{L}$-Lipschitz gradient imply a max-affine approximation g_N satisfying $\|h - g_N\|_\infty \le C_\Omega \tilde{L} N^{-2/d_0}$. The ReLU backbone realizes g_N with at most N affine pieces. Combining this backbone with the exact representation of q yields ϕ_N. A full constructive proof is in Appendix A.6. □

This result makes the structural advantage explicit. A pure ReLU-ICNN must spend affine pieces to simulate the curvature of q, incurring the CPWL lower-bound cost. SOC-ICNN bypasses this cost by representing q exactly and only approximating the smoother residual h. Thus the approximation burden is transferred from the full curved target to the residual after curvature absorption.

4.3 Forward Complexity

Finally, the representational upgrade does not compromise feed-forward inference. Let the ReLU backbone have width m and depth L. A standard forward pass costs $T_{\mathrm{ReLU}} = \Theta(Lm^2 + Ld_0m)$. Each quadratic branch adds $O(d_0 m_h)$ operations, and each conic branch adds $O(d_0 k_g)$, giving

$$T_{\mathrm{SOC}} = \Theta\left(Lm^2 + Ld_0m + d_0 \sum_{h=1}^{H} m_h + d_0 \sum_{g=1}^{G} k_g \right).$$

Under the standard bounded-branch regime $H, G = O(1)$ with $m_h, k_g = O(m)$, this reduces to

$$T_{\mathrm{SOC}} = \Theta(Lm^2 + Ld_0m),$$

the same asymptotic order as the ReLU backbone. Therefore, SOC-ICNN is a genuine structural upgrade from LP to SOCP value functions: it injects curvature and conic geometry without increasing the asymptotic order of forward-pass complexity.

5 Experiments

We evaluate whether the SOCP lift of SOC-ICNN is not only a formal reinterpretation, but a functional architectural upgrade over classical ICNNs. The experiments target three questions. First, does the feed-forward SOC-ICNN value exactly coincide with the lifted SOCP value in Theorem 1? Second, does injecting quadratic and conic primitives improve approximation efficiency under controlled parameter budgets? Third, does the improved surrogate geometry translate into better downstream decisions after optimizing the learned convex model? Table 1 summarizes the key numerical evidence, while full implementation details and tasks are deferred to Appendices B.1–B.3. Additional downstream tests for the convolutional and finite-horizon recurrent SOC-ICNN extensions are reported in Appendices C.9 and D.6, respectively, illustrating that the same LP-to-SOCP lifting principle remains applicable under spatially and temporally structured decision variables.

5.1 Experiment 1: Value-Function Equivalence

The first experiment directly tests the central structural claim: the closed-form SOC-ICNN forward pass and the lifted SOCP in (8) compute the same value. Across 150 random trials with $d_0 = 100$, width 256, depth 6, and two quadratic and conic blocks, the primal–dual gap remains at the level of 10^{-14}, and the discrepancy between the feed-forward value and the external CVXPY SOCP solver value is about 5.6×10^{-7}. All ReLU-chain, rotated-SOC, and standard-SOC feasibility residuals are at numerical precision. Thus, SOC-ICNN is numerically validated as an exact SOCP value-function architecture, not merely a convex network inspired by SOCP structure. The direct feed-forward evaluation is also 3–4$\times$ faster than solving the lifted SOCP externally. Full optimality, feasibility, and runtime diagnostics are reported in Appendix B.1.

5.2 Experiment 2: Structural Efficiency under Controlled Budgets

The second experiment tests whether the new structural primitives reduce the burden placed on the ReLU backbone. We compare SOC-ICNN with ReLU-ICNN, Softplus-ICNN, and the internal

ablations Quad and Norm under controlled parameter budgets. The benchmark contains ten convex targets spanning quadratic, norm, log-sum-exp, Softplus-sum, Huber, ℓ_1, hybrid, and ICKAN-inspired composite structures over dimensions $d \in \{5, 10, 20, 50\}$.

The aggregate $d = 50$ results in Table 1 show a sharp separation. SOC-ICNN achieves the best relative error on all ten targets, with mean error 0.099, compared with 0.772 for ReLU-ICNN and 0.763 for Softplus-ICNN. The gain is especially meaningful because SOC-ICNN uses fewer parameters than these two baselines at $d = 50$. Quad-ICNN is the strongest single-primitive ablation, confirming that explicit PSD curvature already removes a large part of the CPWL burden. The full SOC model further improves the average error by adding norm-based conic primitives. These results support the theoretical message of Sections 4.1–4.2: curvature should not be simulated by proliferating polyhedral pieces when it can be represented as a native structural primitive. Complete per-function and per-dimension results are provided in Appendix B.2.

5.3 Experiment 3: Downstream Decision Quality

The third experiment examines whether better convex surrogate geometry leads to better optimized decisions. We evaluate six parametric convex optimization tasks from SOCP, logistic, log-sum-exp, and Huber families under simplex, box, and capped-simplex constraints. The baselines include ReLU-ICNN, Softplus-ICNN, Norm, Quad, SOC, and two optimization-based models, DCP and PCF [Agrawal et al., 2019, Besançon et al., 2024, Schaller et al., 2025].

The downstream results reveal a clear separation between pointwise fitting accuracy and decision quality. PCF remains the strongest optimization-based baseline on average, achieving the lowest mean regret in the $d = 10$ summary. Within the neural surrogate family, Quad is the most robust variant, reducing mean regret from 0.234 for ReLU-ICNN and 0.507 for Softplus-ICNN to 0.179. SOC-ICNN remains competitive, but the full conic branch is not uniformly beneficial across all downstream tasks. These results show that smaller average pointwise fitting error need not imply smaller decision regret; errors near the surrogate minimizer also matter. The Norm branch is not uniformly advantageous in this regime. The original objectives share a quadratic–linear component, while Appendix G tests objectives with explicit norm-type structure. Complete regret, decision-error, prediction-error, and runtime tables are given in Appendix B.3.

Table 1: Main experimental summary. Exp. 1 validates the SOCP value-function identity using the representative no-passthrough setting; the passthrough setting gives comparable precision and is reported in Appendix B.1. Exp. 2 reports aggregate approximation performance at $d = 50$ over ten convex targets. Exp. 3 reports average downstream regret at $d = 10$ over six parametric optimization tasks.

Experiment	Method / metric	Main score	Dispersion	Wins	Cost / size
Exp. 1	Primal–dual gap	1.06×10^{-14}	max 4.26×10^{-14}	–	–
	Forward–solver error	5.57×10^{-7}	max 7.68×10^{-7}	–	–
	Closed-form forward	6.22 ms	$\pm$0.84 ms	–	–
	CVXPY solver	24.80 ms	$\pm$2.55 ms	–	–
Exp. 2	ReLU-ICNN	0.772	median 0.841	0/10	9,683 params
	Softplus-ICNN	0.763	median 0.831	0/10	9,683 params
	Quad-ICNN	0.127	median 0.098	0/10	9,528 params
	Norm-ICNN	0.611	median 0.721	0/10	9,578 params
	SOC-ICNN	**0.099**	median **0.074**	**10/10**	9,423 params
Exp. 3	DCP	0.284	median 0.280	0/6	solver layer
	PCF	**0.139**	median **0.132**	**4/6**	solver-compatible
	ReLU-ICNN	0.234	median 0.230	1/6	neural
	Softplus-ICNN	0.507	median 0.512	0/6	neural
	Norm-ICNN	0.229	median 0.226	0/6	neural
	Quad-ICNN	0.179	median 0.177	1/6	neural
	SOC-ICNN	0.197	median 0.201	0/6	neural

5.4 Problem-Specific Evaluation: Optimal Transport

To test a standard use of convex potentials, we adapt the Gaussian-to-eight-Gaussians benchmark of Makkuva et al. [2020]. With the same minimax framework and nearly matched parameter counts, SOC obtains the lowest mean distribution discrepancy among the tested ICNNs (Table 2); its mean Sliced-W2 is about 42% below ReLU. The three-seed variability and the remaining gap to the target-sampling floor preclude a claim of universal superiority. Appendix H reports the protocol and full results.

Table 2: Eight-mode OT: mean $\pm$ standard deviation over three seeds. Both metrics are lower-is-better.

Model	Sliced-W2	MMD^2
ReLU	0.627 ± 0.306	0.0228 ± 0.0286
Quad	0.563 ± 0.388	0.0212 ± 0.0344
Norm	0.437 ± 0.096	0.0051 ± 0.0038
SOC	$\mathbf{0.363} \pm 0.069$	$\mathbf{0.0022} \pm 0.0019$

Controlled Brenier-map recovery further isolates the benefit of matching a norm-type component (Appendix H.1). Public-data group-regularized logistic regression and robust convex regression provide complementary decision tests (Appendix G). Across these settings, Quad supplies general quadratic curvature and Norm adds task-dependent geometry. The nonconvex-target comparison in Appendix I delineates the boundary: an unconstrained MLP attains lower prediction errors in all tested settings, including the convex control.

6 Conclusion and Discussion

SOC-ICNN augments a ReLU-ICNN with quadratic and norm primitives, giving a strict extension of its finite-network function class and an explicit SOCP lift. The structural overhead preserves the backbone's asymptotic forward cost under proportional structural-width budgets. Experiments support geometry-dependent gains in approximation, optimal transport, and downstream decisions.

Limitations. The intended setting has known or required convexity in designated decision variables. SOC-ICNN cannot represent negative curvature, is not claimed to cover all SOCP value functions, and does not guarantee better fitted models at a matched parameter budget. Quad remains a strong simpler alternative; Norm allocation and its benefit depend on target geometry. The structural-absorption result assumes a convex smooth residual. The reported three-seed tests do not establish statistical significance, and direct structural branches do not eliminate gradient attenuation in a very deep backbone. The structured extensions concern finite-dimensional inputs and finite recurrent horizons. If the analytical objective is already known, directly optimizing it may be preferable to learning a surrogate.

Broader impact. Convex surrogates can support transparent optimization-based decisions, but global optimization of a learned surrogate does not certify the quality or safety of the resulting real-world decision. Applications require validation of surrogate error, constraints, and domain assumptions.

Funding and Competing Interests

This research was supported in part by the National Key Research and Development Program of China under Grant 2025YFA1017000. The authors declare no competing interests.

Use of large language models. Large language models were used for language polishing and manuscript editing. They are not a component of the proposed model or experimental pipeline. The authors take responsibility for the scientific content and the final manuscript.

References

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, December 2019.

Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proc. 34th Int. Conf. Mach. Learn.*, pages 136–145, Sydney, NSW, Australia, August 2017.

Brandon Amos, Lei Xu, and J. Zico Kolter. Input convex neural networks. In *Proc. 34th Int. Conf. Mach. Learn.*, pages 146–155, Sydney, NSW, Australia, August 2017.

Egor Bakaev, Florestan Brunck, Christoph Hertrich, Daniel Reichman, and Amir Yehudayoff. On the depth of monotone ReLU neural networks and ICNNs. *arXiv preprint arXiv:2505.06169*, 2025.

Mathieu Besançon, Joaquim Dias Garcia, Benoît Legat, and Akshay Sharma. Flexible differentiable optimization via model transformations. *INFORMS J. Comput.*, 36(2):456–478, 2024.

Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal control via neural networks: A convex approach. In *Proc. Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019.

Nicolas Christianson, Wenqi Cui, Steven Low, Weiwei Yang, and Baosen Zhang. Fast and reliable N-k contingency screening with input-convex neural networks. In *Proc. 7th Annu. Learn. Dyn. Control Conf.*, Proceedings of Machine Learning Research, pages 527–539, Ann Arbor, MI, USA, June 2025.

Thomas Deschatre and Xavier Warin. Input convex kolmogorov arnold networks. *arXiv preprint arXiv:2505.21208*, 2025.

Anne Gagneux, Mathurin Massias, Emmanuel Soubies, and Rémi Gribonval. Convexity in ReLU neural networks: Beyond ICNNs? *J. Math. Imaging Vis.*, 67(4):40, 2025.

Pieter-Jan Hoedt and Günter Klambauer. Principled weight initialisation for input-convex neural networks. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 46093–46104, New Orleans, LA, USA, December 2023.

Calder Katyal. Differentiable convex optimization layers in neural architectures: Foundations and perspectives. *arXiv preprint arXiv:2412.20679*, 2024.

Yu Liu, Fabricio Oliveira, and Jan Kronqvist. ICNN-enhanced 2SP: Leveraging input convex neural networks for solving two-stage stochastic programming. *arXiv preprint arXiv:2505.05261*, 2025.

Ashok Vardhan Makkuva, Amirhossein Taghvaei, Sewoong Oh, and Jason D. Lee. Optimal transport mapping via input convex neural networks. In *Proc. 37th Int. Conf. Mach. Learn.*, pages 6672–6681, Virtual Event, July 2020.

Jianming Pan, Zeqi Ye, Xiao Yang, Xu Yang, Weiqing Liu, Lewen Wang, and Jiang Bian. BPQP: A differentiable convex optimization framework for efficient end-to-end learning. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 77468–77493, Vancouver, BC, Canada, December 2024.

Andrew Rosemberg, Mathieu Tanneau, Bruno Fanzeres, Joaquim Garcia, and Pascal Van Hentenryck. Learning optimal power flow value functions with input-convex neural networks. *Electric Power Systems Research*, 235:110643, October 2024.

Maximilian Schaller, Alberto Bemporad, and Stephen Boyd. Learning parametric convex functions. *arXiv preprint arXiv:2506.04183*, 2025.

Renzi Wang, Panagiotis Patrinos, and Alberto Bemporad. Parametric nonconvex optimization via convex surrogates. *arXiv preprint arXiv:2604.05640*, 2026.

Zihao Wang, Yuhan Li, Yao Shi, and Zhe Wu. Input convex lipschitz recurrent neural networks for robust and efficient process modeling and optimization. *arXiv preprint arXiv:2401.07494*, 2024.

Zihao Wang, Donghan Yu, and Zhe Wu. Real-time machine-learning-based optimization using input convex long short-term memory network. *Applied Energy*, 377:124472, January 2025.

Appendix Roadmap

This appendix provides the complete supplementary material supporting the main text. Appendix A contains the full derivations and proofs for the LP value-function form of ReLU-ICNNs, the SOCP value-function representation of SOC-ICNNs, the CPWL structure of finite-width ReLU-ICNNs, the approximation lower bound, the strict extension result, the structural absorption bound, and the forward-complexity calculation. Appendices B.1–B.3 provide full experimental details and per-task numerical results for the three main experiments summarized in Section 5. Appendix C presents the convolutional SOC-ICNN extension, including convexity and SOCP value-function arguments, while Appendix D gives the corresponding finite-horizon recurrent extension. Appendices C.9 and D.6 report additional downstream tests for the convolutional and recurrent extensions on spatially and temporally structured decision problems. Appendices E–I report structural-allocation and stability diagnostics, problem-specific decision and transport experiments, and nonconvex model mismatch. The appendix concludes with the NeurIPS checklist.

A Additional Derivations and Proofs

This appendix provides the complete derivations required by the main text. The presentation follows the main storyline of the paper. We first prove the LP value-function form of ReLU-ICNNs, then establish the main theorem showing that SOC-ICNN is an SOCP value function. We next prove the theoretical statements used in Section 4: the CPWL structure of finite-width ReLU-ICNNs, the lower bound for approximating strongly convex targets, the exact representability and strict extension result for SOC-ICNNs, the structural absorption bound, and finally the complexity calculation.

A.1 Proof of the LP value-function form of ReLU-ICNN

Let

$$\bar{\boldsymbol{z}}_\ell = \sigma(W_\ell \boldsymbol{x} + U_\ell \bar{\boldsymbol{z}}_{\ell-1} + \boldsymbol{b}_\ell)\,, \qquad \ell = 1, \dots, L,$$

denote the hidden states produced by the forward pass, with $\bar{\boldsymbol{z}}_0 = \mathbf{0}$. Since ReLU satisfies $\sigma(a) = \max\{a, 0\}$ elementwise, for every layer ℓ we have

$$\bar{\boldsymbol{z}}_\ell \geq W_\ell \boldsymbol{x} + U_\ell \bar{\boldsymbol{z}}_{\ell-1} + \boldsymbol{b}_\ell, \qquad \bar{\boldsymbol{z}}_\ell \geq \mathbf{0}.$$

Hence $\{\bar{\boldsymbol{z}}_\ell\}_{\ell=1}^L$ is feasible for (2), and therefore

$$\operatorname{val}(2) \leq \boldsymbol{c}^\top \bar{\boldsymbol{z}}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0 = f_{\mathrm{ReLU}}(\boldsymbol{x}).$$

To prove the reverse inequality, take any feasible point $\{\boldsymbol{z}_\ell\}_{\ell=1}^L$ of (2). We show by induction that

$$\boldsymbol{z}_\ell \geq \bar{\boldsymbol{z}}_\ell, \ \ \forall \ell = 1, \dots, L.$$

For $\ell = 1$, feasibility gives

$$\boldsymbol{z}_1 \geq W_1 \boldsymbol{x} + \boldsymbol{b}_1, \qquad \boldsymbol{z}_1 \geq \mathbf{0},$$

hence

$$\boldsymbol{z}_1 \geq \max\{W_1 \boldsymbol{x} + \boldsymbol{b}_1, \mathbf{0}\} = \bar{\boldsymbol{z}}_1.$$

Assume now that $\boldsymbol{z}_k \geq \bar{\boldsymbol{z}}_k$ for some $k \in \{1, \dots, L-1\}$. Since $U_{k+1} \geq 0$ elementwise, monotonicity of matrix multiplication implies

$$U_{k+1} \boldsymbol{z}_k \geq U_{k+1} \bar{\boldsymbol{z}}_k.$$

Using feasibility once more,

$$\boldsymbol{z}_{k+1} \geq W_{k+1} \boldsymbol{x} + U_{k+1} \boldsymbol{z}_k + \boldsymbol{b}_{k+1} \geq W_{k+1} \boldsymbol{x} + U_{k+1} \bar{\boldsymbol{z}}_k + \boldsymbol{b}_{k+1},$$

and $\boldsymbol{z}_{k+1} \geq \mathbf{0}$. Therefore

$$\boldsymbol{z}_{k+1} \geq \max\{W_{k+1} \boldsymbol{x} + U_{k+1} \bar{\boldsymbol{z}}_k + \boldsymbol{b}_{k+1}, \mathbf{0}\} = \bar{\boldsymbol{z}}_{k+1}.$$

The induction is complete.

