A Unified Neural Architecture for Variable-Wise Shape Constraints

Hyunho Kim
KAIST
hyunho.kim@kaist.ac.kr

Dohyun Bu
KAIST
dohyun.bu@kaist.ac.kr

Jong-Seok Lee*
KAIST
jongseok.lee@kaist.ac.kr

Abstract

We introduce COMONet (Convex-Concave and Monotonicity Constrained Neural Networks), a unified architecture for enforcing variable-wise shape constraints in neural networks. COMONet addresses the limitations of prior methods that either support only a restricted subset of monotonicity and curvature constraints or enforce them through penalties without strict architectural guarantees. Our framework assigns each input variable to one of eight shape regimes, covering monotonicity, convexity, concavity, and their combinations. This is achieved through a partially connected architecture that routes each variable to specialized units equipped with sign-constrained weights and constraint-preserving activation functions. We also provide theoretical guarantees showing that COMONet satisfies the prescribed variable-wise shape constraints by construction. Experiments on synthetic and real-world datasets demonstrate that COMONet achieves competitive performance and remains robust to noise while preserving its architectural guarantees. COMONet thus offers a practical and principled framework for incorporating domain knowledge as shape constraints into neural network training. The code is available at https://github.com/retna319/COMONet.

1 Introduction

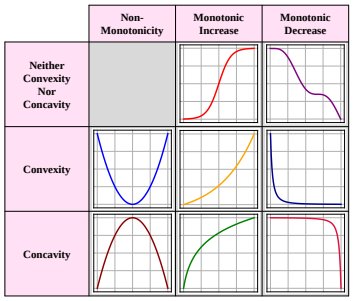
Neural networks trained solely by empirical risk minimization may fail to reflect known domain knowledge [18, 15, 35]. Domain knowledge refers to pre-established information in a given field. In supervised learning, it often appears as known relationships between input and output variables. Incorporating such relationships into neural networks can improve the trustworthiness and interpretability of learned models [56, 34]. The need for trustworthy and interpretable models is particularly important in high-stakes domains such as finance [16, 37], healthcare [45], and law [2], where model predictions can directly influence financial, clinical, or legal decisions, and violations of known relationships may result in risk-sensitive decisions. Therefore, there is a growing need for neural architectures that can incorporate domain knowledge while preserving the flexibility of data-driven learning.

Among various possible constraints, monotonicity and convexity (or concavity) are two representative shape constraints that serve as inductive biases and are widely applied across various domains [3, 29]. Monotonicity refers to a property where the output consistently non-decreases or non-increases[1] as the input increases. Meanwhile, convexity (or concavity) describes a function where, for any two points, the output does not exceed (does not fall below) the straight line connecting them, indicating an increasing (decreasing) rate of change [7]. Fig. 1(a) illustrates various types of shape constraints related to monotonic increase (decrease) and convexity (or concavity) that can be incorporated into a

*Corresponding author.

[1]For readability, we refer to non-decreasing as increasing and non-increasing as decreasing throughout the rest of the paper.

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



(a)

Certified MNN	✓	✓	✗	✗	✗	✗	✗	✗	✓*
COMET	✓	✓	✗	✗	✗	✗	✗	✗	✓*
HLL	✓	✓	✗	✗	✗	✗	✗	✗	✓
LMN	✓	✓	✗	✗	✗	✗	✗	✗	✓
Constrained MNN	✓	✓	✗	✗	✓	✓	✓	✓	✓
SMNN	✓	✓	✗	✗	✓	✓	✓	✓	✓
ICNN	✗	✗	✓	✓	✗	✗	✗	✗	✓
SCNN	✗	✗	✓	✓	✓	✓	✓	✓	✓
Shape-Constrained Lattice	✓	✓	✓	✓	✓	✓	✓	✓	✓*
PenDer	✓	✓	✓	✓	✓	✓	✓	✓	✗
Proposed (Ours)	✓	✓	✓	✓	✓	✓	✓	✓	✓

Constraint Guarantee

(b)

Figure 1: (a) Types of shape constraints. (b) Capability comparison with benchmark methods ($*$ Indicates structural guarantees that may not hold in practice)

model. It is important to note that monotonicity does not necessarily imply convexity or concavity, and convex or concave functions can be monotonically increasing, decreasing, or non-monotonic.

Recent studies have shown growing interest in monotonic neural networks [32, 46, 43, 29] and convex neural networks [3, 23], motivating various efforts to embed such properties into neural architectures. While substantial progress has been made in enforcing these constraints individually, methods that guarantee variable-wise monotonicity and curvature constraints within a unified neural architecture remain limited. Although several prior works attempt to incorporate multiple constraints, they often fail to cover all regimes or to provide strict architectural guarantees, as illustrated in Fig. 1.

These limitations suggest that enforcing mixed shape constraints across different subsets of variables remains challenging for neural networks. In practice, such mixed constraints naturally arise in a wide range of engineering applications, including HVAC control [58], concrete strength prediction [54], network resource allocation [28], portfolio optimization [33], asset pricing [21, 17, 8], and physical dynamics [20, 48, 5]. For instance, in HVAC systems [58], different variables exhibit heterogeneous curvature properties. Fan power increases convexly with airflow due to the nonlinear growth in energy consumption, whereas thermal comfort penalties are typically concave in the supply temperature, reflecting diminishing sensitivity. When airflow is adjusted as a function of temperature, these components interact, resulting in a mixed curvature structure within a single system. In such applications, satisfying these constraints simultaneously is important when learned models are used in downstream optimization or decision-making. Violations of known relationships may lead to implausible predictions or suboptimal decisions.

To address these issues, we propose COMONet, a unified neural architecture for enforcing variable-wise shape constraints within a single model. It adopts a partially connected structure that routes input variables to specialized units with sign-constrained weights and constraint-preserving activation functions. This architecture effectively learns and strictly enforces diverse shape constraints, overcoming limitations of existing approaches. Furthermore, depending on the types and composition of shape constraints that the entire set of input variables must satisfy, COMONet can flexibly utilize either its full structure or only a partial configuration. This flexibility allows the model to enforce monotonicity, convexity (or concavity), or their combination as needed, ensuring strict compliance with the specified constraints.

2 Related Work

Monotonic neural networks: Research on monotonic neural networks can be broadly categorized into two groups, regularization-based approaches and architecture-based approaches. The first group enforces monotonicity using various regularization techniques. For example, Certified MNN [32] applies penalties to partial derivatives, while COMET [46] augments the dataset with so-called counter-examples for instances that violate monotonicity. These approaches have limitations, as they may not fully enforce monotonicity without strong regularization and rely on external solvers such as MILP [24] and SMT [6]. The second group consists of hand-designed neural network architectures that inherently guarantee monotonicity. Methods such as HLL [52], LMN [38], Constrained MNN [43], and SMNN [29] belong to this category. While some of these approaches are theoretically proven to ensure monotonicity, their restricted structures can lead to reduced predictive performance.

Convex neural networks: Convex neural networks, pioneered by ICNN [3], ensure convexity with respect to inputs and have been widely studied for optimization and control-related applications [9, 53]. Building upon this framework, various extensions have been proposed to improve expressiveness and applicability. For instance, some methods approximate more complex functions by decomposing them into convex and concave components [44], while others adapt ICNNs for optimization tasks [59], adversarial robustness [40], or faster training through improved initialization [25].

Shape constrained neural networks: Recent studies also explore combining convexity with monotonicity. Among existing approaches, PenDer [22] and the shape-constrained models of Gupta et al. [23], including SCNN and lattice-based formulations, most closely support the full set of constraints in Fig. 1(a). PenDer relies on regularization, encouraging but not guaranteeing constraint satisfaction, and therefore requires post hoc detection of violations. SCNN cannot represent monotonic yet non-convex functions such as $x + \sin(x)$, nor can it handle settings in which convex and concave variables coexist within the same model. SCL enforces convexity, concavity, and monotonicity through a discretized lattice parameterization. Although the lattice structure can theoretically satisfy these constraints, its resolution induces a trade-off between computational cost and approximation fidelity. As the resolution increases, the number of linear inequality constraints grows rapidly, so SCL implementations often rely on stochastic constraint sampling [10], projecting only a subset of constraints at each iteration. Further, constraint satisfaction may not be guaranteed throughout training. Moreover, lattice interpolation supports per-variable constraints but cannot model joint convex or joint concave curvature, which is essential in many optimization or curvature-sensitive tasks [23]. These limitations motivate a unified, end-to-end training framework that guarantees all shape constraints without approximate projections or post-hoc verification.