In particular, $\boldsymbol{z}_L \geq \bar{\boldsymbol{z}}_L$. Since $\boldsymbol{c} \geq 0$, we obtain

$$\boldsymbol{c}^\top \boldsymbol{z}_L \geq \boldsymbol{c}^\top \bar{\boldsymbol{z}}_L.$$

Hence every feasible point of (2) has objective value at least $f_{\mathrm{ReLU}}(\boldsymbol{x})$, which implies

$$\operatorname{val}(2) \geq f_{\mathrm{ReLU}}(\boldsymbol{x}).$$

Combining both inequalities yields

$$\operatorname{val}(2) = f_{\mathrm{ReLU}}(\boldsymbol{x}).$$

A.2 Proof of Theorem 1

Let $V(\boldsymbol{x})$ denote the optimal value of the SOCP in (8). We prove that

$$V(\boldsymbol{x}) = f_{\mathrm{SOC}}(\boldsymbol{x}).$$

We first show that $V(\boldsymbol{x}) \le f_{\mathrm{SOC}}(\boldsymbol{x})$. By the forward definition of the ReLU backbone, there exist hidden variables $\bar{\boldsymbol{z}}_1, \dots, \bar{\boldsymbol{z}}_L$ such that

$$\boldsymbol{c}^\top \bar{\boldsymbol{z}}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0 = f_{\mathrm{ReLU}}(\boldsymbol{x}).$$

For each quadratic branch h, define

$$\bar{\boldsymbol{q}}_h = B_h \boldsymbol{x} + \boldsymbol{e}_h, \qquad \bar{s}_h = \frac{1}{2}\|B_h \boldsymbol{x} + \boldsymbol{e}_h\|_2^2.$$

Then $(\bar{s}_h, 1, \bar{\boldsymbol{q}}_h) \in \mathcal{Q}_r^{m_h+2}$. For each conic branch g, define

$$\bar{\boldsymbol{u}}_g = A_g \boldsymbol{x} + \boldsymbol{d}_g, \qquad \bar{t}_g = \|A_g \boldsymbol{x} + \boldsymbol{d}_g\|_2,$$

so that $(\bar{\boldsymbol{u}}_g, \bar{t}_g) \in \mathcal{Q}^{k_g+1}$. Therefore,

$$\left(\{\bar{\boldsymbol{z}}_\ell\}_{\ell=1}^L, \{\bar{s}_h, \bar{\boldsymbol{q}}_h\}_{h=1}^H, \{\bar{t}_g, \bar{\boldsymbol{u}}_g\}_{g=1}^G\right)$$

is feasible for (8). Its objective value is

$$\boldsymbol{c}^\top \bar{\boldsymbol{z}}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0 + \sum_{h=1}^H \alpha_h \bar{s}_h + \sum_{g=1}^G \lambda_g \bar{t}_g,$$

which equals

$$f_{\mathrm{ReLU}}(\boldsymbol{x}) + \sum_{h=1}^H \frac{\alpha_h}{2}\|B_h \boldsymbol{x} + \boldsymbol{e}_h\|_2^2 + \sum_{g=1}^G \lambda_g \|A_g \boldsymbol{x} + \boldsymbol{d}_g\|_2 = f_{\mathrm{SOC}}(\boldsymbol{x}).$$

Hence

$$V(\boldsymbol{x}) \le f_{\mathrm{SOC}}(\boldsymbol{x}).$$

We now prove the reverse inequality. Take any feasible point of (8). By Lemma 1, the ReLU constraints imply

$$\boldsymbol{c}^\top \boldsymbol{z}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0 \ge f_{\mathrm{ReLU}}(\boldsymbol{x}).$$

For the quadratic branches, feasibility of $(s_h, 1, \boldsymbol{q}_h) \in \mathcal{Q}_r^{m_h+2}$ together with $\boldsymbol{q}_h = B_h \boldsymbol{x} + \boldsymbol{e}_h$ implies

$$s_h \ge \frac{1}{2}\|B_h \boldsymbol{x} + \boldsymbol{e}_h\|_2^2.$$

Multiplying by $\alpha_h \ge 0$ and summing over h gives

$$\sum_{h=1}^H \alpha_h s_h \ge \sum_{h=1}^H \frac{\alpha_h}{2}\|B_h \boldsymbol{x} + \boldsymbol{e}_h\|_2^2.$$

Similarly, feasibility of $(\boldsymbol{u}_g, t_g) \in \mathcal{Q}^{k_g+1}$ and $\boldsymbol{u}_g = A_g \boldsymbol{x} + \boldsymbol{d}_g$ implies

$$t_g \ge \|A_g \boldsymbol{x} + \boldsymbol{d}_g\|_2,$$

and therefore

$$\sum_{g=1}^G \lambda_g t_g \ge \sum_{g=1}^G \lambda_g \|A_g \boldsymbol{x} + \boldsymbol{d}_g\|_2.$$

Adding the three parts yields

$$\boldsymbol{c}^\top \boldsymbol{z}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0 + \sum_{h=1}^H \alpha_h s_h + \sum_{g=1}^G \lambda_g t_g \ge f_{\mathrm{SOC}}(\boldsymbol{x}).$$

Thus

$$V(\boldsymbol{x}) \ge f_{\mathrm{SOC}}(\boldsymbol{x}).$$

Combining both inequalities proves

$$V(\boldsymbol{x}) = f_{\mathrm{SOC}}(\boldsymbol{x}).$$

A.3 Proof of Proposition 1

Starting from (2), we dualize the LP, introduce dual variables $\boldsymbol{\nu}_\ell \geq 0$ for the constraints

$$\boldsymbol{z}_\ell - U_\ell \boldsymbol{z}_{\ell-1} - W_\ell \boldsymbol{x} - \boldsymbol{b}_\ell \geq \mathbf{0},$$

and $\boldsymbol{\mu}_\ell \geq 0$ for the constraints $\boldsymbol{z}_\ell \geq \mathbf{0}$. The Lagrangian is

$$\begin{aligned}\mathcal{L} &= \boldsymbol{c}^\top \boldsymbol{z}_L + \boldsymbol{v}^\top \boldsymbol{x} + b_0 - \sum_{\ell=1}^{L} \boldsymbol{\nu}_\ell^\top (\boldsymbol{z}_\ell - U_\ell \boldsymbol{z}_{\ell-1} - W_\ell \boldsymbol{x} - \boldsymbol{b}_\ell) - \sum_{\ell=1}^{L} \boldsymbol{\mu}_\ell^\top \boldsymbol{z}_\ell \\ &= \boldsymbol{v}^\top \boldsymbol{x} + b_0 + \sum_{\ell=1}^{L} \boldsymbol{\nu}_\ell^\top (W_\ell \boldsymbol{x} + \boldsymbol{b}_\ell) + \boldsymbol{z}_L^\top (\boldsymbol{c} - \boldsymbol{\nu}_L - \boldsymbol{\mu}_L) + \sum_{\ell=1}^{L-1} \boldsymbol{z}_\ell^\top \big(U_{\ell+1}^\top \boldsymbol{\nu}_{\ell+1} - \boldsymbol{\nu}_\ell - \boldsymbol{\mu}_\ell\big).\end{aligned}$$

To have a finite lower bound when minimizing over $\boldsymbol{z}_\ell$, the coefficients of $\boldsymbol{z}_\ell$ must vanish. This gives the stationarity conditions

$$\boldsymbol{c} - \boldsymbol{\nu}_L - \boldsymbol{\mu}_L = \mathbf{0}, \qquad U_{\ell+1}^\top \boldsymbol{\nu}_{\ell+1} - \boldsymbol{\nu}_\ell - \boldsymbol{\mu}_\ell = \mathbf{0}, \quad \ell = 1, \ldots, L-1.$$

Since $\boldsymbol{\mu}_\ell \geq 0$, these are equivalent to the chain constraints

$$0 \leq \boldsymbol{\nu}_L \leq \boldsymbol{c}, \qquad 0 \leq \boldsymbol{\nu}_\ell \leq U_{\ell+1}^\top \boldsymbol{\nu}_{\ell+1}, \quad \ell = 1, \ldots, L-1. \tag{11}$$

Because each $U_{\ell+1}$ has nonnegative entries, the chain constraints imply that each $\boldsymbol{\nu}_\ell$ is bounded. Hence the dual feasible region $\mathcal{N}$ defined by (11) is a nonempty bounded polyhedron.

The dual problem therefore reads

$$f_{\mathrm{ReLU}}(\boldsymbol{x}) = \max_{\{\boldsymbol{\nu}_\ell\} \in \mathcal{N}} \left[\boldsymbol{v}^\top \boldsymbol{x} + b_0 + \sum_{\ell=1}^{L} \boldsymbol{\nu}_\ell^\top (W_\ell \boldsymbol{x} + \boldsymbol{b}_\ell) \right].$$

Since a linear function over a polyhedron attains its maximum at an extreme point, and the number of extreme points of $\mathcal{N}$ is finite, there exists a finite index set $\mathcal{J}$ such that

$$f_{\mathrm{ReLU}}(\boldsymbol{x}) = \max_{j \in \mathcal{J}} \left[\left(\boldsymbol{v} + \sum_{\ell=1}^{L} W_\ell^\top \boldsymbol{\nu}_\ell^{(j)} \right)^\top \boldsymbol{x} + \left(b_0 + \sum_{\ell=1}^{L} (\boldsymbol{\nu}_\ell^{(j)})^\top \boldsymbol{b}_\ell \right) \right].$$

Defining

$$\boldsymbol{a}_j = \boldsymbol{v} + \sum_{\ell=1}^{L} W_\ell^\top \boldsymbol{\nu}_\ell^{(j)}, \qquad \beta_j = b_0 + \sum_{\ell=1}^{L} (\boldsymbol{\nu}_\ell^{(j)})^\top \boldsymbol{b}_\ell,$$

we obtain the finite max-affine form

$$f_{\mathrm{ReLU}}(\boldsymbol{x}) = \max_{j \in \mathcal{J}} \{\boldsymbol{a}_j^\top \boldsymbol{x} + \beta_j\}.$$

The subgradient expression in (3) follows immediately from the standard subdifferential formula for max-affine functions: every subgradient is a convex combination of active slopes, hence can still be written in the form

$$\boldsymbol{g} = \boldsymbol{v} + \sum_{\ell=1}^{L} W_\ell^\top \boldsymbol{\nu}_\ell,$$

with $\{\boldsymbol{\nu}_\ell\}$ satisfying the same chain constraints (11).

A.4 Proof of Proposition 2

Let

$$g(\boldsymbol{x}) = \max_{1 \leq i \leq N} \{\boldsymbol{a}_i^\top \boldsymbol{x} + b_i\},$$

and define the active region of the ith affine piece by

$$A_i = \left\{ \boldsymbol{x} \in \Omega \;\middle|\; g(\boldsymbol{x}) = \boldsymbol{a}_i^\top \boldsymbol{x} + b_i \right\}.$$

Since g and each $\boldsymbol{a}_i^\top \boldsymbol{x} + b_i$ are continuous, each A_i is a closed set, hence Lebesgue measurable. Clearly $\Omega = \bigcup_{i=1}^N A_i$.

Fix one active region A_i, and take any $\boldsymbol{x}, \boldsymbol{y} \in A_i$. Let

$$\boldsymbol{m} = \frac{\boldsymbol{x} + \boldsymbol{y}}{2}.$$

Because $\boldsymbol{x}, \boldsymbol{y}$ lie on the same active affine piece, and $\|f - g\|_{L_\infty(\Omega)} \le \varepsilon$, we have

$$|f(\boldsymbol{x}) - \ell_i(\boldsymbol{x})| \le \varepsilon, \qquad |f(\boldsymbol{y}) - \ell_i(\boldsymbol{y})| \le \varepsilon, \qquad |f(\boldsymbol{m}) - g(\boldsymbol{m})| \le \varepsilon,$$

where $\ell_i(\boldsymbol{x}) = \boldsymbol{a}_i^\top \boldsymbol{x} + b_i$. By μ-strong convexity,

$$f(\boldsymbol{m}) \le \frac{f(\boldsymbol{x}) + f(\boldsymbol{y})}{2} - \frac{\mu}{8}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2.$$

Substituting the uniform error bounds and using the fact that ℓ_i is affine yields

$$\frac{\mu}{8}\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \le 2\varepsilon.$$

Therefore

$$\|\boldsymbol{x} - \boldsymbol{y}\|_2 \le 4\sqrt{\frac{\varepsilon}{\mu}}.$$

Hence every active region has diameter at most $4\sqrt{\varepsilon/\mu}$.

Applying the isodiametric inequality to each A_i gives

$$\mathrm{vol}_{d_0}(A_i) \le \omega_{d_0} \left(\frac{\mathrm{diam}(A_i)}{2}\right)^{d_0} \le \omega_{d_0} \left(2\sqrt{\frac{\varepsilon}{\mu}}\right)^{d_0}.$$

Since $\Omega \subseteq \bigcup_{i=1}^N A_i$, the subadditivity of Lebesgue measure implies

$$\mathrm{vol}_{d_0}(\Omega) \le \sum_{i=1}^N \mathrm{vol}_{d_0}(A_i) \le N\, \omega_{d_0} \left(2\sqrt{\frac{\varepsilon}{\mu}}\right)^{d_0},$$

which rearranges to

$$N \ge \frac{\mathrm{vol}_{d_0}(\Omega)}{\omega_{d_0}\, 2^{d_0}} \left(\frac{\mu}{\varepsilon}\right)^{d_0/2}.$$

A.5 Proof of Proposition 3

Take any

$$f(\boldsymbol{x}) = \boldsymbol{a}^\top \boldsymbol{x} + b + \frac{1}{2}\boldsymbol{x}^\top Q \boldsymbol{x} + \sum_{g=1}^G \lambda_g \|A_g \boldsymbol{x} + \boldsymbol{d}_g\|_2 \in \mathcal{G}_{\mathrm{SOC}}(r, G).$$

Since $Q \succeq 0$ and $\mathrm{rank}(Q) \le r$, there exists a matrix $B \in \mathbb{R}^{r \times d_0}$ such that $Q = B^\top B$, and hence

$$\frac{1}{2}\boldsymbol{x}^\top Q \boldsymbol{x} = \frac{1}{2}\|B\boldsymbol{x}\|_2^2.$$

Therefore: (i) the affine term $\boldsymbol{a}^\top \boldsymbol{x} + b$ is represented by the affine part of the ReLU backbone (by setting the hidden layers to zero depth); (ii) the quadratic term is represented exactly by one quadratic branch; (iii) each norm term $\lambda_g \|A_g \boldsymbol{x} + \boldsymbol{d}_g\|_2$ is represented exactly by one conic branch. This yields an exact SOC-ICNN representation of f.

To prove strict inclusion, note first that $\mathcal{F}_{\mathrm{ReLU}} \subseteq \mathcal{F}_{\mathrm{SOC}}$ is immediate by setting $H = 0$ and $G = 0$. On the other hand, Proposition 1 shows that every finite-width ReLU-ICNN is CPWL, whereas $\mathcal{G}_{\mathrm{SOC}}(r, G)$ contains many non-CPWL functions, such as nondegenerate PSD quadratic forms (e.g., $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}\|_2^2$) and nondegenerate Euclidean norm compositions. Since these functions are representable by SOC-ICNN but not by finite-width ReLU-ICNN, we conclude that

$$\mathcal{F}_{\mathrm{ReLU}} \subsetneq \mathcal{F}_{\mathrm{SOC}}.$$

A.6 Proof of Proposition 4

Assume

$$f(\boldsymbol{x}) = q(\boldsymbol{x}) + h(\boldsymbol{x}), \qquad q \in \mathcal{G}_{\mathrm{SOC}}(r, G),$$

and that ∇h is $\tilde{L}$-Lipschitz on a compact convex domain $\Omega \subset \mathbb{R}^{d_0}$. By Proposition 3, the structured part q is represented exactly by the SOC branches.

Fix an integer $N \geq 1$. Let $\delta = c_\Omega N^{-1/d_0}$ where c_Ω is a constant depending only on Ω such that the covering number of Ω by Euclidean balls of radius δ satisfies $M \leq N$. Here c_Ω is chosen sufficiently large, with dependence on dimension and the geometry of Ω, and the centers are chosen in Ω. To see existence, enclose Ω in a cube of side D, divide each axis into $k = \lfloor N^{1/d_0} \rfloor$ intervals, and select a point of Ω in each occupied subcube. There are at most $k^{d_0} \leq N$ centers, and the covering radius is at most $\sqrt{d_0}D/k \leq 2\sqrt{d_0}DN^{-1/d_0}$. Let $\mathcal{X}_N = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ be the centers of such a covering, so $M \leq N$ and every point in Ω is within distance δ of some $\boldsymbol{x}_i$.

For each net point $\boldsymbol{x}_i$, define the tangent plane

$$\ell_i(\boldsymbol{x}) = h(\boldsymbol{x}_i) + \nabla h(\boldsymbol{x}_i)^\top (\boldsymbol{x} - \boldsymbol{x}_i),$$

and let

$$g_N(\boldsymbol{x}) = \max_{1 \leq i \leq M} \ell_i(\boldsymbol{x}).$$

Since h is convex, each tangent plane is a global lower support, so $g_N(\boldsymbol{x}) \leq h(\boldsymbol{x})$ for all $\boldsymbol{x} \in \Omega$.

By gradient Lipschitz continuity, the standard descent lemma gives

$$h(\boldsymbol{x}) \leq \ell_i(\boldsymbol{x}) + \frac{\tilde{L}}{2}\|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2, \qquad \forall \boldsymbol{x} \in \Omega.$$

For any $\boldsymbol{x} \in \Omega$, choose a net point $\boldsymbol{x}_i$ with $\|\boldsymbol{x} - \boldsymbol{x}_i\|_2 \leq \delta$. Then

$$h(\boldsymbol{x}) - g_N(\boldsymbol{x}) \leq h(\boldsymbol{x}) - \ell_i(\boldsymbol{x}) \leq \frac{\tilde{L}}{2}\delta^2 \leq \frac{\tilde{L}}{2} c_\Omega^2 N^{-2/d_0}.$$

Therefore,

$$\|h - g_N\|_{L_\infty(\Omega)} \leq C_\Omega \tilde{L}\, N^{-2/d_0},$$

where $C_\Omega = c_\Omega^2/2$ depends only on Ω.