3 Shape Constraints

We consider a continuous, multivariate function $f\colon [0,1]^d \to \mathbb{R}$, not necessarily differentiable. We consider three classes of local shape constraints *partial (joint) convexity*, *partial (joint) concavity* and *partial monotonicity* that apply to subsets of the input coordinates. Let $[d] = \{1, \dots, d\}$ and we denote vectors in $\mathbb{R}^d$ by bold lowercase letters (e.g., $\mathbf{x}$, $\mathbf{t}$) and index-sets by calligraphic uppercase letters (e.g., $\mathcal{CV}$, $\mathcal{MN}$). To impose distinct shape constraints on different input dimensions, we partition the coordinates of $\mathbf{x} \in \mathbb{R}^d$ into six disjoint groups:

$$\mathbf{x} = \big(\mathbf{x}_{cv},\, \mathbf{x}_{mv},\, \mathbf{x}_{cc},\, \mathbf{x}_{mc},\, \mathbf{x}_{mn},\, \mathbf{x}_u\big) \in \mathbb{R}^{|\mathcal{CV}|} \times \mathbb{R}^{|\mathcal{MV}|} \times \mathbb{R}^{|\mathcal{CC}|} \times \mathbb{R}^{|\mathcal{MC}|} \times \mathbb{R}^{|\mathcal{MN}|} \times \mathbb{R}^{|\mathcal{U}|}.$$

where $\mathcal{CV} \cup \mathcal{MV} \cup \mathcal{CC} \cup \mathcal{MC} \cup \mathcal{MN} \cup \mathcal{U} = [d]$. Each index-set enforces a particular constraint on f: $\mathcal{CV}$ (convex only), $\mathcal{MV}$ (both monotonic and convex), $\mathcal{CC}$ (concave only), $\mathcal{MC}$ (both monotonic and concave), $\mathcal{MN}$ (monotonic only), and $\mathcal{U}$ (unconstrained). Further, $\mathcal{V} = \mathcal{CV} \cup \mathcal{MV}$, $\mathcal{C} = \mathcal{CC} \cup \mathcal{MC}$, $\mathcal{M} = \mathcal{MN} \cup \mathcal{MV} \cup \mathcal{MC}$ represent the sets containing all convex, concave, and monotonic variables, respectively.

Definition 3.1 (Partial convexity and partial joint convexity)**.** Partition $\mathbf{x} = (\mathbf{x}_v, \mathbf{x}_{\neg v}) \in \mathbb{R}^d$ with $\mathbf{x}_v \in \mathbb{R}^{|\mathcal{V}|}$ and $\mathbf{x}_{\neg v} \in \mathbb{R}^{d-|\mathcal{V}|}$. For each coordinate $i \in \mathcal{V}$, write $\mathbf{x} = (x_i, \mathbf{x}_{\neg i})$, where $\mathbf{x}_{\neg i}$ denotes all coordinates except x_i. We say that f is *partially convex* in coordinate x_i iff, for any fixed $\mathbf{x}_{\neg i}$, any $x_i, x_i' \in \mathbb{R}$, and any $\lambda \in [0,1]$, $f(\lambda x_i + (1-\lambda)x_i', \mathbf{x}_{\neg i}) \;\le\; \lambda\, f(x_i, \mathbf{x}_{\neg i}) + (1-\lambda)\, f(x_i', \mathbf{x}_{\neg i})$. Further, we say f is *partially joint convex* on $\mathbf{x}_v$ iff, for any fixed $\mathbf{x}_{\neg v}$ and all $\mathbf{x}_v, \mathbf{x}_v' \in \mathbb{R}^{|\mathcal{V}|}$ and $\lambda \in [0,1]$, $f\big(\lambda\, \mathbf{x}_v + (1-\lambda)\, \mathbf{x}_v', \mathbf{x}_{\neg v}\big) \;\le\; \lambda\, f(\mathbf{x}_v, \mathbf{x}_{\neg v}) + (1-\lambda)\, f(\mathbf{x}_v', \mathbf{x}_{\neg v})$.

Definition 3.2 (Partial concavity and partial joint concavity)**.** Partition $\mathbf{x} = (\mathbf{x}_c, \mathbf{x}_{\neg c}) \in \mathbb{R}^d$ with $\mathbf{x}_c \in \mathbb{R}^{|\mathcal{C}|}$ and $\mathbf{x}_{\neg c} \in \mathbb{R}^{d-|\mathcal{C}|}$. For each coordinate $i \in \mathcal{C}$, write $\mathbf{x} = (x_i, \mathbf{x}_{\neg i})$, where $\mathbf{x}_{\neg i}$ denotes all coordinates except x_i. We say that f is *partially concave* in coordinate x_i iff, for any fixed $\mathbf{x}_{\neg i}$, any $x_i, x_i' \in \mathbb{R}$, and any $\lambda \in [0,1]$, $f(\lambda x_i + (1-\lambda)x_i', \mathbf{x}_{\neg i}) \;\ge\; \lambda\, f(x_i, \mathbf{x}_{\neg i}) + (1-\lambda)\, f(x_i', \mathbf{x}_{\neg i})$. Further, we say f is *partially joint concave* on $\mathbf{x}_c$ iff, for any fixed $\mathbf{x}_{\neg c}$, any $\mathbf{x}_c, \mathbf{x}_c' \in \mathbb{R}^{|\mathcal{C}|}$, and any $\lambda \in [0,1]$, $f\big(\lambda\, \mathbf{x}_c + (1-\lambda)\, \mathbf{x}_c', \mathbf{x}_{\neg c}\big) \;\ge\; \lambda\, f(\mathbf{x}_c, \mathbf{x}_{\neg c}) + (1-\lambda)\, f(\mathbf{x}_c', \mathbf{x}_{\neg c})$.

Definition 3.3 (Partial monotonicity)**.** Partition $\mathbf{x} = (\mathbf{x}_m, \mathbf{x}_{\neg m}) \in \mathbb{R}^d$ with $\mathbf{x}_m \in \mathbb{R}^{|\mathcal{M}|}$ and $\mathbf{x}_{\neg m} \in \mathbb{R}^{d-|\mathcal{M}|}$. For each coordinate $i \in \mathcal{M}$, write $\mathbf{x} = (x_i, \mathbf{x}_{\neg i})$, where $\mathbf{x}_{\neg i}$ denotes all coordinates except x_i. We say that f is *partially monotonic increasing* in coordinate x_i iff, for any fixed $\mathbf{x}_{\neg i}$, $x_i \le x_i' \implies f(x_i, \mathbf{x}_{\neg i}) \le f(x_i', \mathbf{x}_{\neg i})$. Similarly, f is *partially monotonic decreasing* in coordinate x_i iff $x_i \le x_i' \implies f(x_i, \mathbf{x}_{\neg i}) \ge f(x_i', \mathbf{x}_{\neg i})$. (For monotonicity, enforcing monotonicity on each variable individually guarantees joint monotonicity.)

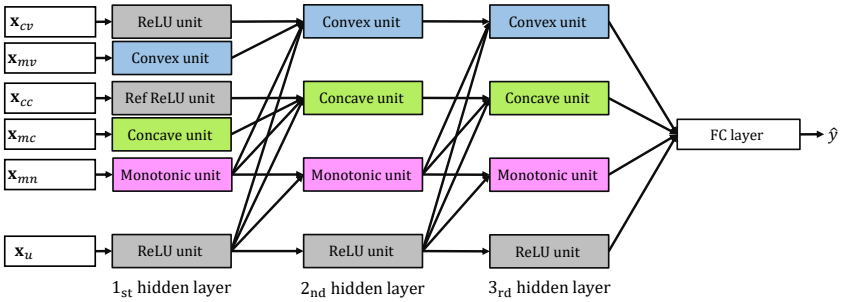


Figure 2: Structure of COMONet that has 3 hidden layers.

By combining Definitions 3.1-3.3, we can specify all shape-constraint types depicted in Fig. 1(a).

4 Proposed Method

Component units of COMONet: A key aspect of our approach is employing five distinct unit types to effectively integrate and enforce diverse shape constraints. These units incorporate either exponentiated or conventional weights, using ReLU or ReLU-n [31] as activation functions. For any scalar z, we define ReLU $= (z)_+ = \max(0, z)$ and ReLU-n $= (z)_+^n = \min\{n, \max(0, z)\}$. The equations below define the five units, each of which takes the vector $\mathbf{t}$ as input:

$$convex\ unit \coloneqq h_{\text{conv}}(\mathbf{t}) = \left(\exp(\mathbf{W})^\top \mathbf{t} + \mathbf{b}\right)_+ \tag{1}$$

$$concave\ unit \coloneqq h_{\text{conc}}(\mathbf{t}) = -\left(-\left(\exp(\mathbf{W})^\top \mathbf{t} + \mathbf{b}\right)\right)_+ \tag{2}$$

$$monotonic\ unit \coloneqq h_{\text{mono}}(\mathbf{t}) = \left(\exp(\mathbf{W})^\top \mathbf{t} + \mathbf{b}\right)_+^n \tag{3}$$

$$relu\ unit \coloneqq h_{\text{relu}}(\mathbf{t}) = \left(\mathbf{W}^\top \mathbf{t} + \mathbf{b}\right)_+ \tag{4}$$

$$ref\text{-}relu\ unit \coloneqq h_{\text{ref-relu}}(\mathbf{t}) = -\left(-\left(\mathbf{W}^\top \mathbf{t} + \mathbf{b}\right)\right)_+ \tag{5}$$

h_{conv}, h_{conc} and h_{mono} utilize exponentiated weights [57, 1, 14] to constrain reparametrized weights to be positive. h_{mono} employs $(z)_+^n$ which contains both convex and concave hinge components. Further, h_{conc} and $h_{\text{ref-relu}}$ employ point-symmetric variants of ReLU, $-(-(z))_+$ and ReLU-n, $-(-(z))_+^n$. We adopt ReLU and ReLU-n by default, using ReLU as our baseline for its computational efficiency, resilience to vanishing gradients, and piecewise linear sparsity that accelerates convergence and boosts generalization [36, 19]. As shown in Appendix G.1, any activation functions meeting the required characteristics can be used instead. Each component unit defined above satisfies the following properties, as formalized in the Lemmas below (Detailed proofs for Lemmas 4.1-4.5 are provided at Appendix A.):

Lemma 4.1. *Let* $h_{\text{conv}} : \mathbb{R}^d \to \mathbb{R}^k$ *and denote its* j*th coordinate by* $f_j(\mathbf{t}) \coloneqq \left[h_{\text{conv}}(\mathbf{t})\right]_j = \left(\exp(\mathbf{w}_j)^\top \mathbf{t} + b_j\right)_+$. *Then,* $\forall j \in [k]$, f_j *is jointly convex in* $\mathbf{t}$ *and* f_j *is coordinatewise increasing.*

Lemma 4.2. *Let* $h_{\text{relu}} : \mathbb{R}^d \to \mathbb{R}^k$ *and denote its* j*th coordinate by* $f_j(\mathbf{t}) \coloneqq \left[h_{\text{relu}}(\mathbf{t})\right]_j = \left(\mathbf{w}_j^\top \mathbf{t} + b_j\right)_+$. *Then,* $\forall j \in [k]$, f_j *is jointly convex in* $\mathbf{t}$.

Lemma 4.3. *Let* $h_{\text{conc}} : \mathbb{R}^d \to \mathbb{R}^k$ *and denote its* j*th coordinate by* $f_j(\mathbf{t}) \coloneqq \left[h_{\text{conc}}(\mathbf{t})\right]_j = -\left(-\left(\exp(\mathbf{w}_j)^\top \mathbf{t} + b_j\right)\right)_+$. *Then,* $\forall j \in [k]$, f_j *is jointly concave in* $\mathbf{t}$ *and coordinatewise increasing.*

Lemma 4.4. *Let* $h_{\text{ref-relu}} : \mathbb{R}^d \to \mathbb{R}^k$ *and denote its* j*th coordinate by* $f_j(\mathbf{t}) \coloneqq \left[h_{\text{ref-relu}}(\mathbf{t})\right]_j = -\left(-\left(\mathbf{w}_j^\top \mathbf{t} + b_j\right)\right)_+$. *Then,* $\forall j \in [k]$, f_j *is jointly concave in* $\mathbf{t}$.

Lemma 4.5. *Let* $h_{\text{mono}} : \mathbb{R}^d \to \mathbb{R}^k$ *and denote its* j*th coordinate by* $f_j(\mathbf{t}) \coloneqq \left[h_{\text{mono}}(\mathbf{t})\right]_j = \left(\exp(\mathbf{w}_j)^\top \mathbf{t} + b_j\right)_+^n$. *Then,* $\forall j \in [k]$, f_j *is coordinatewise increasing in* $\mathbf{t}$.

Network structure: The proposed architecture, illustrated in Fig. 2, adopts a selectively connected design that routes information through designated subsets of connections. This structure is conceptually similar to the architectures proposed in [3] and [29], which also utilize partially connected designs. Such a design preserves the specific properties of individual input variable groups while

allowing structured information flow among groups. It is worth noting that the proposed architecture forms a single unified neural network, rather than a collection of separate subnets. The component units serve as building blocks within this unified model, while the partially connected structure reduces the number of network parameters without requiring additional computational overhead. Let $h^{(i)}$ be the i-th hidden layer. Then the overall formulation of COMONet, which employs the five distinct types of units with depth l defined above, is presented below:

When $i = 1$,

$$h^{(1)} = \big[\, h^{(1)}_{\text{relu},cv}(\mathbf{x}_{cv}),\ h^{(1)}_{\text{conv}}(\mathbf{x}_{mv}),\ h^{(1)}_{\text{ref-relu}}(\mathbf{x}_{cc}),\ h^{(1)}_{\text{conc}}(\mathbf{x}_{mc}),\ h^{(1)}_{\text{mono}}(\mathbf{x}_{mn}),\ h^{(1)}_{\text{relu},u}(\mathbf{x}_u)\big]. \quad (6)$$

When $i = 2$,

$$h^{(2)}_{\text{conv}} = h_{\text{conv}}([h^{(1)}_{\text{relu},cv}(\mathbf{x}_{cv}), h^{(1)}_{\text{conv}}(\mathbf{x}_{mv}), h^{(1)}_{\text{mono}}(\mathbf{x}_{mn}), h^{(1)}_{\text{relu},u}(\mathbf{x}_u)]) \quad (7)$$

$$h^{(2)}_{\text{conc}} = h_{\text{conc}}([h^{(1)}_{\text{ref-relu}}(\mathbf{x}_{cc}), h^{(1)}_{\text{conc}}(\mathbf{x}_{mc}), h^{(1)}_{\text{mono}}(\mathbf{x}_{mn}), h^{(1)}_{\text{relu},u}(\mathbf{x}_u)]) \quad (8)$$

$$h^{(2)}_{\text{mono}} = h_{\text{mono}}([h^{(1)}_{\text{mono}}(\mathbf{x}_{mn}), h^{(1)}_{\text{relu},u}(\mathbf{x}_u)]) \quad (9)$$

$$h^{(2)}_{\text{relu}} = h_{\text{relu}}([h^{(1)}_{\text{relu},u}(\mathbf{x}_u)]) \quad (10)$$

$$h^{(2)} = \left[h^{(2)}_{\text{conv}}, h^{(2)}_{\text{conc}}, h^{(2)}_{\text{mono}}, h^{(2)}_{\text{relu}}\right] \quad (11)$$

When $i \geq 3$,

$$h^{(i)}_{\text{conv}} = h_{\text{conv}}([h^{(i-1)}_{\text{conv}}, h^{(i-1)}_{\text{mono}}, h^{(i-1)}_{\text{relu}}]) \quad (12)$$

$$h^{(i)}_{\text{conc}} = h_{\text{conc}}([h^{(i-1)}_{\text{conc}}, h^{(i-1)}_{\text{mono}}, h^{(i-1)}_{\text{relu}}]) \quad (13)$$

$$h^{(i)}_{\text{mono}} = h_{\text{mono}}([h^{(i-1)}_{\text{mono}}, h^{(i-1)}_{\text{relu}}]) \quad (14)$$

$$h^{(i)}_{\text{relu}} = h_{\text{relu}}([h^{(i-1)}_{\text{relu}}]) \quad (15)$$

$$h^{(i)} = \left[h^{(i)}_{\text{conv}}, h^{(i)}_{\text{conc}}, h^{(i)}_{\text{mono}}, h^{(i)}_{\text{relu}}\right] \quad (16)$$

Let $h^{(l)}$ be the output vector of lth hidden layer, and $f(\mathbf{x})$ be the output node in a fully connected output layer, then,

$$f(\mathbf{x}) = \exp(\mathbf{W})^\top h^{(l)} + \mathbf{b} \quad (17)$$

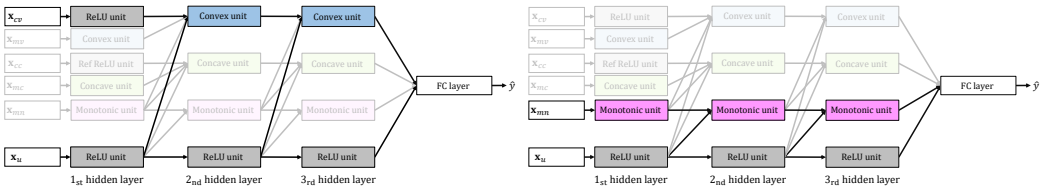
where $\exp(\mathbf{W})$ and $\mathbf{b}$ are the exponentiated weight matrix and bias vector between the l-th hidden layer and output layer. Although we present the formulation with a single output node, it naturally extends to multiple output nodes without issue. The above formulation enables COMONet to train a neural network that enforces the specified constraints on each variable. Monotonically decreasing variables in $\mathcal{MV}, \mathcal{MC}$ and $\mathcal{MN}$ are multiplied by -1 before training to convert them into increasing inputs, and restored at inference, enabling seamless integration with other shape constraints. Following the definitions in Section 3, we show that COMONet satisfies the convexity, concavity, and monotonicity properties. The proofs build on Lemmas 4.1-4.5 which characterize each unit.