Finally, g_N is a finite max-affine convex function. For $M = 1$ it is affine. For $M \geq 2$, an explicit ICNN construction is

$$z_1 = \mathrm{ReLU}(\ell_1 - \ell_2), \qquad z_j = \mathrm{ReLU}(z_{j-1} + \ell_j - \ell_{j+1}), \quad 2 \leq j \leq M-1.$$

Induction gives $z_j + \ell_{j+1} = \max_{i \leq j+1} \ell_i$, so the output $z_{M-1} + \ell_M$ equals g_N. Hidden-to-hidden weights are 1, and the affine passthroughs are unconstrained, as required. This construction uses $M - 1$ scalar hidden layers; it proves existence, not a fixed-depth size bound. Adding this ReLU backbone to the exact SOC representation of q gives an SOC-ICNN ϕ_N such that

$$\|f - \phi_N\|_{L_\infty(\Omega)} \leq C_\Omega \tilde{L}\, N^{-2/d_0}.$$

A.7 Proof of the complexity formulas

Assume the ReLU backbone has width m in each of its L layers. Each layer computes two dominant matrix-vector products: $U_\ell \boldsymbol{z}_{\ell-1}$ with cost $\Theta(m^2)$, and $W_\ell \boldsymbol{x}$ with cost $\Theta(d_0 m)$. Summing over all layers gives

$$T_{\mathrm{ReLU}} = \Theta(Lm^2 + Ld_0 m).$$

For the quadratic branches, the dominant cost of the hth branch is the affine map $B_h \boldsymbol{x} + \boldsymbol{e}_h$, which costs $\Theta(d_0 m_h)$. Summing over $h = 1, \ldots, H$ yields

$$\Delta T_{\mathrm{Quad}} = \Theta\left(d_0 \sum_{h=1}^{H} m_h\right).$$

For the conic branches, the dominant cost of the gth branch is the affine map $A_g\boldsymbol{x} + \boldsymbol{d}_g$, which costs $\Theta(d_0 k_g)$. Summing over $g = 1, \ldots, G$ yields

$$\Delta T_{\mathrm{SOC}} = \Theta\left(d_0 \sum_{g=1}^{G} k_g \right).$$

Combining all terms gives the overall SOC-ICNN forward complexity

$$T_{\mathrm{SOC}} = \Theta\left(Lm^2 + Ld_0 m + d_0 \sum_{h=1}^{H} m_h + d_0 \sum_{g=1}^{G} k_g \right).$$

Under the common regime $H, G = O(1)$ and $m_h, k_g = O(m)$, this simplifies to the same asymptotic order as the backbone alone.

B Experiment Results

B.1 Full Results for Experiment 1

B.1.1 Setting

Table 3 reports the complete diagnostic metrics for the value-function equivalence experiment. Across 150 trials with both passthrough settings, the closed-form forward pass and the external CVXPY solver agree to high precision. We detail the meaning of each metric below:

- **Primal-Dual Gap**: The difference between the closed-form forward value (primal) and the explicit dual objective evaluated using the extracted dual variables. A gap of $\sim 10^{-14}$ confirms strong duality and numerical optimality.
- **Forward vs. Solver Abs. Error**: The absolute difference $|f_{\mathrm{SOC}}(\boldsymbol{x}) - V_{\mathrm{CVXPY}}(\boldsymbol{x})|$. Values $< 10^{-6}$ verify exact equivalence.
- **ReLU Primal Violation**: The maximum violation of the ReLU primal constraints $\boldsymbol{z}_\ell \geq W_\ell \boldsymbol{x} + U_\ell \boldsymbol{z}_{\ell-1} + \boldsymbol{b}_\ell$ and $\boldsymbol{z}_\ell \geq \mathbf{0}$. Zero indicates no detected constraint violation.
- **ReLU Dual Box Violation**: The maximum violation of the dual box constraints $0 \leq \boldsymbol{\nu}_\ell \leq U_{\ell+1}^\top \boldsymbol{\nu}_{\ell+1}$ (derived in Appendix A.3). Zero confirms dual feasibility.
- **ReLU Complementarity Slack**: The absolute value of the inner product $\boldsymbol{\nu}_\ell^\top (\boldsymbol{z}_\ell - W_\ell \boldsymbol{x} - U_\ell \boldsymbol{z}_{\ell-1} - \boldsymbol{b}_\ell)$. Zero verifies complementary slackness for the ReLU block.
- **Quad./Norm Epi. Violation**: Violation of the conic epigraph constraints, e.g., $s_h \geq \frac{1}{2}\|\boldsymbol{q}_h\|_2^2$ for the quadratic branch. Zero indicates no detected conic feasibility violation.
- **Quad./Norm Tightness Slack**: The absolute difference $|s_h - \frac{1}{2}\|\boldsymbol{q}_h\|_2^2|$. Since the objective coefficient $\alpha_h > 0$, this should be zero at optimality, confirming the epigraph constraint is active (tight).
- **Norm Dual Ball/Align. Violation**: The violation of the dual norm cone constraints $\|\boldsymbol{\mu}_g\|_2 \leq \lambda_g$ and the alignment condition $\boldsymbol{\mu}_g^\top \boldsymbol{u}_g = \lambda_g t_g$. Zero confirms dual feasibility and complementarity for the SOC branch.
- **Solver Feasibility**: The same violation metrics evaluated on the solution returned by the CVXPY solver, verifying that the external solver also finds a correct optimal solution.

B.1.2 Results

Table 3: Experiment 1: Full Diagnostic Metrics for Value-Function Equivalence

Metric	Passthrough = `False`		Passthrough = `True`	
	Mean	Max	Mean	Max
Trials	150		150	
Solver Success Rate	1.0		1.0	
Optimality Gaps				
Primal-Dual Gap	1.06×10^{-14}	4.26×10^{-14}	2.88×10^{-14}	1.14×10^{-13}
Forward vs. Solver Abs. Error	5.57×10^{-7}	7.68×10^{-7}	5.57×10^{-7}	7.68×10^{-7}
Closed-Form Feasibility				
ReLU Primal Violation	0	0	0	0
ReLU Dual Box Violation	0	0	0	0
ReLU Complementarity Slack	0	0	0	0
Quadratic Epigraph Violation	0	0	0	0
Quadratic Tightness Slack	0	0	0	0
Norm Epigraph Violation	0	0	0	0
Norm Tightness Slack	0	0	0	0
Norm Dual Ball Violation	8.88×10^{-18}	4.44×10^{-16}	8.88×10^{-18}	4.44×10^{-16}
Norm Dual Alignment Violation	0	0	0	0
Solver Feasibility (CVXPY)				
ReLU Primal Violation	6.80×10^{-16}	9.99×10^{-16}	3.91×10^{-15}	7.11×10^{-15}
Quadratic Epigraph Violation	0	0	0	0
Quadratic Tightness Slack	1.46×10^{-7}	1.65×10^{-7}	1.46×10^{-7}	1.65×10^{-7}
Norm Epigraph Violation	0	0	0	0
Norm Tightness Slack	1.69×10^{-7}	2.61×10^{-7}	1.69×10^{-7}	2.61×10^{-7}
Runtime (ms)				
Closed-Form Forward	6.22 ± 0.84	–	6.66 ± 1.43	–
CVXPY Solver	24.80 ± 2.55	–	19.71 ± 4.34	–

B.2 Full Results for Experiment 2

Target Functions The ten convex functions used in Experiment 2 are defined as follows ($x \in \mathbb{R}^d$, all operations elementwise unless specified).

- **QuadraticIso** (isotropic quadratic): $f(x) = \frac{1}{2}\|x\|_2^2$.
- **QuadraticAniso** (anisotropic quadratic): $f(x) = \frac{1}{2}\sum_{i=1}^{d} w_i x_i^2$, $w_i = 0.5 + 2 \cdot \frac{i-1}{d-1}$.
- **NormEuclid** (Euclidean norm): $f(x) = \|x\|_2$.
- **NormAniso** (anisotropic norm): $f(x) = \sqrt{\sum_{i=1}^{d} w_i x_i^2}$, $w_i = 1 + 9 \cdot \frac{i-1}{d-1}$.
- **Mixed** (hybrid convex): $f(x) = 0.25 \sum w_i^{(1)} x_i^2 + 0.7\sqrt{\sum w_i^{(2)} x_i^2} + \max_k\{a_k^\top x + b_k\}$.
- **SoftplusSum**: $f(x) = \sum_{i=1}^{d} \log(1 + e^{x_i})$.
- **LogSumExpQuad**: $f(x) = \log(\sum_{i=1}^{d} e^{x_i}) + 0.1\|x\|_2^2$.
- **Huber**: $f(x) = \sum_{i=1}^{d} h_\delta(x_i)$, $\delta = 1$.
- **L1Norm**: $f(x) = \|x\|_1$.
- **ICKANPaperTarget**: $f(x) = \sum_{i=1}^{d}(|x_i| + |1 - x_i|) + 0.25 \sum w_i x_i^2$, $w_i \in [0.5, 2.0]$.

Metrics For each configuration we report the relative ℓ_2 test error (RelErr, mean $\pm$ std over 3 seeds) and the number of trainable parameters (Params). Training time (seconds) is also provided for completeness.

Results Tables 4–13 present results for each function. SOC-ICNN consistently achieves the lowest relative error across most functions and dimensions, particularly when the target exhibits explicit curvature or conic structure.

Table 4: Huber function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.828 ± 0.025	0.830 ± 0.002	0.855 ± 0.005	822	1491	2709
Softplus-ICNN	0.765 ± 0.018	0.802 ± 0.010	0.843 ± 0.007	822	1491	2709
Quad-ICNN	0.452 ± 0.053	0.285 ± 0.015	0.168 ± 0.009	848	1592	3110
Norm-ICNN	0.523 ± 0.009	0.548 ± 0.007	0.614 ± 0.002	853	1602	3130
SOC-ICNN	$\mathbf{0.246 \pm 0.066}$	$\mathbf{0.155 \pm 0.021}$	$\mathbf{0.087 \pm 0.009}$	527	1083	2451

Table 5: L1Norm function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.860 ± 0.005	0.885 ± 0.010	0.900 ± 0.003	822	1491	2709
Softplus-ICNN	0.829 ± 0.011	0.860 ± 0.010	0.892 ± 0.003	822	1491	2709
Quad-ICNN	0.536 ± 0.007	0.374 ± 0.004	0.245 ± 0.002	848	1592	3110
Norm-ICNN	0.644 ± 0.020	0.680 ± 0.005	0.728 ± 0.002	853	1602	3130
SOC-ICNN	$\mathbf{0.390 \pm 0.018}$	$\mathbf{0.266 \pm 0.015}$	$\mathbf{0.177 \pm 0.004}$	527	1083	2451

Table 6: NormEuclid function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.732 ± 0.030	0.655 ± 0.009	0.601 ± 0.015	822	1491	2709
Softplus-ICNN	0.659 ± 0.013	0.597 ± 0.010	0.563 ± 0.013	822	1491	2709
Quad-ICNN	0.340 ± 0.106	0.127 ± 0.018	0.073 ± 0.001	848	1592	3110
Norm-ICNN	0.295 ± 0.031	0.135 ± 0.006	0.075 ± 0.002	853	1602	3130
SOC-ICNN	$\mathbf{0.159 \pm 0.013}$	$\mathbf{0.079 \pm 0.024}$	$\mathbf{0.038 \pm 0.001}$	527	1083	2451

Table 7: LogSumExpQuad function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.719 ± 0.010	0.701 ± 0.013	0.697 ± 0.007	822	1491	2709
Softplus-ICNN	0.713 ± 0.056	0.672 ± 0.003	0.675 ± 0.009	822	1491	2709
Quad-ICNN	0.398 ± 0.020	0.176 ± 0.029	0.056 ± 0.011	848	1592	3110
Norm-ICNN	0.379 ± 0.044	0.232 ± 0.020	0.195 ± 0.007	853	1602	3130
SOC-ICNN	$\mathbf{0.255 \pm 0.096}$	$\mathbf{0.058 \pm 0.021}$	$\mathbf{0.020 \pm 0.001}$	527	1083	2451

Table 8: QuadraticIso function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.869 ± 0.017	0.889 ± 0.004	0.897 ± 0.002	822	1491	2709
Softplus-ICNN	0.841 ± 0.004	0.868 ± 0.004	0.893 ± 0.001	822	1491	2709
Quad-ICNN	0.566 ± 0.021	0.400 ± 0.011	0.236 ± 0.014	848	1592	3110
Norm-ICNN	0.654 ± 0.009	0.690 ± 0.005	0.732 ± 0.004	853	1602	3130
SOC-ICNN	$\mathbf{0.393 \pm 0.019}$	$\mathbf{0.279 \pm 0.015}$	$\mathbf{0.167 \pm 0.002}$	527	1083	2451

Table 9: QuadraticAniso function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.914 ± 0.006	0.924 ± 0.005	0.934 ± 0.001	822	1491	2709
Softplus-ICNN	0.899 ± 0.004	0.913 ± 0.003	0.929 ± 0.004	822	1491	2709
Quad-ICNN	0.720 ± 0.003	0.577 ± 0.005	0.407 ± 0.004	848	1592	3110
Norm-ICNN	0.779 ± 0.012	0.798 ± 0.003	0.825 ± 0.002	853	1602	3130
SOC-ICNN	$\mathbf{0.575 \pm 0.016}$	$\mathbf{0.457 \pm 0.007}$	$\mathbf{0.300 \pm 0.007}$	527	1083	2451

Table 10: NormAniso function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.879 ± 0.012	0.858 ± 0.006	0.840 ± 0.005	822	1491	2709
Softplus-ICNN	0.859 ± 0.009	0.831 ± 0.004	0.814 ± 0.007	822	1491	2709
Quad-ICNN	0.616 ± 0.031	0.333 ± 0.015	0.151 ± 0.008	848	1592	3110
Norm-ICNN	0.696 ± 0.017	0.618 ± 0.006	0.548 ± 0.005	853	1602	3130
SOC-ICNN	$\mathbf{0.441 \pm 0.010}$	$\mathbf{0.230 \pm 0.016}$	$\mathbf{0.076 \pm 0.006}$	527	1083	2451

Table 11: Mixed function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.913 ± 0.014	0.901 ± 0.002	0.903 ± 0.001	822	1491	2709
Softplus-ICNN	0.891 ± 0.014	0.884 ± 0.003	0.894 ± 0.003	822	1491	2709
Quad-ICNN	0.683 ± 0.023	0.484 ± 0.009	0.247 ± 0.005	848	1592	3110
Norm-ICNN	0.753 ± 0.004	0.737 ± 0.005	0.739 ± 0.003	853	1602	3130
SOC-ICNN	$\mathbf{0.530 \pm 0.026}$	$\mathbf{0.342 \pm 0.018}$	$\mathbf{0.179 \pm 0.013}$	527	1083	2451
P1-ICKAN	8.94 ± 0.09	10.34 ± 0.30	11.90 ± 0.12	694	1379	2749

Table 12: SoftplusSum function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.731 ± 0.046	0.787 ± 0.012	0.837 ± 0.009	822	1491	2709
Softplus-ICNN	0.706 ± 0.035	0.756 ± 0.008	0.816 ± 0.011	822	1491	2709
Quad-ICNN	0.514 ± 0.083	0.336 ± 0.045	0.204 ± 0.014	848	1592	3110
Norm-ICNN	0.468 ± 0.026	0.502 ± 0.025	0.582 ± 0.005	853	1602	3130
SOC-ICNN	$\mathbf{0.357 \pm 0.081}$	$\mathbf{0.243 \pm 0.003}$	$\mathbf{0.158 \pm 0.016}$	527	1083	2451

Table 13: ICKANPaperTarget function: relative ℓ_2 error and parameter count.

	RelErr (mean ± std)			Params		
Model	d=5	d=10	d=20	d=5	d=10	d=20
ReLU-ICNN	0.948 ± 0.002	0.956 ± 0.005	0.963 ± 0.001	822	1491	2709
Softplus-ICNN	0.935 ± 0.004	0.950 ± 0.003	0.959 ± 0.002	822	1491	2709
Quad-ICNN	0.817 ± 0.004	0.752 ± 0.003	0.650 ± 0.003	848	1592	3110
Norm-ICNN	0.862 ± 0.006	0.881 ± 0.000	0.900 ± 0.001	853	1602	3130
SOC-ICNN	$\mathbf{0.734 \pm 0.006}$	$\mathbf{0.674 \pm 0.003}$	$\mathbf{0.580 \pm 0.003}$	527	1083	2451

Table 14: Approximation results for all functions at $d = 50$ (RelErr, mean $\pm$ std).

Function	ReLU	Softplus	Quad	Norm	SOC
Huber	0.821 ± 0.005	0.815 ± 0.005	0.068 ± 0.002	0.691 ± 0.003	$\mathbf{0.038 \pm 0.001}$
L1Norm	0.875 ± 0.002	0.870 ± 0.003	0.114 ± 0.004	0.782 ± 0.004	$\mathbf{0.089 \pm 0.001}$
NormEuclid	0.345 ± 0.017	0.338 ± 0.019	0.054 ± 0.001	0.039 ± 0.002	$\mathbf{0.029 \pm 0.001}$
LogSumExpQuad	0.585 ± 0.013	0.569 ± 0.018	0.021 ± 0.000	0.190 ± 0.004	$\mathbf{0.007 \pm 0.000}$
QuadraticIso	0.873 ± 0.006	0.863 ± 0.007	0.102 ± 0.004	0.781 ± 0.002	$\mathbf{0.077 \pm 0.004}$
QuadraticAniso	0.915 ± 0.003	0.909 ± 0.004	0.179 ± 0.003	0.856 ± 0.000	$\mathbf{0.149 \pm 0.002}$
NormAniso	0.695 ± 0.001	0.683 ± 0.004	0.066 ± 0.002	0.431 ± 0.007	$\mathbf{0.044 \pm 0.001}$
Mixed	0.861 ± 0.004	0.847 ± 0.004	0.093 ± 0.002	0.750 ± 0.003	$\mathbf{0.071 \pm 0.003}$
SoftplusSum	0.799 ± 0.007	0.792 ± 0.002	0.115 ± 0.001	0.672 ± 0.004	$\mathbf{0.097 \pm 0.002}$
ICKANPaperTarget	0.951 ± 0.001	0.947 ± 0.001	0.460 ± 0.001	0.920 ± 0.001	$\mathbf{0.392 \pm 0.001}$

Table 15: Parameter counts for ICNN variants at $d = 50$.

Model	Parameters
ReLU-ICNN	9,683
Softplus-ICNN	9,683
Quad-ICNN	9,528
Norm-ICNN	9,578
SOC-ICNN	9,423

B.3 Full Results for Experiment 3

Task construction. Each task instance is defined by a context parameter $\theta \in \mathbb{R}^8$, and the goal is to solve

$$\min_{x \in \mathcal{X}} f_\theta(x).$$

We consider three feasible sets:

- **Simplex**: $\{x \in \mathbb{R}^d \mid x \geq 0,\ \sum_{i=1}^d x_i = 1\}$.
- **Box**: $\{x \in \mathbb{R}^d \mid 0 \leq x_i \leq 1,\ \forall i\}$.
- **Budget** (capped simplex): $\{x \in \mathbb{R}^d \mid 0 \leq x_i \leq 1,\ \sum_{i=1}^d x_i = B\}$ with $B = 0.3d$.

The objective function has a common quadratic–linear backbone plus a family-specific structured term:

$$f_\theta(x) = \underbrace{\frac{\alpha}{2} \sum_{i=1}^d w_i (x_i - m_i(\theta))^2 + \sum_{i=1}^d c_i(\theta) x_i}_{\text{backbone}} + f_{\text{struct}}(x; \theta).$$

Here $\alpha = 0.35$ for the logistic and log-sum-exp families and $\alpha = 1.0$ for the SOCP and Huber families; $w_i \in [0.8, 1.6]$ are sampled once and fixed per task; and

$$m(\theta) = m_0 + M\theta, \qquad c(\theta) = c_0 + C\theta$$

are affine maps with random coefficients fixed within each task.

We construct six tasks from four structural families:

- **SOCP family** (`simplex_socp`, `box_socp`):

$$f_{\text{struct}}(x; \theta) = \sum_{j=1}^J \lambda_j(\theta) \, \|A_j x - d_j(\theta)\|_2,$$

  with $J = 1$ for the single-cone tasks and $J = 2$ for the two-cone task `budget_twocone_socp`. The cone weights and offsets depend affinely on θ through softplus-transformed coefficients.

- **Logistic family** (`simplex_logistic`):

$$f_{\text{struct}}(x;\theta) = \sum_{k=1}^{K} \beta_k(\theta) \log\bigl(1 + e^{a_k^\top x - b_k(\theta)}\bigr),$$

  where $K = \max(6, \lfloor d/3 \rfloor)$.

- **LogSumExp family** (`box_logsumexp`):

$$f_{\text{struct}}(x;\theta) = \sum_{j=1}^{2} \beta_j(\theta)\ \text{logsumexp}(A_j x - b_j(\theta)).$$

- **Huber family** (`budget_huber`):

$$f_{\text{struct}}(x;\theta) = \sum_{k=1}^{K} \beta_k(\theta)\, h_\delta(a_k^\top x - b_k(\theta)), \qquad h_\delta(t) = \begin{cases} t^2, & |t| \le \delta, \\ 2\delta|t| - \delta^2, & |t| > \delta, \end{cases}$$

  with $\delta = 0.35$ and $K = \max(8, \lfloor d/2 \rfloor)$.

All random matrices and vectors are drawn once per task and dimension, then fixed across all instances.

Experimental protocol. For each task and dimension $d \in \{10, 20, 50\}$, we generate 1000 training instances, 1000 validation instances, and 200 test instances. Each training instance contains 64 candidate points for supervised learning of the surrogate objective. We compare five neural convex surrogates (ReLU, Softplus, Norm, Quad, and SOC) against two differentiable optimization baselines (PCF and DCP). At test time, the learned surrogate is optimized by projected gradient descent with 5 random restarts and 200 steps per restart.

Metrics. To keep the appendix tables compact, we report the number of trainable parameters (**Params**), the relative ℓ_2 test error (**Test RelErr**), the downstream regret $f(\hat{x}) - f(x^\star)$ (**Regret**), the decision error $\|\hat{x} - x^\star\|_2$, the wall-clock training time in seconds (**Train**), and the reported decision-time cost (**Infer**). Entries are reported as mean $\pm$ standard deviation over random seeds. Neural surrogates use 3 seeds; PCF and DCP use 2–3 seeds depending on the configuration.

Results. Tables 16–18 summarize the complete results. The overall strongest baseline is PCF, which achieves the lowest regret in most task–dimension settings. Within the neural surrogate family, however, the results still support the value of structured curvature injection: Quad is the most robust variant overall, while SOC remains competitive on several SOCP- and logistic-type tasks. A second consistent observation is the mismatch between upstream fitting and downstream decision quality: Softplus can attain relatively small test errors, yet often yields much worse regret than the better-performing curvature-injected models. Finally, the learned neural surrogates are substantially faster than DCP at both training and inference time, while remaining competitive with PCF in decision-time cost. The additional flexibility introduced by the Norm module does not translate into uniformly better downstream decisions, indicating that more elaborate curvature parameterizations are not always advantageous on this benchmark.

Table 16: Experiment 3 full results at dimension $d = 10$.

Task	Model	Params	RelErr	Regret	$\|x - \hat{x}\|_2$	Train	Infer
box_logsumexp	DCP	1245	0.058 ± 0.002	0.089 ± 0.005	0.423 ± 0.034	461.3 ± 0.5	9.60 ± 0.01
	PCF	753	$\mathbf{0.032} \pm 0.001$	$\mathbf{0.057} \pm 0.009$	$\mathbf{0.288} \pm 0.029$	237.4 ± 1.0	1.68 ± 0.00
	Norm	21777	0.078 ± 0.014	0.293 ± 0.015	0.567 ± 0.025	123.2 ± 2.6	1.30 ± 0.01
	Quad	21586	0.096 ± 0.026	0.261 ± 0.027	0.517 ± 0.019	111.7 ± 10.6	1.07 ± 0.00
	ReLU	26371	0.072 ± 0.010	0.358 ± 0.023	0.724 ± 0.038	114.0 ± 0.7	$\mathbf{0.81} \pm 0.00$
	SOC	21968	0.081 ± 0.019	0.299 ± 0.017	0.574 ± 0.024	$\mathbf{84.7} \pm 9.7$	1.55 ± 0.01
	Softplus	26371	0.115 ± 0.001	0.522 ± 0.078	0.979 ± 0.082	114.1 ± 0.6	$\mathbf{0.81} \pm 0.01$
box_socp	DCP	1245	0.094 ± 0.004	0.225 ± 0.020	0.480 ± 0.035	463.7 ± 3.3	9.61 ± 0.05
	PCF	753	$\mathbf{0.062} \pm 0.003$	$\mathbf{0.124} \pm 0.003$	$\mathbf{0.343} \pm 0.010$	240.5 ± 7.6	1.68 ± 0.00

Table 16 – continued from previous page

Task	Model	Params	RelErr	Regret	$\|x - \hat{x}\|_2$	Train	Infer
	Norm	21777	0.166 ± 0.056	0.325 ± 0.037	0.569 ± 0.026	119.1 ± 10.4	1.30 ± 0.01
	Quad	21586	0.155 ± 0.058	0.214 ± 0.029	0.391 ± 0.025	$\mathbf{105.8} \pm 24.1$	1.08 ± 0.01
	ReLU	26371	0.117 ± 0.022	0.328 ± 0.016	0.533 ± 0.032	115.0 ± 0.1	$\mathbf{0.82} \pm 0.01$
	SOC	21968	0.202 ± 0.011	0.218 ± 0.015	0.395 ± 0.019	113.0 ± 20.8	1.56 ± 0.02
	Softplus	26371	0.171 ± 0.010	0.718 ± 0.145	0.827 ± 0.102	114.5 ± 0.1	$\mathbf{0.82} \pm 0.01$
budget_huber	DCP	1245	0.072 ± 0.007	0.368 ± 0.046	0.807 ± 0.007	462.6 ± 0.3	12.19 ± 0.02
	PCF	753	$\mathbf{0.041} \pm 0.002$	$\mathbf{0.139} \pm 0.033$	$\mathbf{0.516} \pm 0.048$	235.2 ± 0.1	4.49 ± 0.04
	Norm	21777	0.106 ± 0.043	0.211 ± 0.022	0.632 ± 0.036	127.1 ± 0.6	3.22 ± 0.02
	Quad	21586	0.072 ± 0.011	0.140 ± 0.022	0.529 ± 0.021	116.1 ± 9.2	3.02 ± 0.02
	ReLU	26371	0.063 ± 0.004	0.211 ± 0.003	0.631 ± 0.018	$\mathbf{114.0} \pm 0.7$	$\mathbf{2.72} \pm 0.03$
	SOC	21968	0.089 ± 0.014	0.161 ± 0.004	0.571 ± 0.006	114.8 ± 14.9	3.33 ± 0.01
	Softplus	26371	0.057 ± 0.001	0.501 ± 0.006	0.907 ± 0.028	114.2 ± 0.5	$\mathbf{2.72} \pm 0.04$
budget_twocone_socp	DCP	1245	0.064 ± 0.007	0.516 ± 0.049	0.674 ± 0.028	464.3 ± 3.2	12.39 ± 0.24
	PCF	753	0.038 ± 0.002	$\mathbf{0.217} \pm 0.014$	$\mathbf{0.422} \pm 0.017$	234.9 ± 0.3	4.45 ± 0.01
	Norm	21777	$\mathbf{0.033} \pm 0.001$	0.240 ± 0.037	0.477 ± 0.039	126.7 ± 1.3	3.21 ± 0.02
	Quad	21586	0.046 ± 0.004	0.228 ± 0.012	0.468 ± 0.016	110.0 ± 9.6	3.01 ± 0.02
	ReLU	26371	0.050 ± 0.005	0.249 ± 0.018	0.483 ± 0.021	113.4 ± 0.6	2.74 ± 0.03
	SOC	21968	0.035 ± 0.006	0.228 ± 0.025	0.461 ± 0.025	134.3 ± 0.7	3.33 ± 0.01
	Softplus	26371	0.049 ± 0.001	0.676 ± 0.035	0.749 ± 0.020	$\mathbf{109.9} \pm 4.6$	$\mathbf{2.69} \pm 0.00$
simplex_logistic	DCP	1245	0.012 ± 0.002	0.334 ± 0.032	0.600 ± 0.019	462.5 ± 0.2	11.98 ± 0.02
	PCF	753	$\mathbf{0.004} \pm 0.001$	0.206 ± 0.025	0.442 ± 0.029	235.1 ± 0.4	4.26 ± 0.01
	Norm	21777	0.078 ± 0.025	0.202 ± 0.011	0.444 ± 0.013	126.2 ± 0.9	3.08 ± 0.03
	Quad	21586	0.069 ± 0.013	$\mathbf{0.139} \pm 0.018$	$\mathbf{0.410} \pm 0.016$	$\mathbf{99.5} \pm 5.9$	2.89 ± 0.02
	ReLU	26371	0.028 ± 0.009	0.171 ± 0.033	0.428 ± 0.032	114.1 ± 0.6	2.63 ± 0.05
	SOC	21968	0.170 ± 0.037	0.184 ± 0.013	0.433 ± 0.011	118.2 ± 10.7	3.25 ± 0.01
	Softplus	26371	0.013 ± 0.000	0.388 ± 0.008	0.610 ± 0.009	114.3 ± 1.1	$\mathbf{2.61} \pm 0.02$
simplex_socp	DCP	1245	0.101 ± 0.009	0.170 ± 0.019	0.373 ± 0.011	464.5 ± 3.7	11.99 ± 0.01
	PCF	753	$\mathbf{0.065} \pm 0.007$	0.093 ± 0.016	0.276 ± 0.028	236.3 ± 1.8	4.28 ± 0.03
	Norm	21777	0.088 ± 0.031	0.102 ± 0.013	0.276 ± 0.013	107.3 ± 25.8	3.15 ± 0.03
	Quad	21586	0.084 ± 0.032	0.091 ± 0.013	0.268 ± 0.020	$\mathbf{106.4} \pm 16.7$	2.90 ± 0.05
	ReLU	26371	0.110 ± 0.026	$\mathbf{0.086} \pm 0.010$	$\mathbf{0.263} \pm 0.013$	114.7 ± 0.7	2.66 ± 0.03
	SOC	21968	0.121 ± 0.066	0.091 ± 0.010	0.270 ± 0.016	132.8 ± 3.8	3.26 ± 0.01
	Softplus	26371	0.085 ± 0.005	0.234 ± 0.024	0.388 ± 0.008	114.8 ± 0.8	$\mathbf{2.62} \pm 0.04$

Table 17: Experiment 3 full results at dimension $d = 20$.

Task	Model	Params	RelErr	Regret	$\|x - \hat{x}\|_2$	Train	Infer
box_logsumexp	DCP	1815	0.058 ± 0.000	0.187 ± 0.001	0.707 ± 0.019	460.7 ± 0.3	9.65 ± 0.02
	PCF	1263	$\mathbf{0.031} \pm 0.002$	$\mathbf{0.094} \pm 0.002$	$\mathbf{0.435} \pm 0.018$	234.2 ± 0.5	1.68 ± 0.01
	Norm	25127	0.171 ± 0.039	0.788 ± 0.030	1.123 ± 0.046	92.4 ± 8.7	1.27 ± 0.00
	Quad	24546	0.122 ± 0.020	0.614 ± 0.020	0.914 ± 0.018	76.7 ± 0.9	1.07 ± 0.00
	ReLU	29261	0.110 ± 0.015	0.804 ± 0.042	1.204 ± 0.042	107.1 ± 11.1	$\mathbf{0.81} \pm 0.00$
	SOC	25708	0.121 ± 0.030	0.671 ± 0.010	0.967 ± 0.023	$\mathbf{71.8} \pm 6.7$	1.52 ± 0.00
	Softplus	29261	0.111 ± 0.005	1.033 ± 0.034	1.458 ± 0.057	114.7 ± 0.6	$\mathbf{0.81} \pm 0.00$
box_socp	DCP	1815	0.095 ± 0.009	0.602 ± 0.099	0.840 ± 0.060	460.2 ± 0.2	9.62 ± 0.03
	PCF	1263	$\mathbf{0.060} \pm 0.001$	$\mathbf{0.209} \pm 0.022$	$\mathbf{0.489} \pm 0.021$	234.2 ± 0.6	1.69 ± 0.01
	Norm	25127	0.099 ± 0.020	0.414 ± 0.008	0.701 ± 0.010	102.8 ± 7.2	1.28 ± 0.01
	Quad	24546	0.110 ± 0.024	0.306 ± 0.014	0.571 ± 0.021	77.7 ± 5.3	1.08 ± 0.01
	ReLU	29261	0.185 ± 0.033	0.738 ± 0.132	0.908 ± 0.088	99.3 ± 23.2	$\mathbf{0.82} \pm 0.01$
	SOC	25708	0.131 ± 0.042	0.385 ± 0.031	0.659 ± 0.019	$\mathbf{74.0} \pm 5.1$	1.52 ± 0.01
	Softplus	29261	0.144 ± 0.007	1.116 ± 0.038	1.147 ± 0.004	115.8 ± 0.6	$\mathbf{0.82} \pm 0.01$
budget_huber	DCP	1815	0.084 ± 0.008	0.583 ± 0.015	1.036 ± 0.043	460.0 ± 0.1	12.19 ± 0.03
	PCF	1263	$\mathbf{0.057} \pm 0.004$	$\mathbf{0.310} \pm 0.005$	$\mathbf{0.793} \pm 0.020$	234.2 ± 0.1	4.40 ± 0.00
	Norm	25127	0.669 ± 0.501	0.496 ± 0.044	1.016 ± 0.025	119.9 ± 7.3	3.19 ± 0.01
	Quad	24546	0.512 ± 0.174	0.365 ± 0.060	0.901 ± 0.030	$\mathbf{80.8} \pm 11.2$	3.01 ± 0.01
	ReLU	29261	0.468 ± 0.266	0.467 ± 0.038	1.024 ± 0.021	113.3 ± 2.4	$\mathbf{2.71} \pm 0.03$
	SOC	25708	0.371 ± 0.040	0.391 ± 0.051	0.933 ± 0.006	108.2 ± 20.9	3.33 ± 0.01
	Softplus	29261	0.098 ± 0.004	1.031 ± 0.111	1.303 ± 0.058	115.2 ± 0.9	$\mathbf{2.71} \pm 0.03$

Table 17 – continued from previous page

Task	Model	Params	RelErr	Regret	$\|x - \hat{x}\|_2$	Train	Infer
budget_twocone_socp	DCP	1815	0.061 ± 0.001	0.695 ± 0.120	0.870 ± 0.083	462.1 ± 0.5	12.18 ± 0.03
	PCF	1263	0.037 ± 0.001	$\mathbf{0.389} \pm 0.040$	$\mathbf{0.656} \pm 0.037$	237.0 ± 3.6	4.44 ± 0.01
	Norm	25127	$\mathbf{0.036} \pm 0.003$	0.536 ± 0.043	0.780 ± 0.038	125.0 ± 2.4	3.22 ± 0.01
	Quad	24546	0.097 ± 0.037	0.403 ± 0.015	0.692 ± 0.012	91.3 ± 21.8	3.01 ± 0.02
	ReLU	29261	0.081 ± 0.016	0.477 ± 0.047	0.748 ± 0.045	110.3 ± 8.7	2.71 ± 0.01
	SOC	25708	0.061 ± 0.021	0.477 ± 0.025	0.736 ± 0.027	121.9 ± 18.8	3.33 ± 0.00
	Softplus	29261	0.084 ± 0.033	1.199 ± 0.300	1.077 ± 0.126	$\mathbf{85.6} \pm 41.7$	$\mathbf{2.70} \pm 0.02$
simplex_logistic	DCP	1815	0.010 ± 0.001	0.467 ± 0.030	0.747 ± 0.015	460.2 ± 0.2	11.97 ± 0.02
	PCF	1263	$\mathbf{0.003} \pm 0.000$	$\mathbf{0.284} \pm 0.029$	$\mathbf{0.606} \pm 0.021$	234.9 ± 0.2	4.24 ± 0.02
	Norm	25127	0.215 ± 0.009	0.349 ± 0.023	0.666 ± 0.019	93.0 ± 48.6	3.07 ± 0.02
	Quad	24546	0.296 ± 0.084	0.300 ± 0.042	0.687 ± 0.012	100.5 ± 16.3	2.85 ± 0.02
	ReLU	29261	0.139 ± 0.036	0.305 ± 0.028	0.684 ± 0.015	102.5 ± 8.7	2.66 ± 0.06
	SOC	25708	0.172 ± 0.050	0.320 ± 0.021	0.671 ± 0.018	116.4 ± 6.2	3.23 ± 0.01
	Softplus	29261	0.013 ± 0.003	0.493 ± 0.031	0.792 ± 0.016	$\mathbf{89.3} \pm 6.5$	$\mathbf{2.61} \pm 0.01$
simplex_socp	DCP	1815	0.086 ± 0.001	0.267 ± 0.030	0.442 ± 0.021	465.3 ± 7.4	12.01 ± 0.05
	PCF	1263	$\mathbf{0.063} \pm 0.006$	$\mathbf{0.165} \pm 0.006$	0.372 ± 0.013	236.5 ± 3.6	4.28 ± 0.05
	Norm	25127	0.329 ± 0.183	0.177 ± 0.020	0.355 ± 0.021	112.2 ± 14.9	3.10 ± 0.04
	Quad	24546	0.589 ± 0.133	0.167 ± 0.010	$\mathbf{0.349} \pm 0.016$	105.9 ± 8.3	2.89 ± 0.04
	ReLU	29261	0.465 ± 0.279	0.197 ± 0.016	0.380 ± 0.030	84.4 ± 17.0	2.67 ± 0.05
	SOC	25708	0.728 ± 0.100	0.181 ± 0.014	0.352 ± 0.020	118.9 ± 25.5	3.25 ± 0.00
	Softplus	29261	0.134 ± 0.017	0.354 ± 0.035	0.488 ± 0.027	$\mathbf{49.8} \pm 3.6$	$\mathbf{2.64} \pm 0.05$

Table 18: Experiment 3 full results at dimension $d = 50$.