Theorem 4.6 (Convexity of COMONet)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ denote the proposed COMONet with l hidden layers. Partition the input as $\mathbf{x} = (\mathbf{x}_v, \mathbf{x}_{\neg v})$, $\mathbf{x}_v \in \mathbb{R}^{|\mathcal{V}|}$. Then f is* partially jointly convex *with respect to $\mathbf{x}_v$.*

Theorem 4.7 (Concavity of COMONet)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ denote the proposed COMONet with l hidden layers. Partition the input as $\mathbf{x} = (\mathbf{x}_c, \mathbf{x}_{\neg c})$, $\mathbf{x}_c \in \mathbb{R}^{|\mathcal{C}|}$. Then f is* partially jointly concave *with respect to $\mathbf{x}_c$.*

Theorem 4.8 (Monotonicity of COMONet)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ denote the proposed COMONet with l hidden layers. Partition the input as $\mathbf{x} = (\mathbf{x}_m, \mathbf{x}_{\neg m})$, $\mathbf{x}_m \in \mathbb{R}^{|\mathcal{M}|}$. Then f is* partially monotonic increasing *with respect to $\mathbf{x}_m$.*

Detailed proofs for Theorem 4.6-4.8 are provided at Appendix A. Flow diagrams for each variable group appear in Appendix B. As illustrated in Appendix B, the unconstrained variables $\mathbf{x}_u$ are processed by multiple standard ReLU layers allowing them to fully exploit their expressive capacity as they propagate through the network. Moreover, at each hidden layer, activations of $\mathbf{x}_u$ are routed to the convex, concave, and monotonic units, enabling interactions with all other variable groups. Appendix G.2 demonstrates the overall effectiveness of the proposed network structure.



(a) (b)

Figure 3: **Configuration examples:** (a) Configuration of COMONet when the input consists only of $\mathbf{x}_{cv}$ and $\mathbf{x}_u$, (b) Configuration of COMONet when the input consists only of $\mathbf{x}_{mn}$ and $\mathbf{x}_u$.

A Unified and Configurable Architecture: The proposed model demonstrates high modularity and flexibility, enabling it to be easily tailored to accommodate various relationships and properties of input variables. This adaptability stems from its partially connected structure. For instance, when the only constraint is convexity, we partition the input as $\mathbf{x} = (\mathbf{x}_{cv}, \mathbf{x}_u) \in \mathbb{R}^{|\mathcal{CV}|} \times \mathbb{R}^{|\mathcal{U}|}$, and the resulting configuration, shown in Fig. 3(a), closely resembles the PICNN [3] architecture. Similarly, when the only constraint is monotonicity, we partition the input as $\mathbf{x} = (\mathbf{x}_{mn}, \mathbf{x}_u) \in \mathbb{R}^{|\mathcal{MN}|} \times \mathbb{R}^{|\mathcal{U}|}$, and the resulting configuration, shown in Fig. 3(b), aligns with the SMNN [29] architecture.

Interaction layer: We introduce an optional interaction layer to enhance the expressive power of COMONet while strictly preserving variable-wise convexity and concavity constraints. Formally, let $f(\mathbf{x})$ denote the output of the base COMONet. With the optional interaction layer, the final prediction $\tilde{f}(\mathbf{x})$ is given by

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + I_{\text{cross}}(\mathbf{x}) + I_{\text{intra}}(\mathbf{x}). \tag{18}$$

$$\mathcal{I}_{\text{cross}}(\mathbf{x}) = \sum_{i \in \mathcal{CC}} \sum_{j \in \mathcal{CV}} \alpha_{i,j}\, x_i x_j. \tag{19}$$

$$\mathcal{I}_{\text{intra}}(\mathbf{x}) = \sum_{i \in \mathcal{CV},\, j \in \mathcal{CV},\, i \neq j} \beta_{i,j}\, x_i x_j + \sum_{i \in \mathcal{CC},\, j \in \mathcal{CC},\, i \neq j} \beta_{i,j}\, x_i x_j. \tag{20}$$

First, cross-group interactions in (19) enable interactions between $\mathcal{CV}$ and $\mathcal{CC}$, which are otherwise processed independently. Such interactions can enhance the representational capacity of the model. Second, intra-group interactions in (20) enable interactions within $\mathcal{CV}$ and $\mathcal{CC}$. Although COMONet inherently enforces strong global curvature constraints, such as joint convexity or joint concavity, some tasks may require more flexible variable-wise curvature properties. Intra-group interactions provide a controlled way to relax joint curvature while preserving the assigned variable-wise curvature. To preserve the assigned shape constraints, the interaction function $\mathcal{I}$ should not alter the signs of the second-order partial derivatives with respect to each involved variable, thereby maintaining the assigned convexity or concavity. Theorems 4.9 and 4.10 formalize the curvature-preserving properties of the proposed bilinear interactions. Theorem 4.9 shows that cross-group bilinear interactions between convex and concave variables preserve their variable-wise curvature. Theorem 4.10 further shows that intra-group bilinear interactions preserve the assigned variable-wise curvature while allowing joint curvature to be relaxed.

Theorem 4.9 (Curvature preservation of cross-group bilinear interactions)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ satisfy the prescribed partial convexity and partial concavity constraints. For any $i \in \mathcal{CV}$ and $j \in \mathcal{CC}$, define*

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + \alpha_{ij} x_i x_j, \qquad \alpha_{ij} \in \mathbb{R}.$$

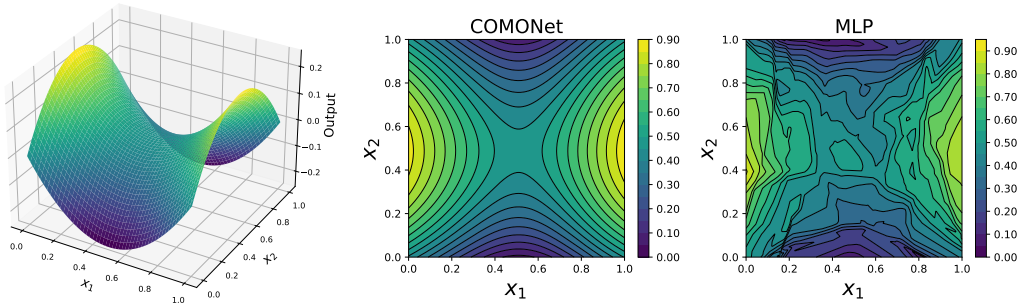
Then, adding the bilinear term $\alpha_{ij} x_i x_j$ does not change the partial convexity of f with respect to x_i or the partial concavity of f with respect to x_j.

Theorem 4.10 (Curvature preservation of intra-group bilinear interactions)**.** *Let $f : \mathbb{R}^d \to \mathbb{R}$ satisfy the prescribed partial convexity or partial concavity constraints. For any two distinct variables $i, j \in \mathcal{CV}$ or $i, j \in \mathcal{CC}$, define*

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + \beta_{ij} x_i x_j, \qquad \beta_{ij} \in \mathbb{R}.$$

Then, adding the bilinear term $\beta_{ij} x_i x_j$ preserves the assigned variable-wise convexity or concavity. However, it does not necessarily preserve joint convexity or joint concavity over the pair (x_i, x_j).

The proofs of Theorems 4.9 and 4.10 are provided in Appendix A, and further details of the interaction layer appear in Appendix C.



(a) (b)

Figure 4: (a) Hyperbolic Paraboloid function (b) Contour plots of the fitted models by COMONet (Left) and MLP (Right) with respect to x_1 and x_2 when $\lambda = 0.05$

5 Numerical Experiments

5.1 Experiments on synthetic datasets

We first evaluate COMONet on synthetic datasets to demonstrate its ability to satisfy diverse shape constraints. Since our method is theoretically proven to guarantee these constraints and has been shown to be adaptable to different types of inputs, testing all possible combinations of shape constraints introduced in Section 1 would be unnecessary. Instead, we consider two representative mixed-constraint datasets that are challenging for existing methods.

Convexity and Concavity: The well-known hyperbolic paraboloid shown in (21) and Fig. 4(a) was chosen because it exhibits both convexity and concavity. In (21), x_1 is a convex input, while x_2 is a concave input of y. Existing approaches remain limited in this setting, as they either cannot accommodate mixed convex-concave constraints within a unified architecture or do not provide strict guarantees of constraint satisfaction. To evaluate whether COMONet can fit this function even under noisy observations, we introduced Gaussian noise ϵ and varied the noise level parameter λ.

$$y = (x_1 - 0.5)^2 - (x_2 - 0.5)^2 \quad + \lambda\epsilon, \quad \epsilon \sim N(0,1), x_i \in [0,1], i = 1,2, \quad \lambda \in \{0, 0.05, 0.1, 0.2\}. \tag{21}$$

Since the inputs are exclusively convex and concave, we utilized the COMONet structure shown in Appendix E.5 for this experiment. Specifically, the ReLU layer and the convex layer were applied to x_1, while the reflected ReLU layer and the concave layer were used for x_2. For comparison, a traditional MLP was employed as the baseline method. At each noise level, we generated 1,000 instances and split them into training (80%) and test (20%) sets. The test mean squared errors (MSEs) are presented in Table 1. As shown in Table 1, MLP outperformed COMONet in the absence of noise. However, as the noise level increased, COMONet demonstrated better performance, with a smaller increase in MSE compared to MLP. This demonstrates that COMONet provides a robust fit to the function. More importantly, consider Fig. 4(b), which displays the contour plots of the fitted models from both methods. Even with a small amount of noise, MLP failed to preserve convexity and concavity, whereas our method consistently maintained these constraints. Although not shown, our method maintains these constraints at higher noise levels.