Task	Model	Params	RelErr	Regret	$\|x - \hat{x}\|_2$	Train	Infer
box_logsumexp	DCP	3525	0.047 ± 0.002	0.679 ± 0.007	1.488 ± 0.013	460.5 ± 0.1	9.66 ± 0.00
	PCF	2793	$\mathbf{0.029} \pm 0.001$	$\mathbf{0.549} \pm 0.002$	$\mathbf{1.030} \pm 0.005$	235.8 ± 0.2	1.67 ± 0.00
	Norm	101961	0.042 ± 0.000	1.583 ± 0.057	1.633 ± 0.031	80.0 ± 1.7	1.28 ± 0.00
	Quad	99010	0.042 ± 0.001	1.688 ± 0.067	1.788 ± 0.076	61.5 ± 6.4	1.07 ± 0.01
	ReLU	117275	0.058 ± 0.000	1.592 ± 0.018	1.724 ± 0.028	99.6 ± 14.1	$\mathbf{0.81} \pm 0.00$
	SOC	104912	0.040 ± 0.000	1.597 ± 0.034	1.601 ± 0.043	$\mathbf{61.4} \pm 5.9$	1.53 ± 0.00
	Softplus	117275	0.083 ± 0.003	2.638 ± 0.067	2.575 ± 0.048	117.5 ± 0.6	$\mathbf{0.81} \pm 0.00$
box_socp	DCP	3525	0.071 ± 0.001	2.881 ± 0.189	1.813 ± 0.055	460.5 ± 0.9	9.62 ± 0.00
	PCF	2793	0.041 ± 0.001	$\mathbf{0.628} \pm 0.010$	$\mathbf{0.796} \pm 0.017$	237.1 ± 1.2	1.68 ± 0.01
	Norm	101961	$\mathbf{0.034} \pm 0.001$	1.676 ± 0.095	1.243 ± 0.051	111.2 ± 7.5	1.28 ± 0.00
	Quad	99010	0.039 ± 0.001	1.346 ± 0.087	1.145 ± 0.044	63.0 ± 5.1	1.08 ± 0.00
	ReLU	117275	0.068 ± 0.000	2.032 ± 0.283	1.402 ± 0.083	94.7 ± 10.0	$\mathbf{0.81} \pm 0.00$
	SOC	104912	0.036 ± 0.001	1.376 ± 0.019	1.085 ± 0.012	$\mathbf{55.4} \pm 10.2$	1.53 ± 0.02
	Softplus	117275	0.078 ± 0.002	3.917 ± 0.293	2.008 ± 0.067	117.0 ± 0.5	0.82 ± 0.00
budget_huber	DCP	3525	0.056 ± 0.002	1.674 ± 0.151	1.880 ± 0.040	460.6 ± 0.1	12.19 ± 0.04
	PCF	2793	0.038 ± 0.004	$\mathbf{1.220} \pm 0.002$	$\mathbf{1.561} \pm 0.033$	235.8 ± 0.1	4.47 ± 0.01
	Norm	101961	$\mathbf{0.035} \pm 0.000$	1.650 ± 0.049	1.833 ± 0.029	94.1 ± 6.5	3.23 ± 0.01
	Quad	99010	$\mathbf{0.035} \pm 0.000$	1.270 ± 0.038	1.682 ± 0.027	59.1 ± 2.9	3.05 ± 0.01
	ReLU	117275	0.057 ± 0.001	1.665 ± 0.220	1.872 ± 0.124	117.3 ± 0.6	$\mathbf{2.73} \pm 0.03$
	SOC	104912	$\mathbf{0.035} \pm 0.001$	1.617 ± 0.010	1.839 ± 0.008	$\mathbf{50.8} \pm 8.5$	3.34 ± 0.01
	Softplus	117275	0.060 ± 0.000	3.864 ± 0.049	2.538 ± 0.012	117.4 ± 0.6	2.76 ± 0.04
budget_twocone_socp	DCP	3525	0.062 ± 0.002	1.815 ± 0.053	1.523 ± 0.028	462.3 ± 1.2	12.25 ± 0.02
	PCF	2793	0.037 ± 0.001	$\mathbf{0.809} \pm 0.094$	$\mathbf{1.005} \pm 0.042$	236.0 ± 0.3	4.45 ± 0.00
	Norm	101961	$\mathbf{0.026} \pm 0.000$	1.254 ± 0.040	1.247 ± 0.015	80.2 ± 4.8	3.21 ± 0.01
	Quad	99010	0.029 ± 0.000	1.145 ± 0.072	1.212 ± 0.016	$\mathbf{56.0} \pm 6.3$	3.03 ± 0.01
	ReLU	117275	0.061 ± 0.005	1.333 ± 0.122	1.305 ± 0.062	99.6 ± 24.3	$\mathbf{2.73} \pm 0.03$
	SOC	104912	$\mathbf{0.026} \pm 0.001$	1.309 ± 0.013	1.282 ± 0.018	64.3 ± 11.3	3.33 ± 0.01
	Softplus	117275	0.060 ± 0.000	3.487 ± 0.179	1.981 ± 0.036	116.2 ± 2.1	2.76 ± 0.05
simplex_logistic	DCP	3525	0.007 ± 0.001	0.575 ± 0.003	0.608 ± 0.027	460.4 ± 0.7	12.03 ± 0.01
	PCF	2793	$\mathbf{0.001} \pm 0.000$	0.394 ± 0.050	$\mathbf{0.516} \pm 0.017$	236.3 ± 0.2	4.30 ± 0.02
	Norm	101961	0.097 ± 0.032	0.450 ± 0.020	0.558 ± 0.010	76.1 ± 42.4	3.07 ± 0.01
	Quad	99010	0.095 ± 0.057	$\mathbf{0.363} \pm 0.033$	0.532 ± 0.016	$\mathbf{55.0} \pm 24.6$	2.87 ± 0.02
	ReLU	117275	0.046 ± 0.018	0.411 ± 0.035	0.560 ± 0.021	91.7 ± 17.4	$\mathbf{2.64} \pm 0.00$

Table 18 – continued from previous page

Task	Model	Params	RelErr	Regret	$\|x - \hat{x}\|_2$	Train	Infer
	SOC	104912	0.082 ± 0.040	0.367 ± 0.018	0.524 ± 0.005	89.6 ± 9.7	3.24 ± 0.00
	Softplus	117275	0.008 ± 0.001	0.604 ± 0.037	0.597 ± 0.011	98.0 ± 23.8	$\mathbf{2.64} \pm 0.04$
simplex_socp	DCP	3525	0.057 ± 0.003	0.425 ± 0.008	0.490 ± 0.008	467.4 ± 10.1	12.22 ± 0.29
	PCF	2793	$\mathbf{0.028} \pm 0.002$	$\mathbf{0.286} \pm 0.014$	$\mathbf{0.423} \pm 0.014$	238.2 ± 2.3	4.31 ± 0.05
	Norm	101961	0.656 ± 0.174	0.356 ± 0.004	0.430 ± 0.015	112.6 ± 14.8	3.14 ± 0.01
	Quad	99010	1.174 ± 0.334	0.347 ± 0.006	0.434 ± 0.008	116.4 ± 13.2	2.94 ± 0.04
	ReLU	117275	0.578 ± 0.064	0.365 ± 0.011	0.449 ± 0.005	109.2 ± 6.2	2.66 ± 0.03
	SOC	104912	0.859 ± 0.211	0.354 ± 0.004	0.428 ± 0.001	126.1 ± 22.3	3.27 ± 0.02
	Softplus	117275	0.105 ± 0.017	0.449 ± 0.007	0.481 ± 0.024	$\mathbf{76.8} \pm 20.5$	$\mathbf{2.64} \pm 0.04$

C Extension to Convolutional SOC-ICNNs

In the main text, the proposed SOC-ICNN is presented in fully connected form for clarity. We now show that the same construction extends directly to convolutional architectures. This extension does *not* change the underlying optimization class: it only replaces dense affine maps by convolutional linear operators with parameter sharing and sparse structure. Consequently, the convexity results, the LP value-function interpretation of the ReLU backbone in Lemma 1, and the SOCP value-function interpretation in Theorem 1 all remain valid.

C.1 Convolution as a Structured Linear Operator

Let the input be a tensor

$$X \in \mathbb{R}^{C_0 \times H_0 \times W_0}.$$

For any tensor Y, let $\mathrm{vec}(Y)$ denote its vectorization into a column vector. Fix a discrete convolution operator $\mathcal{K}$ together with its stride, padding, and dilation. Then there exists a matrix $T(\mathcal{K})$ such that

$$\mathrm{vec}(\mathcal{K}(Y)) = T(\mathcal{K})\,\mathrm{vec}(Y). \tag{12}$$

The matrix $T(\mathcal{K})$ is the usual sparse Toeplitz / block-Toeplitz matrix induced by the convolution. Thus, every convolutional layer is still a linear map with respect to the input tensor, and every affine convolutional layer is of the form

$$Y \mapsto \mathcal{K}(Y) + B$$

for some bias tensor B.

Therefore, after vectorization, a convolutional architecture is simply a fully connected architecture with highly structured sparse matrices. This observation is the only fact needed to transfer the value-function interpretation from the fully connected setting to the convolutional setting.

C.2 Convolutional ReLU-ICNN Backbone

We define the convolutional ReLU-ICNN backbone recursively as follows. Let $Z_0 = 0$, and for $\ell = 1, \ldots, L$ define

$$Z_\ell = \sigma\Big(\mathcal{W}_\ell(X) + \mathcal{U}_\ell(Z_{\ell-1}) + B_\ell\Big), \tag{13}$$

where:

- $\mathcal{W}_\ell$ is an unconstrained input-to-hidden convolutional affine map;
- $\mathcal{U}_\ell$ is a hidden-to-hidden convolutional linear map whose kernel coefficients are constrained to be elementwise nonnegative;
- B_ℓ is a bias tensor;
- $\sigma(t) = \max\{t, 0\}$ is applied elementwise.

The scalar output is defined by

$$f_{\text{ReLU-CNN}}(X) = \langle C, Z_L \rangle + \langle V, X \rangle + b_0, \tag{14}$$

where $C \geq 0$ elementwise, V is unconstrained, and $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product between same-sized tensors.

This is the exact convolutional analogue of the fully connected ReLU-ICNN in Section 3, with dense affine maps replaced by convolutional affine maps.

C.3 Convolutional Backbone Remains Convex

We now make explicit why replacing dense layers by convolutional layers does not destroy input convexity.

Proposition 5 (Convexity of the convolutional ReLU-ICNN backbone)**.** *The function* $X \mapsto f_{\text{ReLU-CNN}}(X)$ *defined by* (13)–(14) *is convex with respect to the input tensor* X.

Proof. The proof follows the same inductive logic as for standard ICNNs, but we spell it out because the convolutional case is easy to misunderstand.

Step 1: Convolution is affine in the input. For each layer ℓ, both $\mathcal{W}_\ell(X)$ and $\mathcal{U}_\ell(Z_{\ell-1})$ are affine / linear with respect to their arguments. Hence, if each entry of $Z_{\ell-1}(X)$ is a convex function of X, then each entry of $\mathcal{U}_\ell(Z_{\ell-1}(X))$ is a nonnegative linear combination of translated entries of $Z_{\ell-1}(X)$, and is therefore convex in X.

More explicitly, fix an output channel and spatial position index p. The corresponding entry of $\mathcal{U}_\ell(Z_{\ell-1}(X))$ can be written as

$$\big[\mathcal{U}_\ell(Z_{\ell-1}(X))\big]_p = \sum_q \sum_r u^{(\ell)}_{pqr} \, [Z_{\ell-1}(X)]_{q,r},$$

where the coefficients $u^{(\ell)}_{pqr}$ are the convolution kernel coefficients. Because $\mathcal{U}_\ell \geq 0$ elementwise, we have $u^{(\ell)}_{pqr} \geq 0$ for all indices. Thus, if each $[Z_{\ell-1}(X)]_{q,r}$ is convex in X, then the above sum is convex in X.

Step 2: The pre-activation is convex. Define the pre-activation tensor

$$S_\ell(X) = \mathcal{W}_\ell(X) + \mathcal{U}_\ell(Z_{\ell-1}(X)) + B_\ell.$$

Since $\mathcal{W}_\ell(X)$ is affine in X, $\mathcal{U}_\ell(Z_{\ell-1}(X))$ is convex in X by Step 1, and adding a bias preserves convexity, every entry of $S_\ell(X)$ is convex in X.

Step 3: ReLU preserves convexity because it is convex and monotone nondecreasing. For each scalar entry $s(X)$ of $S_\ell(X)$, the corresponding activation is

$$\sigma(s(X)) = \max\{s(X), 0\}.$$

Since $s(X)$ is convex and σ is a convex, nondecreasing scalar function, the composition $\sigma \circ s$ is convex. Equivalently, one may directly note that $\max\{s(X), 0\}$ is the maximum of two convex functions, hence convex.

Therefore, every entry of $Z_\ell(X)$ is convex in X.

Step 4: Induction over layers. At layer $\ell = 1$, we have

$$Z_1(X) = \sigma(\mathcal{W}_1(X) + B_1),$$

whose entries are convex because $\mathcal{W}_1(X) + B_1$ is affine. Applying Steps 1–3 inductively proves that every entry of every hidden tensor $Z_\ell(X)$ is convex in X.

Step 5: Final readout preserves convexity. The output

$$f_{\text{ReLU-CNN}}(X) = \langle C, Z_L(X) \rangle + \langle V, X \rangle + b_0$$

is the sum of: (i) a nonnegative linear combination of convex entries of $Z_L(X)$, because $C \geq 0$ elementwise; (ii) an affine term $\langle V, X \rangle + b_0$. Hence $f_{\text{ReLU-CNN}}(X)$ is convex in X. □

The key point is therefore unchanged from the fully connected setting: *the hidden-to-hidden operator must preserve convexity, and this is ensured by constraining its coefficients to be nonnegative.* A convolutional kernel is just a structured collection of linear coefficients; as long as those coefficients are nonnegative, the ICNN convexity argument survives verbatim.

C.4 LP Value-Function Interpretation of the Convolutional Backbone

The previous proposition establishes convexity directly. We now show that the convolutional ReLU backbone also admits the same LP value-function interpretation as Lemma 1.

Proposition 6 (Convolutional LP value-function representation)**.** *For every input tensor X, the output $f_{\text{ReLU-CNN}}(X)$ equals the optimal value of the following linear program:*

$$\begin{aligned} f_{\text{ReLU-CNN}}(X) = \min_{\{Z_\ell\}_{\ell=1}^{L}} \quad & \langle C, Z_L \rangle + \langle V, X \rangle + b_0 \\ \text{s.t.} \quad & Z_\ell \geq \mathcal{W}_\ell(X) + \mathcal{U}_\ell(Z_{\ell-1}) + B_\ell, \quad \ell = 1, \dots, L, \\ & Z_\ell \geq 0, \quad \ell = 1, \dots, L, \\ & Z_0 = 0. \end{aligned} \tag{15}$$

Proof. The argument is exactly parallel to Lemma 1.

For fixed input X, the ReLU recursion in (13) computes the componentwise smallest feasible activation tensor satisfying the linear inequalities in (15). Because the output coefficient tensor C is elementwise nonnegative, the objective is monotone nondecreasing in the entries of Z_L. Hence the smallest feasible tensor minimizes the objective, and the LP optimum equals the forward output.

Equivalently, after vectorizing all tensors, (15) becomes

$$\begin{aligned} \min_{\{z_\ell\}_{\ell=1}^{L}} \quad & c^\top z_L + v^\top x + b_0 \\ \text{s.t.} \quad & z_\ell \geq W_\ell x + U_\ell z_{\ell-1} + b_\ell, \quad \ell = 1, \dots, L, \\ & z_\ell \geq 0, \quad \ell = 1, \dots, L, \\ & z_0 = 0, \end{aligned} \tag{16}$$

where $x = \text{vec}(X)$, $z_\ell = \text{vec}(Z_\ell)$, and the matrices W_ℓ, U_ℓ are precisely the structured sparse matrices induced by the convolutional operators $\mathcal{W}_\ell, \mathcal{U}_\ell$. This is exactly the LP form in Lemma 1. □

Thus, moving from fully connected layers to convolutional layers changes only the parameterization of the affine operators, not the underlying LP value-function nature of the ReLU-ICNN backbone.

C.5 Convolutional SOC-ICNN

We now augment the convolutional ReLU backbone with the same two geometric primitives used in the main text.

Quadratic convolutional branch. For each $h = 1, \dots, H$, let $\mathcal{B}_h$ be an affine convolutional operator acting on X. We define the h-th quadratic branch by

$$\frac{\alpha_h}{2} \left\| \mathcal{B}_h(X) + E_h \right\|_F^2, \qquad \alpha_h \geq 0, \tag{17}$$

where E_h is a bias tensor and $\| \cdot \|_F$ denotes the Frobenius norm.

Conic convolutional branch. For each $g = 1, \dots, G$, let $\mathcal{A}_g$ be another affine convolutional operator acting on X. We define the g-th conic branch by

$$\lambda_g \left\| \mathcal{A}_g(X) + D_g \right\|_F, \qquad \lambda_g \geq 0, \tag{18}$$

where D_g is a bias tensor.

The full convolutional SOC-ICNN is then defined as

$$f_{\text{SOC-CNN}}(X) = f_{\text{ReLU-CNN}}(X) + \sum_{h=1}^{H} \frac{\alpha_h}{2} \left\| \mathcal{B}_h(X) + E_h \right\|_F^2 + \sum_{g=1}^{G} \lambda_g \left\| \mathcal{A}_g(X) + D_g \right\|_F. \tag{19}$$

The role of the three terms is exactly the same as in (7): the convolutional ReLU backbone captures polyhedral structure, the quadratic convolutional branches inject explicit PSD curvature, and the conic convolutional branches inject Euclidean norm geometry at the level of feature maps.

C.6 Convolutional SOC-ICNN Remains Convex

We now make explicit why the full convolutional SOC-ICNN still preserves input convexity.

Proposition 7 (Convexity of convolutional SOC-ICNN)**.** *The function $X \mapsto f_{\text{SOC-CNN}}(X)$ defined in* (19) *is convex with respect to the input tensor X.*

Proof. By Proposition 5, the backbone term $f_{\text{ReLU-CNN}}(X)$ is convex in X.

For each quadratic branch, the map

$$X \mapsto \mathcal{B}_h(X) + E_h$$

is affine in X, because $\mathcal{B}_h$ is an affine convolutional operator. Hence

$$X \mapsto \frac{\alpha_h}{2}\|\mathcal{B}_h(X) + E_h\|_F^2$$

is convex for every $\alpha_h \geq 0$, since the squared Frobenius norm of an affine map is convex.

Likewise, for each conic branch, the map

$$X \mapsto \mathcal{A}_g(X) + D_g$$

is affine in X, so

$$X \mapsto \lambda_g\|\mathcal{A}_g(X) + D_g\|_F$$

is convex for every $\lambda_g \geq 0$, since the Frobenius norm is convex and convexity is preserved under composition with affine maps.

Finally, a nonnegative sum of convex functions is convex. Therefore,

$$f_{\text{SOC-CNN}}(X) = f_{\text{ReLU-CNN}}(X) + \sum_{h=1}^{H} \frac{\alpha_h}{2}\|\mathcal{B}_h(X) + E_h\|_F^2 + \sum_{g=1}^{G} \lambda_g\|\mathcal{A}_g(X) + D_g\|_F$$

is convex in X. □

This proposition shows that the transition from dense affine maps to convolutional affine maps does not alter the convex-analytic foundation of SOC-ICNN. The reason is simple but fundamental: *convexity in the main text relies only on affine dependence on the input, nonnegative hidden-to-hidden coefficients in the ICNN backbone, and convex norm / squared-norm primitives. Convolution preserves all three properties.*

C.7 SOCP Value-Function Interpretation

We now state the convolutional counterpart of Theorem 1.

For each quadratic branch, introduce an auxiliary tensor

$$Q_h = \mathcal{B}_h(X) + E_h$$

and an epigraph variable s_h satisfying

$$(s_h,\, 1,\, \text{vec}(Q_h)) \in \mathcal{Q}_r^{n_h+2}, \tag{20}$$

where $n_h = \dim(\text{vec}(Q_h))$.

For each conic branch, introduce an auxiliary tensor

$$U_g = \mathcal{A}_g(X) + D_g$$

and an epigraph variable t_g satisfying

$$(\text{vec}(U_g),\, t_g) \in \mathcal{Q}^{m_g+1}, \tag{21}$$

where $m_g = \dim(\text{vec}(U_g))$.

Theorem 2 (Convolutional SOC-ICNN as an SOCP value function)**.** *For every input tensor X, the output in* (19) *is exactly equal to the optimal value of the following SOCP:*

$$\begin{aligned} f_{\text{SOC-CNN}}(X) = \min_{\eta} \quad & \langle C, Z_L \rangle + \langle V, X \rangle + b_0 + \sum_{h=1}^{H} \alpha_h s_h + \sum_{g=1}^{G} \lambda_g t_g \\ \text{s.t.} \quad & Z_\ell \geq \mathcal{W}_\ell(X) + \mathcal{U}_\ell(Z_{\ell-1}) + B_\ell, \quad \ell = 1, \dots, L, \\ & Z_\ell \geq 0, \quad \ell = 1, \dots, L, \\ & Z_0 = 0, \\ & Q_h = \mathcal{B}_h(X) + E_h, \quad (s_h, 1, \text{vec}(Q_h)) \in \mathcal{Q}_r^{n_h+2}, \quad h = 1, \dots, H, \\ & U_g = \mathcal{A}_g(X) + D_g, \quad (\text{vec}(U_g), t_g) \in \mathcal{Q}^{m_g+1}, \quad g = 1, \dots, G, \end{aligned} \tag{22}$$

where

$$\eta = \{Z_\ell, Q_h, s_h, U_g, t_g\}.$$

Proof. The proof is the exact convolutional analogue of Theorem 1.

Let $V(X)$ denote the optimal value of (22).

Upper bound: $V(X) \leq f_{\text{SOC-CNN}}(X)$. Take the forward activations $\bar{Z}_\ell$ produced by the convolutional ReLU backbone, so that

$$\langle C, \bar{Z}_L \rangle + \langle V, X \rangle + b_0 = f_{\text{ReLU-CNN}}(X).$$

For each quadratic branch, set

$$\bar{Q}_h = \mathcal{B}_h(X) + E_h, \qquad \bar{s}_h = \frac{1}{2}\|\bar{Q}_h\|_F^2.$$

Then

$$(\bar{s}_h, 1, \text{vec}(\bar{Q}_h)) \in \mathcal{Q}_r^{n_h+2}.$$

For each conic branch, set

$$\bar{U}_g = \mathcal{A}_g(X) + D_g, \qquad \bar{t}_g = \|\bar{U}_g\|_F.$$

Then

$$(\text{vec}(\bar{U}_g), \bar{t}_g) \in \mathcal{Q}^{m_g+1}.$$

Therefore this choice is feasible for (22), and its objective value is exactly

$$f_{\text{ReLU-CNN}}(X) + \sum_{h=1}^{H} \frac{\alpha_h}{2}\|\mathcal{B}_h(X) + E_h\|_F^2 + \sum_{g=1}^{G} \lambda_g \|\mathcal{A}_g(X) + D_g\|_F = f_{\text{SOC-CNN}}(X).$$

Hence,

$$V(X) \leq f_{\text{SOC-CNN}}(X).$$

Lower bound: $V(X) \geq f_{\text{SOC-CNN}}(X)$. Take any feasible point of (22). By Proposition 6,

$$\langle C, Z_L \rangle + \langle V, X \rangle + b_0 \geq f_{\text{ReLU-CNN}}(X).$$

Moreover, the rotated cone constraints imply

$$s_h \geq \frac{1}{2}\|Q_h\|_F^2 = \frac{1}{2}\|\mathcal{B}_h(X) + E_h\|_F^2,$$

and the standard cone constraints imply

$$t_g \geq \|U_g\|_F = \|\mathcal{A}_g(X) + D_g\|_F.$$

Multiplying by the nonnegative coefficients α_h, λ_g and summing yields

$$\sum_{h=1}^{H} \alpha_h s_h \geq \sum_{h=1}^{H} \frac{\alpha_h}{2}\|\mathcal{B}_h(X) + E_h\|_F^2,$$

and

$$\sum_{g=1}^{G} \lambda_g t_g \geq \sum_{g=1}^{G} \lambda_g \|\mathcal{A}_g(X) + D_g\|_F.$$

Therefore every feasible objective value in (22) is at least

$$f_{\text{ReLU-CNN}}(X) + \sum_{h=1}^{H} \frac{\alpha_h}{2} \|\mathcal{B}_h(X) + E_h\|_F^2 + \sum_{g=1}^{G} \lambda_g \|\mathcal{A}_g(X) + D_g\|_F = f_{\text{SOC-CNN}}(X).$$

Hence,

$$V(X) \geq f_{\text{SOC-CNN}}(X).$$

Combining the two inequalities proves

$$V(X) = f_{\text{SOC-CNN}}(X).$$

□

C.8 Relation to the Main Theoretical Results

The convolutional extension is not a different theory, but a different parameterization of the same theory.

Proposition 8 (Equivalence after vectorization)**.** *After vectorization, every convolutional SOC-ICNN can be written as a fully connected SOC-ICNN whose affine operators are structured sparse matrices induced by convolution. Consequently, the representational and optimization-theoretic conclusions of the main text extend directly to the convolutional setting.*

Proof. By (12), every convolutional operator admits a matrix representation after vectorization. Applying this to all operators $\mathcal{W}_\ell, \mathcal{U}_\ell, \mathcal{B}_h, \mathcal{A}_g$ transforms the convolutional model into the fully connected form of (2), (7), and (8), except that the corresponding matrices are sparse and share parameters according to convolutional structure.

Therefore: (i) Lemma 1 becomes Proposition 6; (ii) Theorem 1 becomes Theorem 2; and (iii) the function-class enlargement argument of Proposition 3 carries over directly, since the quadratic and conic branches remain exact structured primitives after vectorization. □

In particular, the strict enlargement from polyhedral LP geometry to conic SOCP geometry is unchanged. What changes is only the inductive bias: the convolutional version is tailored to spatially structured inputs and benefits from local connectivity and parameter sharing, while retaining the same convexity guarantees and the same exact value-function interpretation.

Remark 1 (On allowed CNN modules)**.** The above extension applies directly to any *linear* convolutional module, including standard convolutions, strided convolutions, dilated convolutions, transposed convolutions, 1×1 convolutions, and average pooling. To preserve the exact LP/SOCP value-function interpretation in the present form, one should avoid inserting additional nonlinear operators such as batch normalization inside the convex backbone, unless they are analyzed separately.

C.9 Convolutional Downstream Test on Real Images

To complement the synthetic approximation and downstream decision experiments in the main text, we include a small real-image downstream test for the convolutional SOC-ICNN extension. The purpose of this experiment is not to compete with specialized image restoration methods, but to verify that the same LP-to-SOCP lifting principle remains operationally useful when the decision variable has real spatial structure. The convolutional ReLU baseline follows the input-convex construction principle of ICNNs [Amos et al., 2017], with nonnegative hidden-to-hidden convolutional weights to preserve convexity with respect to the image decision variable.

Table 19: Convolutional downstream image restoration test on Olivetti Faces 64×64 grayscale images. Regret and decision error are evaluated using the true convex restoration energy after optimizing each learned convolutional surrogate. Lower is better for RelErr, regret, and decision error.

Model	RelErr	Regret	Decision error	Infer. (ms)
Conv-ReLU	0.0492	0.0254 ± 0.0033	0.2497 ± 0.0174	86.73
Conv-Softplus	0.4434	0.0821 ± 0.0192	0.4736 ± 0.0527	93.09
Conv-Quad	0.0474	0.0111 ± 0.0014	0.1633 ± 0.0107	103.72
Conv-Norm	0.0473	0.0074 ± 0.0026	0.1377 ± 0.0282	66.77
Conv-SOC	**0.0461**	$\mathbf{0.0041} \pm 0.0011$	$\mathbf{0.1303} \pm 0.0207$	84.84

Task. We use the Olivetti Faces dataset, which consists of real grayscale face images at native 64×64 resolution. For each clean image $X^\natural$, we generate a corrupted observation Y by applying a mild Gaussian blur, additive Gaussian noise, a light random pixel mask, and several rectangular missing blocks. The decision variable is the restored image $X \in [0,1]^{1\times 64\times 64}$. The true convex restoration energy is

$$F_{\mathrm{img}}(X;Y,M) = \frac{\rho}{2|\Omega|}\|M \odot (H * X - Y)\|_F^2 + \frac{\beta}{2|\Omega|}\|L * X\|_F^2 + \frac{\lambda}{|\Omega|}\sum_{(i,j)\in\Omega}\sqrt{(D_x X)_{i,j}^2 + (D_y X)_{i,j}^2 + \varepsilon},$$

where M is the observation mask, Ω is the image lattice, H is a Gaussian blur operator, L is the discrete Laplacian filter, and D_x, D_y are finite-difference filters. This objective combines masked data fidelity, convolutional quadratic smoothing, and a TV-like conic regularizer, making it a natural downstream test for convolutional SOC-ICNNs.

Models. We compare five convolutional convex surrogates: Conv-ReLU, Conv-Softplus, Conv-Quad, Conv-Norm, and Conv-SOC. All models use the same three-layer convolutional ICNN backbone with 10 hidden channels. The Conv-Quad and Conv-Norm variants add only the quadratic or conic branch, respectively, while Conv-SOC includes both branches. The quadratic branch uses 16 convolutional output channels, and the conic branch uses 8 two-dimensional norm groups. All hidden-to-hidden convolutional kernels in the ICNN backbone are constrained to be elementwise nonnegative.

Data and optimization protocol. We split the 400 Olivetti images into 280 training images, 60 validation images, and 60 test images. For each image, we generate 16 candidate restored images and evaluate the true convex energy to train the surrogate by mean-squared error regression. The candidate set includes corrupted observations, clean images, blurred images, random images, noisy perturbations, and convex mixtures. All models are trained for 150 epochs using AdamW with learning rate 2×10^{-3}, batch size 96, and weight decay 10^{-6}. For downstream evaluation, each learned surrogate is optimized over $X \in [0,1]^{1\times 64\times 64}$ using projected Adam for 350 steps and 3 random restarts. The approximate reference solution $X^\star$ is computed by approximately minimizing the true convex energy using projected Adam for 700 steps and the same box projection. We report relative prediction error, true-energy regret $F_{\mathrm{img}}(\hat{X};Y,M) - F_{\mathrm{img}}(X^\star;Y,M)$, pixel-level decision error $\|\hat{X} - X^\star\|_F/\sqrt{|\Omega|}$, and downstream inference time.

Implementation. The experiment is implemented in Python 3.11 with PyTorch 2.0 and run on a single NVIDIA RTX 4060 Ti GPU. All random seeds are fixed to 0. The corruption parameters are $\rho = 1.0$, $\beta = 0.035$, $\lambda = 0.055$, $\varepsilon = 10^{-4}$, Gaussian blur standard deviation 0.7, Gaussian noise standard deviation 0.025, base pixel keep probability 0.90, and 2–4 rectangular holes per image with side lengths between 8 and 18 pixels.

D Extension to Recurrent SOC-ICNNs

In addition to convolutional architectures, the same SOC-ICNN construction also extends to finite-horizon recurrent architectures. The key point is that a recurrent network over a fixed horizon can



Figure 2: Qualitative downstream image restoration examples on Olivetti Faces 64×64 grayscale images. Each row is a test instance, and each learned convolutional surrogate is optimized over the restored image. The structured Quad, Norm, and SOC variants recover the dominant facial structure under the learned convex energy; Conv-SOC gives the lowest relative prediction error, true-energy regret, and pixel-level decision error in Table 19.

be unrolled into a feed-forward computation graph with shared or time-varying affine maps. Once unrolled, the same monotonicity and epigraph arguments used in the main text apply directly. Thus, the recurrent extension does not change the underlying value-function class: it replaces a dense feed-forward backbone by a structured affine recurrence while preserving input convexity and the SOCP value-function interpretation.

D.1 Finite-Horizon Recurrent Backbone

Let the input be a sequence

$$\boldsymbol{x}_{1:T} := (\boldsymbol{x}_1, \dots, \boldsymbol{x}_T), \qquad \boldsymbol{x}_t \in \mathbb{R}^{d_t}.$$

A finite-horizon recurrent ReLU-ICNN backbone is defined by

$$\boldsymbol{z}_t = \sigma(W_t \boldsymbol{x}_t + R_t \boldsymbol{z}_{t-1} + \boldsymbol{b}_t), \qquad t = 1, \dots, T, \qquad \boldsymbol{z}_0 = \mathbf{0}, \tag{23}$$

where $\sigma(a) = \max\{a, 0\}$ is applied elementwise. The recurrent hidden-to-hidden weights satisfy

$$R_t \geq 0, \qquad \boldsymbol{c} \geq 0,$$

elementwise. The terminal-output recurrent ICNN is

$$f_{\text{ReLU-RNN}}(\boldsymbol{x}_{1:T}) = \boldsymbol{c}^\top \boldsymbol{z}_T + \sum_{t=1}^{T} \boldsymbol{v}_t^\top \boldsymbol{x}_t + b_0. \tag{24}$$

The matrices W_t, vectors $\boldsymbol{v}_t$, and biases $\boldsymbol{b}_t$ are unconstrained. The time-varying notation is used for generality; the usual weight-sharing RNN is obtained by setting $W_t = W$, $R_t = R$, and $\boldsymbol{b}_t = \boldsymbol{b}$.

D.2 LP Value-Function Interpretation of the Recurrent Backbone

For any fixed sequence $\boldsymbol{x}_{1:T}$, the recurrent backbone admits an LP value-function representation:

$$\begin{aligned} f_{\text{ReLU-RNN}}(\boldsymbol{x}_{1:T}) = \min_{\{\boldsymbol{z}_t\}_{t=1}^T} \quad & \boldsymbol{c}^\top \boldsymbol{z}_T + \sum_{t=1}^T \boldsymbol{v}_t^\top \boldsymbol{x}_t + b_0 \\ \text{s.t.} \quad & \boldsymbol{z}_t \geq W_t \boldsymbol{x}_t + R_t \boldsymbol{z}_{t-1} + \boldsymbol{b}_t, \quad \boldsymbol{z}_t \geq \boldsymbol{0}, \quad t = 1, \dots, T, \\ & \boldsymbol{z}_0 = \boldsymbol{0}. \end{aligned} \tag{25}$$

Indeed, the ReLU recurrence in (23) constructs the componentwise minimal feasible hidden sequence. Because $R_t \geq 0$, monotonicity propagates through time, and because $\boldsymbol{c} \geq 0$, the terminal objective is minimized by the forward recurrent activations. Thus, the recurrent ReLU-ICNN is still an LP value-function model after unrolling.

D.3 Recurrent SOC-ICNN Architecture

We now add the same quadratic and conic structural branches used in the main SOC-ICNN. For a sequence input, the branch maps are affine functions of the entire sequence:

$$\mathcal{B}_h(\boldsymbol{x}_{1:T}) = \sum_{t=1}^T B_{h,t} \boldsymbol{x}_t + \boldsymbol{e}_h, \qquad \mathcal{A}_g(\boldsymbol{x}_{1:T}) = \sum_{t=1}^T A_{g,t} \boldsymbol{x}_t + \boldsymbol{d}_g.$$

The recurrent SOC-ICNN is then defined as

$$\begin{aligned} f_{\text{SOC-RNN}}(\boldsymbol{x}_{1:T}) = & f_{\text{ReLU-RNN}}(\boldsymbol{x}_{1:T}) + \sum_{h=1}^H \frac{\alpha_h}{2} \left\| \mathcal{B}_h(\boldsymbol{x}_{1:T}) \right\|_2^2 \\ & + \sum_{g=1}^G \lambda_g \left\| \mathcal{A}_g(\boldsymbol{x}_{1:T}) \right\|_2, \end{aligned} \tag{26}$$

where $\alpha_h \geq 0$ and $\lambda_g \geq 0$. This is the direct recurrent analogue of (7): the ReLU recurrent backbone captures polyhedral temporal structure, while the quadratic and conic branches inject PSD curvature and Euclidean norm geometry over the full input sequence.