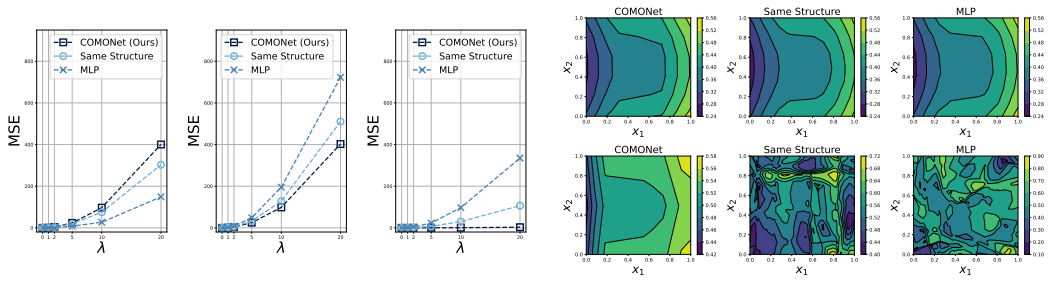
Table 1: Test MSEs of COMONet and MLP

λ	**0**	**0.05**	**0.1**	**0.2**
COMONet	0.0010	0.0028	0.0117	0.0448
MLP	0.0001	0.0030	0.0124	0.0529

Monotonicity and Convexity: We extended our experiment to a case with more shape constraints. Specifically, we designed a 4-dimensional example, with the function defined in (22).

$$y = \frac{2\pi x_1 + \sin(2\pi x_1)}{2\pi} + (x_2 - 0.5)^2 + e^{x_3} + \cos(2\pi x_4) + \lambda\epsilon, \quad \epsilon \sim N(0,1). \tag{22}$$
$$x_i \in [0,1], i = 1,2,3,4, \quad \lambda \in \{0, 1, 2, 5, 10, 20\}.$$

As shown in (22), a distinct shape constraint was assigned to each input variable: x_1 is a monotonically increasing variable, x_2 is convex, x_3 is monotonic-convex, and x_4 is unconstrained in the noiseless setting. Similar to the previous experiment, we introduced Gaussian noise and controlled the noise level by adjusting the λ value. At each noise level, we generated 3,000 instances and split them into training (80%) and test (20%) sets. For comparison, two baseline methods were used in this



(a) (b)

Figure 5: (a) MSEs from the experiments at different λ values. (left) Training MSEs, (middle) Test MSEs, (right) Denoised Test MSEs. (b) Contour plots in the x_1-x_2 plane for $\lambda = 0$ (top) and $\lambda = 1$ (bottom), comparing outputs of COMONet, same structure, and MLP.

experiment. In addition to the traditional MLP, we included a model referred to as "Same Structure" (shown in Appendix F.1) which shares the same architecture as COMONet but replaces all units with ReLU layers, meaning no shape constraints were enforced. To ensure a fair comparison, all models were constructed with an identical number of nodes per unit.

The performance results in terms of MSE are presented in Fig. 5(a). The results indicate that MLP achieved the lowest training MSE across all λ values, followed by the Same Structure model, while COMONet exhibited the highest training error. In contrast, for test MSE, COMONet consistently outperformed the other models, while MLP exhibited the highest error. This indicates that MLP overfits the noise during training, whereas our method does not. Notably, in the denoised test MSE, which evaluates performance in predicting the noise-free ground truth, COMONet maintained robust predictions even as λ increased. These results quantitatively confirm that the shape constraints enforced by COMONet not only mitigate the impact of noise but also significantly improve the model's generalization performance. Fig. 5(b) shows the contour plots of the fitted models for all three approaches. In the absence of noise, all models produced reasonable fits. However, even with a small noise level ($\lambda = 1$), the MLP and Same Structure models completely failed to satisfy the constraints, whereas COMONet successfully preserved monotonicity with respect to x_1 and convexity with respect to x_2. This figure confirms that our method effectively maintains the imposed constraints.

In terms of trustworthiness, the LIME [41] analysis in Fig. 13 in Appendix F.4 demonstrates that the proposed method prevents incorrect interpretations. It shows the LIME values for x_1 and x_3 at the different noise levels. Notice that x_1 is a monotonically increasing variable and x_3 is monotonic-convex, meaning their LIME values should always be positive for a correct interpretation. As shown in the figure, the LIME values for both x_1 and x_3 computed from the COMONet models are consistently positive, with low variance, which aligns with the expected interpretation. By contrast, LIME values from the MLP models fluctuate in sign and exhibit high variance. These results validate that embedding shape constraints yields more reliable interpretations.

5.2 Experiments on real-world datasets

We now present the general performance level of our proposed method on real-world datasets through a comparative study with benchmark techniques. The study consists of two parts, comparison with monotonic neural networks and comparison with methods incorporating both monotonicity and convexity. For the first part, five datasets were used. The Auto-MPG [49] and Blog Feedback [47] datasets were used for regression tasks, while the Heart Disease [50], COMPAS [4], and Loan Defaulter [51] datasets were used for classification tasks. The benchmark methods in this comparison include the most recent monotonic neural networks approaches reviewed in Section 2. For the second part, the Car Sales [26], Puzzle Sales [12], and Wine Ratings [13] datasets were used, all of which were designed for regression tasks. The benchmark methods selected for comparison were SCNN and PenDer, as they incorporate both convexity and monotonic-convexity constraints. For regression tasks, we reported metrics including mean squared error (MSEs) and root mean squared error (RMSE), while for classification tasks, we reported accuracy. Further details on the experiments and additional information about the datasets can be found in Appendix E. All experiments in this section were conducted over multiple iterations, with the mean and standard deviation reported. The best performance for each dataset is highlighted in bold, and dagger symbol (†) indicates statistical

Table 2: Results on real-world datasets for comparison with monotonic neural networks

Method	Auto MPG	Heart Disease	COMPAS	Blog Feedback	Loan Defaulter
	Test MSE ↓	Test Acc ↑	Test Acc ↑	Test RMSE ↓	Test Acc ↑
DLN [55]	13.34 ± 2.42	0.86 ± 0.02	67.9 ± 0.3	0.161 ± 0.001	65.1 ± 0.2
Min-Max Net [11]	10.14 ± 1.54	0.75 ± 0.04	67.8 ± 0.1	0.163 ± 0.001	64.9 ± 0.1
Non-Neg-DNN	– – –	– – –	67.3 ± 0.9	0.168 ± 0.001	65.1 ± 0.1
COMET [46]	8.81 ± 1.81	0.86 ± 0.03	– – –	– – –	– – –
Certified MNN [32]	– – –	– – –	68.8 ± 0.9†	0.158 ± 0.001	65.2 ± 0.1
LMN [38]	7.58 ± 1.20†	$\mathbf{0.90 \pm 0.02}$	69.3 ± 0.1†	0.160 ± 0.001	$\mathbf{65.4 \pm 0.0}$
Constrained MNN [43]	8.37 ± 0.08	0.89 ± 0.00†	69.2 ± 0.2†	0.156 ± 0.001	65.3 ± 0.1†
SMNN [29]	7.44 ± 1.20†	0.88 ± 0.04†	69.3 ± 0.9†	$\mathbf{0.150 \pm 0.001}$	65.0 ± 0.1
COMONet (Ours)	$\mathbf{7.38 \pm 1.32}$	0.87 ± 0.04†	$\mathbf{69.5 \pm 1.0}$	0.153 ± 0.001	64.9 ± 0.1

Table 3: Results on real-world datasets for comparison with SCNN and PenDer

Method	**Car Sales** (Test MSE ↓)		**Puzzle Sales** (Test MSE ↓)		**Wine Ratings** (Test MSE ↓)	
	(conv)	(conv, decr)	(conc)	(conc, incr)	(conc)	(conc, incr)
SCNN [23]	11093 ± 487	10880 ± 291	9460 ± 256†	$\mathbf{9258 \pm 319}$	6.32 ± 0.19	6.43 ± 0.18
PenDer [22]	10411 ± 107†	10415 ± 104†	9428 ± 113†	9519 ± 92†	$\mathbf{5.19 \pm 0.11}$	5.27 ± 0.20†
COMONet (Ours)	$\mathbf{10391 \pm 140}$	$\mathbf{10410 \pm 128}$	$\mathbf{9409 \pm 41}$	9263 ± 86†	5.53 ± 0.46†	$\mathbf{5.26 \pm 0.06}$

tie with the best-performing method. We consider two methods to be in a statistical tie when their mean test MSE $\pm$ one standard deviation intervals overlap.

Comparison with monotonic neural networks: The results shown in Table 2 represent the means and standard deviations obtained from cross-validation. As shown in Table 2, our method generally performed well, achieving the best performance on some datasets and remaining comparable to other methods on the rest. Specifically, it achieved the best results for the Auto-MPG and COMPAS datasets. For the Heart Disease dataset, it was statistically tied with the best-performing method. On the Blog Feedback dataset, it ranked second. Although its ranking for the Loan Defaulter dataset was lower, its accuracy remained within a reasonable range compared to other methods.

Comparison with SCNN and PenDer: Our experimental evaluation compares COMONet against SCNN and PenDer on three real-world datasets, Car Sales, Puzzle Sales, and Wine Ratings, under two constraint settings per dataset: convex (concave) only, and convex (concave) monotonic. For each dataset, we used the provided train/test split and averaged the test MSE over five independent runs using the optimal hyperparameter settings found. Table 3 shows performance of the proposed method and the comparison methods on these three datasets. COMONet achieves the best Test MSE in four of the six settings Car Sales (conv), (conv, decr), Puzzle Sales (conc) and Wine Ratings (conc, incr), and when accounting for statistical ties matches or outperforms all baselines across all six settings. While PenDer matches or outperforms across all datasets and settings, its shape-conformance metrics $\mathcal{M}_k$ and $\mathcal{C}_k$ sometimes fall below 1, indicating that it fails to fully satisfy the prescribed constraints. Here, $\mathcal{M}_k$ and $\mathcal{C}_k$ denote the proportions of samples satisfying monotonicity and convexity constraints respectively [22]. For example, the convexity score on the Puzzle Sales dataset and both the monotonicity and convexity scores on the Wine Ratings dataset are 0.98 or 0.99 values close to one but nevertheless indicative of incomplete constraint satisfaction. Table 8 in Appendix F.2 shows the detailed numerical results for PenDer's performance and its constraint conformance. These results demonstrate that COMONet not only delivers comparable or superior predictive performance but also guarantees full adherence to the enforced shape constraints.

6 Conclusion

In this work, we introduced COMONet, a gradient-descent–trained neural architecture that embeds domain knowledge as inductive biases enforcing convexity, concavity, monotonicity, and their combinations while permitting selective application of these shape constraints per variable. Empirical results on synthetic and real-world datasets show that COMONet achieves competitive predictive performance, often matching or outperforming existing baselines, while strictly satisfying the prescribed constraints by construction. However, COMONet requires a priori knowledge of each variable's shape constraints, meaning incomplete or erroneous domain information may impair its effectiveness. In addition, while COMONet focuses on global variable-wise shape constraints, extending the framework toward conditional or interaction-dependent shape behaviors represents a technically challenging yet promising direction. Looking ahead, we will seek theoretical guarantees that COMONet serves as a universal approximator for arbitrary functions under prescribed shape constraints and will explore its use as a modular component in time-series and image-based tasks.

Acknowledgments and Disclosure of Funding

This research was supported in part by the National Research Foundation of Korea grant funded by the Korean government (MSIT) (Grant numbers RS-2022-NR068758 and RS-2024-00361377), in part by the Korea Evaluation Institute of Industrial Technology grant funded by the Korean government (MOTIR) (Grant number RS-2025-25458052), and in part by the National IT Industry Promotion Agency grant funded by the Korean government (MSIT) (Grant number RS-2026-25621650).

References

[1] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34:4699–4711, 2021.

[2] Michael Aikenhead. The uses and abuses of neural networks in law. *Santa Clara Computer & High Tech. LJ*, 12:31, 1996.

[3] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *International conference on machine learning*, pages 146–155. PMLR, 2017.

[4] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. Machine bias: There's software used across the country to predict future criminals. And it's biased against blacks. ProPublica, 23 May, 2016.

[5] Brian Armstrong-Hélouvry, Pierre Dupont, and Carlos Canudas De Wit. A survey of models, analysis tools and compensation methods for the control of machines with friction. *Automatica*, 30(7):1083–1138, 1994.

[6] Clark Barrett and Cesare Tinelli. *Satisfiability modulo theories*. Springer, 2018.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] Matthias Breuer and David Windisch. Investment Dynamics and Earnings-Return Properties: A Structural Approach. *Journal of Accounting Research*, 57(3):639–674, 2019.

[9] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Optimal Control Via Neural Networks: A Convex Approach. In *International Conference on Learning Representations*, 2018.

[10] Andrew Cotter, Maya Gupta, and Jan Pfeifer. A light touch for heavily constrained SGD. In *Conference on Learning Theory*, pages 729–771. PMLR, 2016.

[11] Hennie Daniels and Marina Velikova. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

[12] dbahri / Kaggle. Puzzles Data. `https://www.kaggle.com/datasets/dbahri/puzzles`, 2024. accessed: 2025-05-05.

[13] dbahri / Kaggle. Wine Ratings Data. `https://www.kaggle.com/datasets/dbahri/wine-ratings`, 2024. accessed: 2025-05-05.

[14] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations*, 2017.

[15] Charles Dugas, Yoshua Bengio, François Bélisle, Claude Nadeau, and René Garcia. Incorporating Functional Knowledge in Neural Networks. *Journal of Machine Learning Research*, 10(6), 2009.

[16] Liran Einav, Mark Jenkins, and Jonathan Levin. The impact of credit scoring on consumer lending. *The RAND Journal of Economics*, 44(2):249–274, 2013.

[17] Eugene F Fama and Kenneth R French. A five-factor asset pricing model. *Journal of financial economics*, 116(1):1–22, 2015.

[18] Ad J Feelders. Prior knowledge in economic applications of data mining. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 395–400. Springer, 2000.

[19] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[20] Herbert Goldstein, Charles P Poole, and John Safko. *Classical mechanics*, volume 2. Addison-wesley Reading, MA, 1950.

[21] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273, 2020.

[22] Akhil Gupta, Lavanya Marla, Ruoyu Sun, Naman Shukla, and Arinbjörn Kolbeinsson. Pender: Incorporating shape constraints via penalized derivatives. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11536–11544, 2021.

[23] Maya Gupta, Dara Bahri, Andrew Cotter, and Kevin Canini. Diminishing returns shape constraints for interpretability and regularization. *Advances in neural information processing systems*, 31, 2018.

[24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL `https://www.gurobi.com`.

[25] Pieter-Jan Hoedt and Günter Klambauer. Principled weight initialisation for input-convex neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[26] hsinha53 / Kaggle. Car Sales Data. `https://www.kaggle.com/datasets/hsinha53/car-sales/data`, 2023. accessed: 2025-05-05.

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[28] Frank P Kelly, Aman K Maulloo, and David Kim Hong Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research society*, 49(3):237–252, 1998.

[29] Hyunho Kim and Jong-Seok Lee. Scalable Monotonic Neural Networks. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=DjIsNDEOYX`.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Shan Sung Liew, Mohamed Khalil-Hani, and Rabia Bakhteri. Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. *Neurocomputing*, 216:718–734, 2016.

[32] Xingchao Liu, Xing Han, Na Zhang, and Qiang Liu. Certified monotonic neural networks. *Advances in Neural Information Processing Systems*, 33:15427–15438, 2020.

[33] Harry M Markowitz. *Portfolio selection: efficient diversification of investments*. Yale university press, 2008.

[34] Nikhil Muralidhar, Mohammad Raihanul Islam, Manish Marwah, Anuj Karpatne, and Naren Ramakrishnan. Incorporating prior domain knowledge into deep neural networks. In *2018 IEEE international conference on big data (big data)*, pages 36–45. IEEE, 2018.

[35] Ryan J Murdock, Steven K Kauwe, Anthony Yu-Tung Wang, and Taylor D Sparks. Is domain knowledge necessary for machine learning materials properties? *Integrating Materials and Manufacturing Innovation*, 9:221–227, 2020.

[36] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[37] David MQ Nelson, Adriano CM Pereira, and Renato A De Oliveira. Stock market's price movement prediction with LSTM neural networks. In *2017 International joint conference on neural networks (IJCNN)*, pages 1419–1426. Ieee, 2017.

[38] Niklas Nolte, Ouail Kitouni, and Mike Williams. Expressive Monotonic Neural Networks. In *The Eleventh International Conference on Learning Representations*, 2022.

[39] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[40] Samuel Pfrommer, Brendon Anderson, Julien Piet, and Somayeh Sojoudi. Asymmetric certified robustness via feature-convex neural networks. *Advances in Neural Information Processing Systems*, 36, 2024.

[41] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[42] R Tyrrell Rockafellar. *Convex analysis*, volume 28. Princeton university press, 1997.

[43] Davor Runje and Sharath M Shankaranarayana. Constrained monotonic neural networks. In *International Conference on Machine Learning*, pages 29338–29353. PMLR, 2023.