D.4 Recurrent SOC-ICNN Remains Convex

Proposition 9 (Convexity of recurrent SOC-ICNN)**.** *For any finite horizon T, the function $\boldsymbol{x}_{1:T} \mapsto f_{\text{SOC-RNN}}(\boldsymbol{x}_{1:T})$ defined in* (26) *is convex with respect to the full input sequence.*

Proof. We first prove that the recurrent ReLU backbone is convex in $\boldsymbol{x}_{1:T}$. The initial state $\boldsymbol{z}_0 = \boldsymbol{0}$ is constant and hence convex. Assume that each component of $\boldsymbol{z}_{t-1}$ is convex in $\boldsymbol{x}_{1:t-1}$. Since $R_t \geq 0$, the term $R_t \boldsymbol{z}_{t-1}$ is a nonnegative linear combination of convex functions. The term $W_t \boldsymbol{x}_t + \boldsymbol{b}_t$ is affine in $\boldsymbol{x}_t$. Therefore each component of

$$W_t \boldsymbol{x}_t + R_t \boldsymbol{z}_{t-1} + \boldsymbol{b}_t$$

is convex in $\boldsymbol{x}_{1:t}$. Because ReLU is convex and nondecreasing, applying σ elementwise preserves convexity. Thus each component of $\boldsymbol{z}_t$ is convex by induction. Since $\boldsymbol{c} \geq 0$, the terminal term $\boldsymbol{c}^\top \boldsymbol{z}_T$ is convex, and the passthrough term $\sum_{t=1}^T \boldsymbol{v}_t^\top \boldsymbol{x}_t + b_0$ is affine. Hence $f_{\text{ReLU-RNN}}$ is convex.

For each quadratic branch, $\mathcal{B}_h(\boldsymbol{x}_{1:T})$ is affine in the full sequence, so

$$\boldsymbol{x}_{1:T} \mapsto \frac{\alpha_h}{2} \left\| \mathcal{B}_h(\boldsymbol{x}_{1:T}) \right\|_2^2$$

is convex for every $\alpha_h \geq 0$. Likewise, each conic branch

$$\boldsymbol{x}_{1:T} \mapsto \lambda_g \left\| \mathcal{A}_g(\boldsymbol{x}_{1:T}) \right\|_2$$

is convex for every $\lambda_g \geq 0$. A nonnegative sum of convex functions is convex, proving the claim. □

D.5 SOCP Value-Function Interpretation

We now state the recurrent counterpart of Theorem 1. For each quadratic branch, introduce

$$\boldsymbol{q}_h = \mathcal{B}_h(\boldsymbol{x}_{1:T})$$

and an epigraph variable s_h satisfying

$$(s_h, 1, \boldsymbol{q}_h) \in \mathcal{Q}_r^{m_h+2}.$$

For each conic branch, introduce

$$\boldsymbol{u}_g = \mathcal{A}_g(\boldsymbol{x}_{1:T})$$

and an epigraph variable t_g satisfying

$$(\boldsymbol{u}_g, t_g) \in \mathcal{Q}^{k_g+1}.$$

Here $m_h = \dim(\boldsymbol{q}_h)$ and $k_g = \dim(\boldsymbol{u}_g)$.

Proposition 10 (Recurrent SOC-ICNN as an SOCP value function)**.** *For every finite horizon T and every sequence input $\boldsymbol{x}_{1:T}$, the recurrent SOC-ICNN in* (26) *is equal to the optimal value of the following SOCP:*

$$\begin{aligned} f_{\text{SOC-RNN}}(\boldsymbol{x}_{1:T}) = \min_{\eta} \quad & \boldsymbol{c}^\top \boldsymbol{z}_T + \sum_{t=1}^{T} \boldsymbol{v}_t^\top \boldsymbol{x}_t + b_0 + \sum_{h=1}^{H} \alpha_h s_h + \sum_{g=1}^{G} \lambda_g t_g \\ \text{s.t.} \quad & \boldsymbol{z}_t \geq W_t \boldsymbol{x}_t + R_t \boldsymbol{z}_{t-1} + \boldsymbol{b}_t, \quad \boldsymbol{z}_t \geq \boldsymbol{0}, \quad t = 1, \dots, T, \\ & \boldsymbol{z}_0 = \boldsymbol{0}, \\ & \boldsymbol{q}_h = \sum_{t=1}^{T} B_{h,t} \boldsymbol{x}_t + \boldsymbol{e}_h, \quad (s_h, 1, \boldsymbol{q}_h) \in \mathcal{Q}_r^{m_h+2}, \quad h = 1, \dots, H, \\ & \boldsymbol{u}_g = \sum_{t=1}^{T} A_{g,t} \boldsymbol{x}_t + \boldsymbol{d}_g, \quad (\boldsymbol{u}_g, t_g) \in \mathcal{Q}^{k_g+1}, \quad g = 1, \dots, G, \end{aligned} \tag{27}$$

where

$$\eta = \{\boldsymbol{z}_t, \boldsymbol{q}_h, s_h, \boldsymbol{u}_g, t_g : \ t = 1, \dots, T, \ h = 1, \dots, H, \ g = 1, \dots, G\}.$$

Proof. The proof is identical in structure to Theorem 1. Let $V(\boldsymbol{x}_{1:T})$ denote the optimal value of (27). For the upper bound, choose the recurrent forward activations $\bar{\boldsymbol{z}}_t$, set

$$\bar{\boldsymbol{q}}_h = \sum_{t=1}^{T} B_{h,t} \boldsymbol{x}_t + \boldsymbol{e}_h, \qquad \bar{s}_h = \frac{1}{2} \|\bar{\boldsymbol{q}}_h\|_2^2,$$

and

$$\bar{\boldsymbol{u}}_g = \sum_{t=1}^{T} A_{g,t} \boldsymbol{x}_t + \boldsymbol{d}_g, \qquad \bar{t}_g = \|\bar{\boldsymbol{u}}_g\|_2.$$

This point is feasible for (27) and attains exactly the value in (26). Hence $V(\boldsymbol{x}_{1:T}) \leq f_{\text{SOC-RNN}}(\boldsymbol{x}_{1:T})$.

For the lower bound, any feasible point satisfies the LP lower bound from (25). The rotated-cone constraints imply

$$s_h \geq \frac{1}{2} \left\| \sum_{t=1}^{T} B_{h,t} \boldsymbol{x}_t + \boldsymbol{e}_h \right\|_2^2,$$

and the standard-cone constraints imply

$$t_g \geq \left\| \sum_{t=1}^{T} A_{g,t} \boldsymbol{x}_t + \boldsymbol{d}_g \right\|_2.$$

Multiplying by $\alpha_h, \lambda_g \geq 0$ and summing gives $V(\boldsymbol{x}_{1:T}) \geq f_{\text{SOC-RNN}}(\boldsymbol{x}_{1:T})$. Combining both inequalities proves equality. □

This proposition shows that the recurrent extension preserves the same mathematical backbone as the fully connected and convolutional SOC-ICNNs: after finite-horizon unrolling, the ReLU recurrent part is an LP value function, and the quadratic/conic branches lift it to an SOCP value function. The extension is therefore architectural rather than conceptual; it adds temporal parameter sharing and recurrent structure without leaving the LP-to-SOCP value-function framework.

D.6 Recurrent Downstream Test on Trajectory Smoothing

To further test the finite-horizon recurrent SOC-ICNN extension, we consider a downstream trajectory-smoothing problem. The goal of this experiment is to verify that the recurrent LP-to-SOCP lifting remains useful when the decision variable is a temporal sequence rather than a static vector or image. The recurrent ReLU baseline follows the input-convex construction principle of ICNNs [Amos et al., 2017], with nonnegative hidden-to-hidden recurrent weights to preserve convexity with respect to the decision sequence.

Task. For each reference trajectory $R = (r_1, \dots, r_T)$, the decision variable is a smoothed trajectory $U = (u_1, \dots, u_T)$, where $u_t \in \mathbb{R}^p$. We use $T = 40$ and $p = 4$. The true convex sequence energy is

$$\begin{aligned} F_{\mathrm{seq}}(U; R) =& \frac{\rho}{2T} \sum_{t=1}^{T} \|u_t - r_t\|_2^2 + \frac{\beta}{2(T-2)} \sum_{t=2}^{T-1} \|u_{t+1} - 2u_t + u_{t-1}\|_2^2 \\ &+ \frac{\lambda}{T-1} \sum_{t=2}^{T} \sqrt{\|u_t - u_{t-1}\|_2^2 + \varepsilon} + \frac{\gamma}{2T} \sum_{t=1}^{T} \|u_t\|_2^2, \end{aligned}$$

with box constraints $u_t \in [-1, 1]^p$. The first term tracks the reference sequence, the second term penalizes acceleration, the third term is a TV-like conic switching cost, and the last term is a weak magnitude regularizer. This objective naturally matches the SOC-RNN architecture: the quadratic branch models temporal curvature, while the conic branch models switching costs.

Models. We compare five recurrent convex surrogates: RNN-ReLU, RNN-Softplus, RNN-Quad, RNN-Norm, and RNN-SOC. All models use the same recurrent ICNN backbone with hidden dimension 20. The RNN-Quad and RNN-Norm variants add only the temporal quadratic or temporal conic branch, respectively, while RNN-SOC includes both branches. The quadratic branch uses 24 temporal convolutional channels, and the conic branch uses 12 two-dimensional norm groups. The hidden-to-hidden recurrent weights in the ICNN backbone are constrained to be elementwise nonnegative.

Data and optimization protocol. We generate 1200 training, 200 validation, and 200 test reference trajectories. Each reference trajectory combines sinusoidal components, low-frequency drift, and several piecewise offsets. For each trajectory, we generate 14 candidate decision sequences and evaluate the true convex energy to train the surrogate by mean-squared error regression. All models are trained for 150 epochs using AdamW with learning rate 2×10^{-3}, batch size 256, and weight decay 10^{-6}. For downstream evaluation, each learned surrogate is optimized over $U \in [-1, 1]^{T \times p}$ using projected Adam for 400 steps and 4 random restarts. The approximate reference solution $U^\star$ is computed by approximately minimizing the true convex sequence energy using projected Adam for 900 steps and the same box projection. We report relative prediction error, true-energy regret $F_{\mathrm{seq}}(\hat{U}; R) - F_{\mathrm{seq}}(U^\star; R)$, decision error $\|\hat{U} - U^\star\|_F / \sqrt{Tp}$, and downstream inference time.

Implementation. The experiment is implemented in Python 3.11 with PyTorch 2.0 and run on a single NVIDIA RTX 4060 Ti GPU. All random seeds are fixed to 0. The energy parameters are $\rho = 1.0$, $\beta = 0.20$, $\lambda = 0.18$, $\gamma = 0.02$, and $\varepsilon = 10^{-5}$.

Table 20: Recurrent downstream trajectory-smoothing test. Regret and decision error are evaluated using the true convex sequence energy after optimizing each learned recurrent surrogate. Lower is better for RelErr, regret, and decision error.

Model	RelErr	Regret	Decision error	Infer. (ms)
RNN-ReLU	0.1920	0.4515 ± 0.0664	0.5458 ± 0.0320	405.98
RNN-Softplus	0.2359	0.4150 ± 0.0676	0.5528 ± 0.0335	547.33
RNN-Quad	0.0914	0.0697 ± 0.0152	0.3094 ± 0.0433	381.80
RNN-Norm	0.1154	0.0903 ± 0.0177	0.3708 ± 0.0407	554.24
RNN-SOC	**0.0671**	$\mathbf{0.0524} \pm 0.0156$	$\mathbf{0.2810} \pm 0.0489$	431.76

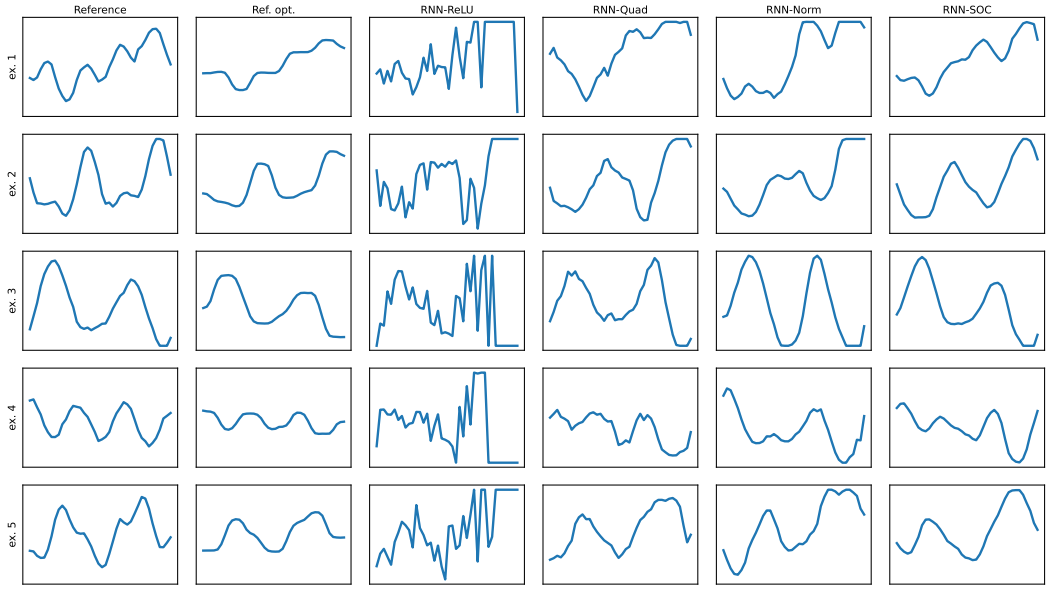

Figure 3: Qualitative downstream trajectory-smoothing examples. Each row is a test instance, and each column shows one trajectory component of the optimized decision. RNN-ReLU produces oscillatory and frequently clipped trajectories, while the structured Quad, Norm, and SOC variants better recover the reference optimum. RNN-SOC gives the lowest relative prediction error, true-energy regret, and decision error in Table 20.

E Structural Budgets and Allocation

Quadratic grouping. For nonnegative α_h, define

$$\bar{B} = \begin{bmatrix} \sqrt{\alpha_1} B_1 \\ \vdots \\ \sqrt{\alpha_H} B_H \end{bmatrix}, \qquad \bar{e} = \begin{bmatrix} \sqrt{\alpha_1} e_1 \\ \vdots \\ \sqrt{\alpha_H} e_H \end{bmatrix}.$$

Then $\sum_h \frac{\alpha_h}{2} \|B_h x + e_h\|_2^2 = \frac{1}{2}\|\bar{B}x + \bar{e}\|_2^2$, and its Hessian is $\bar{B}^\top \bar{B}$. With $r_Q = \sum_h m_h$, the Hessian rank is at most $\min(d_0, r_Q)$. Once $r_Q \geq d_0$, all PSD Hessians are representable; increasing width can still change optimization behavior. We recommend $H = 1$ as a canonical representation, while retaining the configurations actually used in the original experiments. This representation equivalence does not imply identical training dynamics or weight-decay penalties under different parameterizations.

We verify the identity at $d_0 = r_Q = 32$, fixing the affine parameter count at $32(32 + 1) = 1056$ and regrouping the same affine map. Table 21 reports floating-point discrepancies, not independently trained-model differences.

Table 21: Quadratic regrouping at fixed total width.

$H \times k$	Relative value error	Relative gradient error
1×32	0	0
2×16	1.09×10^{-16}	1.48×10^{-16}
4×8	1.06×10^{-16}	1.64×10^{-16}
8×4	1.11×10^{-16}	1.70×10^{-16}
16×2	1.24×10^{-16}	1.81×10^{-16}
32×1	1.79×10^{-16}	2.82×10^{-16}

Norm allocation. In general, $\sum_g \lambda_g \|A_g x + d_g\|_2$ cannot be replaced by a single norm of a stacked affine map. We fix $r_C = 32$ and a shared random feature map, vary (G, k) with $Gk = 32$, and fit nonnegative atom coefficients and a common intercept. Table 22 reports mean test NMSE over three seeds. The matched grouping reaches near-machine precision. The residual errors for mismatched groupings concern this fixed-feature experiment; they are not lower bounds for end-to-end trainable affine maps.

When groups are known, atom dimensions can follow them. Otherwise, both G and the widths should be selected on validation data under a fixed total budget; a balanced allocation $G \approx k \approx \sqrt{r_C}$ is only a starting heuristic, not a demonstrated universal optimum.

Table 22: Norm-allocation test with fixed affine features (mean NMSE over three seeds).

Target grouping	1×32	2×16	4×8	8×4	16×2	32×1
1×32	≈ 0	0.0078	0.0293	0.0680	0.1220	0.1796
4×8	0.0979	0.0432	≈ 0	0.0441	0.1063	0.1728
16×2	0.1986	0.1518	0.1130	0.0665	≈ 0	0.0759

F Structural-Branch Gradients and Deep-Backbone Stress Test

For $c_g(x) = \lambda_g \|A_g x + d_g\|_2$, let $u_g = A_g x + d_g$. When $u_g \neq 0$,

$$\nabla_{A_g} c_g = \lambda_g \frac{u_g}{\|u_g\|_2} x^\top, \qquad \|\nabla_{A_g} c_g\|_F \leq \lambda_g \|x\|_2.$$

At $u_g = 0$, the same bound holds for subgradients obtained from the unit ball of norm subgradients. For $q_h(x) = \frac{\alpha_h}{2}\|B_h x + e_h\|_2^2$,

$$\|\nabla_{B_h} q_h\|_F \leq \alpha_h (\|B_h\|_{\mathrm{op}} \|x\|_2 + \|e_h\|_2) \|x\|_2.$$

These are derivatives of the atom output, not bounds on the full training-loss gradient; the latter also depends on the loss derivative. They do not bound higher-order derivatives used in gradient-based objectives. Direct input-to-output branches avoid Jacobian products through the backbone, but do not guarantee stable training for arbitrary scales or initialization.

The stress test uses input dimension 3 and a 100-layer ReLU-ICNN backbone. Quad adds a width-5 quadratic branch (Hessian rank at most 3), Norm adds two width-3 atoms, and SOC adds both. Each model is trained with three seeds without gradient clipping. None of the 12 runs produces a non-finite update. Early backbone layers still exhibit gradient attenuation; nonzero structural-branch gradients do not resolve that issue. Table 23 summarizes the observed diagnostics. The result is empirical evidence for this configuration, not a general no-explosion guarantee.

Table 23: Deep-backbone stress test. Gradient columns summarize the reported gradient diagnostics; entries with uncertainty are mean $\pm$ standard deviation over three seeds.