[44] Parameswaran Sankaranarayanan and Raghunathan Rengaswamy. CDiNN–Convex difference neural networks. *Neurocomputing*, 495:153–168, 2022.

[45] Nida Shahid, Tim Rappon, and Whitney Berta. Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PloS one*, 14(2):e0212356, 2019.

[46] Aishwarya Sivaraman, Golnoosh Farnadi, Todd Millstein, and Guy Van den Broeck. Counterexample-guided learning of monotonic neural networks. *Advances in Neural Information Processing Systems*, 33:11936–11948, 2020.

[47] Myra Spiliopoulou, Lars Schmidt-Thieme, and Ruth Janning. *Data analysis, machine learning and knowledge discovery*. Springer, 2014.

[48] Mark W Spong, Seth Hutchinson, Mathukumalli Vidyasagar, et al. *Robot modeling and control*, volume 3. Wiley New York, 2006.

[49] UCI Machine Learning Repository. Auto MPG Data Set. `https://archive.ics.uci.edu/ml/datasets/auto+mpg`, 1983–2021. accessed: 2025-05-05.

[50] UCI Machine Learning Repository. Heart Disease Data Set. `https://archive.ics.uci.edu/ml/datasets/Heart+Disease`, 1988–2021. accessed: 2025-05-05.

[51] Wendy Kan / Kaggle. Lending Club Loan Data. `https://www.kaggle.com/wendykan/lending-club-loan-data`, 2024. accessed: 2025-05-05.

[52] Hiroki Yanagisawa, Kohei Miyaguchi, and Takayuki Katsuki. Hierarchical Lattice Layer for Partially Monotone Neural Networks. In *Advances in Neural Information Processing Systems*, 2022.

[53] Shu Yang and B Wayne Bequette. Optimization-based control using input convex neural networks. *Computers & Chemical Engineering*, 144:107143, 2021.

[54] I-Cheng Yeh. Generalization of strength versus water–cementitious ratio relationship to age. *Cement and Concrete Research*, 36(10):1865–1873, 2006.

[55] Seungil You, David Ding, Kevin Canini, Jan Pfeifer, and Maya Gupta. Deep lattice networks and partial monotonic functions. *Advances in neural information processing systems*, 30, 2017.

[56] Ting Yu, Simeon Simoff, and Tony Jan. VQSVM: a case study for incorporating prior domain knowledge into inductive machine learning. *Neurocomputing*, 73(13-15):2614–2623, 2010.

[57] Hong Zhang and Zhen Zhang. Feedforward networks with monotone constraints. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 3, pages 1820–1823. IEEE, 1999.

[58] Xuan Zhang, Wenbo Shi, Bin Yan, Ali Malkawi, and Na Li. Decentralized and distributed temperature control via HVAC systems in energy efficient buildings. *arXiv preprint arXiv:1702.03308*, 2017.

[59] Sen Zhao, Erez Louidor, and Maya Gupta. Global optimization networks. In *International Conference on Machine Learning*, pages 26927–26957. PMLR, 2022.

A Proofs

Composition principles. We first state the standard composition principles used throughout the proofs [7, 42].

Proposition A.1 (Composition principles)**.** *Let $g : \mathbb{R}^m \to \mathbb{R}$ and $u_r : \mathbb{R}^d \to \mathbb{R}$, $r \in [m]$.*

If g is convex and coordinatewise increasing, and each u_r is convex, then

$$f(\mathbf{x}) = g(u_1(\mathbf{x}), \ldots, u_m(\mathbf{x}))$$

is convex.

If g is concave and coordinatewise increasing, and each u_r is concave, then

$$f(\mathbf{x}) = g(u_1(\mathbf{x}), \ldots, u_m(\mathbf{x}))$$

is concave.

If g and each u_r are coordinatewise increasing, then

$$f(\mathbf{x}) = g(u_1(\mathbf{x}), \ldots, u_m(\mathbf{x}))$$

is coordinatewise increasing.

Proof. We prove each statement directly from Definitions 3.1–3.3.

First, suppose that g is convex and coordinatewise increasing, and each u_r is convex. For any $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ and $\lambda \in [0, 1]$, convexity of each u_r gives

$$u_r(\lambda\mathbf{x} + (1-\lambda)\mathbf{x}') \le \lambda u_r(\mathbf{x}) + (1-\lambda)u_r(\mathbf{x}'), \qquad r \in [m].$$

Since g is coordinatewise increasing,

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{x}') \le g\big(\lambda u_1(\mathbf{x}) + (1-\lambda)u_1(\mathbf{x}'), \ldots, \lambda u_m(\mathbf{x}) + (1-\lambda)u_m(\mathbf{x}')\big).$$

By convexity of g,

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{x}') \le \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{x}').$$

Therefore, by the standard definition of convexity (equivalently, Definition 3.1 with all coordinates considered jointly), f is convex.

Second, suppose that g is concave and coordinatewise increasing, and each u_r is concave. Then

$$u_r(\lambda\mathbf{x} + (1-\lambda)\mathbf{x}') \ge \lambda u_r(\mathbf{x}) + (1-\lambda)u_r(\mathbf{x}'), \qquad r \in [m].$$

Since g is coordinatewise increasing,

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{x}') \ge g\big(\lambda u_1(\mathbf{x}) + (1-\lambda)u_1(\mathbf{x}'), \ldots, \lambda u_m(\mathbf{x}) + (1-\lambda)u_m(\mathbf{x}')\big).$$

By concavity of g,

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{x}') \ge \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{x}').$$

Therefore, by the standard definition of concavity (equivalently, Definition 3.2 with all coordinates considered jointly), f is concave.

Finally, suppose that g and each u_r are coordinatewise increasing. If $\mathbf{x} \preceq \mathbf{x}'$ ($\preceq$ denotes componentwise inequality), then

$$u_r(\mathbf{x}) \le u_r(\mathbf{x}'), \qquad r \in [m].$$

Since g is coordinatewise increasing,

$$f(\mathbf{x}) = g(u_1(\mathbf{x}), \ldots, u_m(\mathbf{x})) \le g(u_1(\mathbf{x}'), \ldots, u_m(\mathbf{x}')) = f(\mathbf{x}').$$

Therefore, by the order-based definition of monotonicity (equivalently, Definition 3.3 with all coordinates considered), f is coordinatewise increasing. □

Lemma 4.1. Let $h_{\text{conv}} : \mathbb{R}^d \to \mathbb{R}^k$ and denote its jth coordinate by $f_j(\mathbf{t}) \coloneqq \big[h_{\text{conv}}(\mathbf{t})\big]_j = \big(\exp(\mathbf{w}_j)^\top \mathbf{t} + b_j\big)_+$. Then, $\forall j \in [k]$, f_j is jointly convex in $\mathbf{t}$ and coordinatewise increasing.

Proof. Define

$$g_j(\mathbf{t}) = \exp(\mathbf{w}_j)^\top \mathbf{t} + b_j.$$

Since g_j is affine, for any $\mathbf{t}, \mathbf{t}' \in \mathbb{R}^d$ and $\lambda \in [0, 1]$,

$$g_j(\lambda\mathbf{t} + (1-\lambda)\mathbf{t}') = \lambda g_j(\mathbf{t}) + (1-\lambda) g_j(\mathbf{t}').$$

Further, ReLU

$$\sigma(z) = (z)_+$$

is convex. Therefore,

$$\begin{aligned} f_j(\lambda\mathbf{t} + (1-\lambda)\mathbf{t}') &= \sigma\big(g_j(\lambda\mathbf{t} + (1-\lambda)\mathbf{t}')\big) \\ &= \sigma\big(\lambda g_j(\mathbf{t}) + (1-\lambda) g_j(\mathbf{t}')\big) \\ &\le \lambda\sigma(g_j(\mathbf{t})) + (1-\lambda)\sigma(g_j(\mathbf{t}')) \\ &= \lambda f_j(\mathbf{t}) + (1-\lambda) f_j(\mathbf{t}'). \end{aligned}$$

Hence, by Definition 3.1, f_j is jointly convex.

Now let $\mathbf{t} \preceq \mathbf{t}'$. Since $\exp(\mathbf{w}_j)$ is elementwise positive,

$$g_j(\mathbf{t}) \le g_j(\mathbf{t}').$$

Further, ReLU is increasing, so

$$f_j(\mathbf{t}) = \sigma(g_j(\mathbf{t})) \le \sigma(g_j(\mathbf{t}')) = f_j(\mathbf{t}').$$

Therefore, by Definition 3.3, f_j is coordinatewise increasing. □

Lemma 4.2. Let $h_{\text{relu}} : \mathbb{R}^d \to \mathbb{R}^k$ and denote its jth coordinate by $f_j(\mathbf{t}) \coloneqq \big[h_{\text{relu}}(\mathbf{t})\big]_j = (\mathbf{w}_j^\top \mathbf{t} + b_j)_+$. Then, $\forall j \in [k]$, f_j is jointly convex in $\mathbf{t}$.