Model	Params	Test NMSE	Global p95	Max. gradient	Branch p95	NaN/Inf
ReLU	13914	0.624 ± 0.003	0.708 ± 0.017	0.838 ± 0.017	–	0
Quad	13934	0.447 ± 0.014	0.716 ± 0.018	0.842 ± 0.017	0.310 ± 0.015	0
Norm	13938	0.509 ± 0.003	0.944 ± 0.030	1.178 ± 0.051	0.600 ± 0.022	0
SOC	13958	0.453 ± 0.005	0.998 ± 0.070	1.289 ± 0.166	0.683 ± 0.120	0

G Decision Tasks with Explicit Norm Structure

Quad is a useful simpler model when quadratic curvature dominates. The following experiments examine when explicit norm-type structure instead benefits the full SOC model. The model name "SOC" denotes the mixed quadratic-plus-norm architecture throughout this appendix.

G.1 Public-data group-regularized logistic regression

We use the Breast Cancer Wisconsin (Diagnostic) dataset distributed through `sklearn.datasets.load_breast_cancer`: 569 observations with 30 features. Each of ten morphological measurements has mean, standard-error, and worst-value features, defining ten three-dimensional groups. The surrogate approximates the empirical risk as a function of classifier parameters (w, b), rather than acting directly as an image or feature classifier:

$$F_{\lambda_G}(w,b) = \frac{1}{n}\sum_{i=1}^{n} \log(1+\exp[-y_i(a_i^\top w + b)]) + \lambda_G \sum_{j=1}^{10} \sqrt{\|w_{G_j}\|_2^2 + \varepsilon^2} + \frac{10^{-3}}{2}\|w\|_2^2.$$

Here $y_i \in \{-1, 1\}$. The decision dimension is 31. We use three stratified split seeds, with 75% of observations for training and 25% for testing, and $3200/700/1200$ surrogate training/validation/test queries. ReLU, Quad, Norm, and SOC have 1880, 1864, 1890, and 1874 parameters, respectively. We evaluate $\lambda_G \in \{0, 0.01, 0.03, 0.10\}$.

Table 24 reports the decision and classification metrics. Quad has the lowest mean normalized regret at $\lambda_G = 0$. With positive group regularization, SOC has the lowest mean regret and decision error in each tested setting; relative to Quad, the mean regret reductions are 35.3%, 65.9%, and 51.2%. These are reductions in the optimization objective's regret, not classification gains: for example, SOC has lower AUC at $\lambda_G = 0.01$. The classification metric is auxiliary to the surrogate-optimization task.

G.2 Robust convex regression

The target combines squared residuals, smoothed residual norms, group regularization, and ridge curvature:

$$f(w) = \frac{1-\eta}{2n}\sum_i \|C_i w - y_i\|_2^2 + \frac{\eta}{n}\sum_i \sqrt{\|C_i w - y_i\|_2^2 + \epsilon^2} + \lambda \sum_g \sqrt{\|R_g w\|_2^2 + \epsilon^2} + \frac{\mu}{2}\|w\|_2^2.$$

Table 25 compares quadratic, mixed-robust, and norm-dominant settings over three seeds. Quad has the lowest mean regret in the quadratic control; SOC has the lowest mean regret in the two settings

Table 24: Public-data group-regularized logistic task: mean ± standard deviation over three stratified split seeds. Regret and decision error are lower-is-better; AUC is higher-is-better.

λ_G	Model	Normalized regret	Decision error	ROC-AUC
0.00	ReLU	0.703 ± 0.095	1.967 ± 0.174	$\mathbf{0.979 \pm 0.006}$
0.00	Quad	$\mathbf{0.453 \pm 0.182}$	1.832 ± 0.149	0.969 ± 0.007
0.00	Norm	0.622 ± 0.151	1.733 ± 0.071	0.971 ± 0.021
0.00	SOC	0.597 ± 0.224	$\mathbf{1.530 \pm 0.043}$	0.978 ± 0.012
0.01	ReLU	0.589 ± 0.098	2.089 ± 0.373	$\mathbf{0.975 \pm 0.004}$
0.01	Quad	0.772 ± 0.244	1.858 ± 0.236	0.955 ± 0.021
0.01	Norm	1.006 ± 0.219	2.332 ± 0.205	0.972 ± 0.015
0.01	SOC	$\mathbf{0.500 \pm 0.086}$	$\mathbf{1.306 \pm 0.104}$	0.940 ± 0.051
0.03	ReLU	1.215 ± 0.191	2.307 ± 0.190	0.967 ± 0.007
0.03	Quad	0.875 ± 0.136	1.636 ± 0.130	0.915 ± 0.041
0.03	Norm	0.933 ± 0.158	1.994 ± 0.201	$\mathbf{0.979 \pm 0.005}$
0.03	SOC	$\mathbf{0.298 \pm 0.078}$	$\mathbf{0.889 \pm 0.158}$	0.976 ± 0.009
0.10	ReLU	1.754 ± 0.354	1.902 ± 0.288	$\mathbf{0.979 \pm 0.015}$
0.10	Quad	0.826 ± 0.030	0.973 ± 0.017	0.957 ± 0.005
0.10	Norm	1.798 ± 0.085	2.194 ± 0.118	0.978 ± 0.015
0.10	SOC	$\mathbf{0.403 \pm 0.185}$	$\mathbf{0.652 \pm 0.164}$	0.966 ± 0.030

containing norm-type structure. The between-seed variation is substantial, so the table supports a task-dependent interpretation rather than statistical significance or uniform dominance. In particular, Quad is not consistently better than ReLU in every regime.

Table 25: Robust convex regression: mean ± standard deviation over three seeds. All metrics are lower-is-better.

Setting	Model	Test NMSE	Normalized regret	Decision error
Quadratic	ReLU	0.0189 ± 0.0075	0.0714 ± 0.0309	0.530 ± 0.041
Quadratic	Quad	$\mathbf{0.0127 \pm 0.0065}$	$\mathbf{0.0469 \pm 0.0331}$	$\mathbf{0.316 \pm 0.110}$
Quadratic	SOC	0.0175 ± 0.0039	0.0648 ± 0.0127	0.336 ± 0.016
Mixed robust	ReLU	0.0193 ± 0.0087	0.1629 ± 0.0498	0.428 ± 0.044
Mixed robust	Quad	0.0193 ± 0.0049	0.1417 ± 0.0565	0.427 ± 0.140
Mixed robust	SOC	$\mathbf{0.0182 \pm 0.0068}$	$\mathbf{0.1070 \pm 0.0459}$	$\mathbf{0.369 \pm 0.120}$
Norm-dominant	ReLU	0.0170 ± 0.0042	0.1481 ± 0.0602	0.340 ± 0.075
Norm-dominant	Quad	0.0182 ± 0.0033	0.1581 ± 0.0616	0.328 ± 0.029
Norm-dominant	SOC	$\mathbf{0.0159 \pm 0.0020}$	$\mathbf{0.1321 \pm 0.0447}$	$\mathbf{0.312 \pm 0.041}$

Smooth norm primitives. The smoothed norm used in these tasks is itself a Euclidean norm in one additional coordinate:

$$\sqrt{\|Ax + d\|_2^2 + \varepsilon^2} = \left\| \begin{bmatrix} Ax + d \\ \varepsilon \end{bmatrix} \right\|_2 .$$

Thus the target terms are consistent with the conic primitive. A prescribed constant coordinate need not be trainable; any parameter-count comparison must distinguish fixed coordinates from learned parameters.

H Optimal-Transport Experiments

H.1 Controlled Brenier-map recovery

We construct the strongly convex potential

$$\phi^\star(x) = \tfrac{1}{2} x^\top Q x + \beta \sum_{g=1}^{G} \lambda_g \sqrt{\|A_g x + d_g\|_2^2 + \varepsilon^2}, \qquad Q \succ 0, \quad T^\star = \nabla \phi^\star.$$

Under the usual absolute-continuity and finite-second-moment conditions on the source, this gradient is the quadratic-cost optimal map from that source to its pushforward. The experiment is a controlled

recovery test with an analytical map, separate from the two-distribution minimax experiment below. Its target family deliberately isolates geometry represented by the structural branches.

Quad-Matched and SOC use the same shared backbone, shared-backbone initialization, and minibatch order, with matched additional parameter counts. Table 26 lists the six tested (d, β) combinations and mean errors over three seeds. At $\beta = 0$, the errors are nearly identical; adding norm structure favors SOC in every listed dimension. At $(d, \beta) = (64, 1.5)$, doubling training steps gives map NMSEs 0.04012 (Quad) and 0.03851 (SOC). This check suggests that merely doubling the Quad training budget does not close the observed gap; it does not establish convergence or an irreducible approximation gap.

Table 26: Controlled map recovery: mean map and norm-component NMSE over three seeds. Component error is not reported when $\beta = 0$. Only means are reported for this experiment.

d	β	Quad map	SOC map	Quad component	SOC component
8	0	4.4111×10^{-5}	4.4047×10^{-5}	–	–
8	0.75	0.01484	**0.01337**	0.07244	**0.06526**
8	1.5	0.02910	**0.02655**	0.07263	**0.06625**
16	1.5	0.03013	**0.02839**	0.07104	**0.06694**
32	1.5	0.03301	**0.03134**	0.07255	**0.06858**
64	1.5	0.04013	**0.03864**	0.08120	**0.07821**

H.2 Gaussian-to-eight-Gaussians transport

We adapt the benchmark of Makkuva et al. [2020], using source distribution $\mathcal{N}(0, I_2)$ and a target mixture of eight equally weighted Gaussians centered on a radius-5 circle, each with covariance $0.5^2 I_2$. The experiment retains the dual-potential minimax framework with transport $T(y) = \nabla g(y)$. The ICNN backbone has width 32 and depth 3; each outer iteration uses three g updates, for 8000 outer iterations and three random seeds. Per-potential parameter counts differ by less than 2% among the four ICNN variants.

Table 27 includes an affine moment baseline and a target-sampling reference. Sliced-W2 and MMD^2 measure distribution discrepancy; nearest-mode distance and inter-mode mass provide complementary geometric diagnostics. SOC attains the lowest mean values among the tested models, while remaining above the target-sampling floor. ReLU and Quad each have one seed with incomplete mode separation, contributing to their large variation. All ICNN models have zero observed monotonicity violations on the evaluated samples; this empirical diagnostic is distinct from verifying transport optimality.

Table 27: Eight-mode transport: mean $\pm$ standard deviation over three seeds. The target-sampling row is a finite-sample reference, not a trained method.

Model	Params/potential	Sliced-W2	MMD^2	Nearest mode	Inter-mode mass
Affine moment	0	1.045 ± 0.026	0.0213 ± 0.0018	2.325 ± 0.025	$73.9\% \pm 0.6\%$
ReLU-ICNN	2436	0.627 ± 0.306	0.0228 ± 0.0286	1.450 ± 0.427	$41.0\% \pm 20.8\%$
Quad-ICNN	2455	0.563 ± 0.388	0.0212 ± 0.0344	1.399 ± 0.525	$42.5\% \pm 20.3\%$
Norm-ICNN	2464	0.437 ± 0.096	0.0051 ± 0.0038	1.275 ± 0.103	$37.4\% \pm 7.7\%$
SOC-ICNN	2483	$\mathbf{0.363 \pm 0.069}$	$\mathbf{0.0022 \pm 0.0019}$	$\mathbf{1.196 \pm 0.076}$	$\mathbf{32.1\% \pm 5.0\%}$
Target sampling floor	–	0.138 ± 0.067	0.0002 ± 0.0000	0.619 ± 0.002	$1.0\% \pm 0.0\%$

I Nonconvex Model Mismatch and an Unconstrained MLP

We form targets

$$f_\rho(x) = f_{\text{cvx}}(x) + \frac{\rho}{J} \sum_j \sin(\omega a_j^\top x + b_j), \qquad x \in [-2, 2]^{10},$$

and compare parameter-matched MLP, ReLU, Quad, and SOC models. Table 28 reports pointwise and gradient errors for four perturbation strengths. The reported negative-curvature fraction increases from 0 to 0.743. The unconstrained MLP has the lowest mean errors in every tested setting, including

the convex control. SOC cannot represent negative curvature, and neither structural enrichment nor convexity alone guarantees the best finite-budget fit.

For squared loss, with $m(x) = \mathbb{E}[Y \mid X = x]$, the risk decomposes as

$$\mathbb{E}[(f(X) - Y)^2] = \mathbb{E}[\mathrm{Var}(Y \mid X)] + \|f - m\|^2_{L_2(P_X)}.$$

Restricting f to a convex function class can therefore introduce a nonzero approximation floor. If projection language is used, existence requires an appropriate nonempty closed convex subset of $L_2(P_X)$; a finite-network class should not automatically be identified with such a subset. With the same retained backbone, setting norm coefficients to zero embeds Quad in SOC. This gives a non-strict inequality between optimal risks over nested classes, but does not guarantee an improvement for every target, matched-total-budget architecture, or trained model.

Table 28: Nonconvex-target comparison: reported means $\pm$ standard deviations over three seeds. Lower errors are better. The curvature column describes the target.

ρ	Neg. curvature	Model	Test NMSE	Gradient rel. RMSE
0.0	0.000	MLP	$\mathbf{0.00264 \pm 0.00015}$	$\mathbf{0.0676 \pm 0.0017}$
0.0	0.000	ReLU	0.02510 ± 0.0042	0.2924 ± 0.0057
0.0	0.000	Quad	0.01042 ± 0.0112	0.1399 ± 0.100
0.0	0.000	SOC	0.01436 ± 0.0133	0.1691 ± 0.085
0.3	0.076	MLP	$\mathbf{0.00763 \pm 0.00038}$	$\mathbf{0.1240 \pm 0.0014}$
0.3	0.076	ReLU	0.02733 ± 0.0035	0.3001 ± 0.0055
0.3	0.076	Quad	0.01548 ± 0.0105	0.1812 ± 0.081
0.3	0.076	SOC	0.01975 ± 0.0127	0.2038 ± 0.072
0.5	0.520	MLP	$\mathbf{0.01520 \pm 0.00056}$	$\mathbf{0.1776 \pm 0.0018}$
0.5	0.520	ReLU	0.02983 ± 0.0048	0.3083 ± 0.015
0.5	0.520	Quad	0.02480 ± 0.0083	0.2306 ± 0.059
0.5	0.520	SOC	0.02862 ± 0.0109	0.2472 ± 0.056
0.7	0.743	MLP	$\mathbf{0.02458 \pm 0.0017}$	$\mathbf{0.2255 \pm 0.0008}$
0.7	0.743	ReLU	0.03170 ± 0.0064	0.3225 ± 0.016
0.7	0.743	Quad	0.03832 ± 0.0047	0.2832 ± 0.041
0.7	0.743	SOC	0.04174 ± 0.0086	0.2958 ± 0.039

J Code and Data Availability

The code and data needed to reproduce the experiments will be publicly released at `https://github.com/Kanyooo/SOC-ICNN` with the camera-ready publication. The release will include reproduction instructions and the scripts for synthetic-data generation and public-dataset loading. The public-data experiments use Breast Cancer Wisconsin (Diagnostic) and Olivetti Faces through scikit-learn; third-party datasets remain subject to their original terms of use.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main claims: ReLU-ICNNs can be interpreted as LP value-function networks, SOC-ICNNs lift this structure to SOCP value functions by adding quadratic and conic primitives, and this yields a strict representational extension without increasing the asymptotic order of forward-pass complexity. These claims are supported by the theoretical results in Sections 3–4 and by the experiments in Section 5 and the appendices.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The conclusion discusses convex-target assumptions, the scope of the SOCP representation, target-dependent Norm benefits, finite-budget training, the conditional residual decomposition, limited seed counts, and deep-backbone gradient attenuation; Appendix I quantifies nonconvex mismatch.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: The theoretical statements specify the architectural assumptions required for input convexity, including nonnegative hidden-to-hidden weights and nonnegative output or branch scaling coefficients. The main text provides proof sketches, and complete proofs for the LP value-function representation, SOCP value-function representation, CPWL structure, approximation lower bound, strict extension, structural absorption, and complexity analysis are provided in the appendix.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [No]

   Justification: The paper specifies the model, theoretical assumptions, benchmark objectives, and the principal evaluation protocols. Some implementation-level details of the additional experiments are not fully enumerated in the paper; the code and reproduction instructions will accompany the camera-ready release (Appendix J).

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code and data needed to reproduce the experiments will be publicly released with the camera-ready publication at `https://github.com/Kanyooo/SOC-ICNN`, together with reproduction instructions, synthetic-data generation scripts, and public-dataset loading scripts; see Appendix J.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [No]

Justification: The appendices describe the benchmark tasks, model budgets, data splits, and reported training and evaluation settings. They do not enumerate every hyperparameter and implementation detail for all additional experiments; these details will be provided through the accompanying code release.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The principal empirical comparisons report means and standard deviations, with aggregation described in the corresponding experimental sections and captions. The three-seed controlled Brenier and fixed-feature Norm-allocation tables report means only; no statistical-significance claim is made for these comparisons. The CNN/RNN tests use one training seed and do not estimate between-training-seed variability.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: The paper identifies the software and GPU environment for the structured experiments and reports runtime measurements for the original benchmarks. It does not provide a complete per-experiment accounting of memory usage and execution time for all additional experiments.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is methodological research on convex neural architectures and optimization-based surrogate modeling. The experiments use synthetic data, generated convex optimization tasks, and a public benchmark dataset accessed through standard interfaces, and the paper does not involve deceptive, privacy-invasive, or harmful data collection.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The work may have positive impact by improving the expressiveness and transparency of convex surrogate models used in optimization-based decision pipelines. The paper also acknowledges that, as with other learned surrogate models, deployment in high-stakes decision settings would require careful validation of the learned objective, downstream constraints, and domain-specific assumptions.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release high-risk pretrained models, generative models, scraped datasets, or dual-use foundation models. The released assets are experimental code for convex neural architectures and do not require special misuse safeguards beyond standard research-code documentation.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [No]

Justification: The paper credits prior methods and identifies the public datasets and software interfaces used. It does not enumerate the licenses and versions of all third-party assets. Public datasets remain subject to their original terms of use, as noted in Appendix J.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The new asset is the SOC-ICNN implementation and its experimental scripts; no new dataset is introduced. Reproduction instructions will accompany the public camera-ready release at the repository in Appendix J.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, surveys, or newly collected human-subject data. The real-image appendix experiment uses an existing public benchmark dataset only as input data for a convex image restoration test.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not conduct human-subject research or collect new data from participants. Therefore, IRB approval or an equivalent review is not applicable.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not a component of the research method or experimental pipeline. They were used for language polishing and manuscript editing, as voluntarily disclosed before the references. The authors take responsibility for the scientific content and final manuscript.