Proof. Define

$$g_j(\mathbf{t}) = \mathbf{w}_j^\top \mathbf{t} + b_j.$$

Since g_j is affine,

$$g_j(\lambda\mathbf{t} + (1-\lambda)\mathbf{t}') = \lambda g_j(\mathbf{t}) + (1-\lambda) g_j(\mathbf{t}').$$

Further, ReLU is convex. Therefore,

$$\begin{aligned} f_j(\lambda\mathbf{t} + (1-\lambda)\mathbf{t}') &= (g_j(\lambda\mathbf{t} + (1-\lambda)\mathbf{t}'))_+ \\ &= (\lambda g_j(\mathbf{t}) + (1-\lambda) g_j(\mathbf{t}'))_+ \\ &\le \lambda(g_j(\mathbf{t}))_+ + (1-\lambda)(g_j(\mathbf{t}'))_+ \\ &= \lambda f_j(\mathbf{t}) + (1-\lambda) f_j(\mathbf{t}'). \end{aligned}$$

Hence, by Definition 3.1, f_j is jointly convex. □

Theorem 4.6. Let $f : \mathbb{R}^d \to \mathbb{R}$ denote the proposed COMONet with l hidden layers. Partition the input as $\mathbf{x} = (\mathbf{x}_v, \mathbf{x}_{\neg v})$, $\mathbf{x}_v \in \mathbb{R}^{|\mathcal{V}|}$. Then f is partially jointly convex with respect to $\mathbf{x}_v$.

Proof. By construction, variables in $\mathbf{x}_v$ are connected only through h_{conv} and h_{relu} units. By Lemma 4.1 and Lemma 4.2, the corresponding hidden representations are jointly convex with respect to $\mathbf{x}_v$.

Assume that hidden representations in layer k are jointly convex with respect to $\mathbf{x}_v$. Since layer $k+1$ is formed by compositions of jointly convex functions with coordinatewise increasing convex units, Proposition A.1 implies that layer $k+1$ is also jointly convex with respect to $\mathbf{x}_v$.

Therefore, by induction, all hidden representations and the final output $f(\mathbf{x})$ are jointly convex with respect to $\mathbf{x}_v$. Finally, fixing $\mathbf{x}_{\neg v}$, Definition 3.1 implies that f is partially jointly convex with respect to $\mathbf{x}_v$. □

Lemma 4.3. Let $h_{\text{conc}} : \mathbb{R}^d \to \mathbb{R}^k$ and denote its jth coordinate by $f_j(\mathbf{t}) \coloneqq \big[h_{\text{conc}}(\mathbf{t})\big]_j = -\big(-(\exp(\mathbf{w}_j)^\top \mathbf{t} + b_j)\big)_+$. Then, $\forall j \in [k]$, f_j is jointly concave in $\mathbf{t}$ and coordinatewise increasing.

Proof. Define

$$g_j(\mathbf{t}) = \exp(\mathbf{w}_j)^\top \mathbf{t} + b_j, \qquad \rho(z) = -(-z)_+.$$

Since g_j is affine, for any $\mathbf{t}, \mathbf{t}' \in \mathbb{R}^d$ and $\lambda \in [0,1]$,

$$g_j(\lambda \mathbf{t} + (1-\lambda)\mathbf{t}') = \lambda g_j(\mathbf{t}) + (1-\lambda) g_j(\mathbf{t}').$$

Further, ρ is concave. Therefore,

$$\begin{aligned} f_j(\lambda \mathbf{t} + (1-\lambda)\mathbf{t}') &= \rho\big(g_j(\lambda \mathbf{t} + (1-\lambda)\mathbf{t}')\big) \\ &= \rho\big(\lambda g_j(\mathbf{t}) + (1-\lambda) g_j(\mathbf{t}')\big) \\ &\geq \lambda \rho(g_j(\mathbf{t})) + (1-\lambda)\rho(g_j(\mathbf{t}')) \\ &= \lambda f_j(\mathbf{t}) + (1-\lambda) f_j(\mathbf{t}'). \end{aligned}$$

Hence, by Definition 3.2, f_j is jointly concave.

Now let $\mathbf{t} \preceq \mathbf{t}'$. Since $\exp(\mathbf{w}_j)$ is elementwise positive,

$$g_j(\mathbf{t}) \leq g_j(\mathbf{t}').$$

Further, ρ is increasing, so

$$f_j(\mathbf{t}) = \rho(g_j(\mathbf{t})) \leq \rho(g_j(\mathbf{t}')) = f_j(\mathbf{t}').$$

Therefore, by Definition 3.3, f_j is coordinatewise increasing. □

Lemma 4.4. Let $h_{\text{ref-relu}} : \mathbb{R}^d \to \mathbb{R}^k$ and denote its jth coordinate by $f_j(\mathbf{t}) \coloneqq \big[h_{\text{ref-relu}}(\mathbf{t})\big]_j = -\big(-(\mathbf{w}_j^\top \mathbf{t} + b_j)\big)_+$. Then, $\forall j \in [k]$, f_j is jointly concave in $\mathbf{t}$.

Proof. Define

$$g_j(\mathbf{t}) = \mathbf{w}_j^\top \mathbf{t} + b_j, \qquad \rho(z) = -(-z)_+.$$

Since g_j is affine, for any $\mathbf{t}, \mathbf{t}' \in \mathbb{R}^d$ and $\lambda \in [0,1]$,

$$g_j(\lambda \mathbf{t} + (1-\lambda)\mathbf{t}') = \lambda g_j(\mathbf{t}) + (1-\lambda) g_j(\mathbf{t}').$$

Further, ρ is concave. Therefore,

$$\begin{aligned} f_j(\lambda \mathbf{t} + (1-\lambda)\mathbf{t}') &= \rho\big(g_j(\lambda \mathbf{t} + (1-\lambda)\mathbf{t}')\big) \\ &= \rho\big(\lambda g_j(\mathbf{t}) + (1-\lambda) g_j(\mathbf{t}')\big) \\ &\geq \lambda \rho(g_j(\mathbf{t})) + (1-\lambda)\rho(g_j(\mathbf{t}')) \\ &= \lambda f_j(\mathbf{t}) + (1-\lambda) f_j(\mathbf{t}'). \end{aligned}$$

Hence, by Definition 3.2, f_j is jointly concave. □

Theorem 4.7 Let $f : \mathbb{R}^d \to \mathbb{R}$ denote the proposed COMONet with l hidden layers. Partition the input as $\mathbf{x} = (\mathbf{x}_c, \mathbf{x}_{\neg c}), \qquad \mathbf{x}_c \in \mathbb{R}^{|\mathcal{C}|}$. Then f is partially jointly concave with respect to $\mathbf{x}_c$.

Proof. By construction, variables in $\mathbf{x}_c$ are connected only through h_{conc} and $h_{\text{ref-relu}}$ units. By Lemma 4.3 and Lemma 4.4, the corresponding hidden representations are jointly concave with respect to $\mathbf{x}_c$.

Assume that hidden representations in layer k are jointly concave with respect to $\mathbf{x}_c$. Since layer $k+1$ is formed by compositions of jointly concave functions with coordinatewise increasing concave units, Proposition A.1 implies that layer $k+1$ is also jointly concave with respect to $\mathbf{x}_c$.

Therefore, by induction, all hidden representations and the final output $f(\mathbf{x})$ are jointly concave with respect to $\mathbf{x}_c$. Finally, fixing $\mathbf{x}_{\neg c}$, Definition 3.2 implies that f is partially jointly concave with respect to $\mathbf{x}_c$. □

Lemma 4.5. Let $h_{\text{mono}} : \mathbb{R}^d \to \mathbb{R}^k$ and denote its jth coordinate by $f_j(\mathbf{t}) \coloneqq \big[h_{\text{mono}}(\mathbf{t})\big]_j = \big(\exp(\mathbf{w}_j)^\top \mathbf{t} + b_j\big)_+^n$. Then, $\forall j \in [k]$, f_j is coordinatewise increasing in $\mathbf{t}$.

Proof. Define

$$g_j(\mathbf{t}) = \exp(\mathbf{w}_j)^\top \mathbf{t} + b_j, \qquad \eta_n(z) \coloneqq (z)_+^n = \min\{n, \max\{0, z\}\}.$$

Then,

$$f_j(\mathbf{t}) = \eta_n(g_j(\mathbf{t})).$$

Let $\mathbf{t} \preceq \mathbf{t}'$. Since $\exp(\mathbf{w}_j)$ is elementwise positive,

$$g_j(\mathbf{t}) \le g_j(\mathbf{t}').$$

Moreover, η_n is increasing. Therefore,

$$f_j(\mathbf{t}) = \eta_n(g_j(\mathbf{t})) \le \eta_n(g_j(\mathbf{t}')) = f_j(\mathbf{t}').$$

Hence, by Definition 3.3, f_j is coordinatewise increasing. □

Theorem 4.8. Let $f : \mathbb{R}^d \to \mathbb{R}$ denote the proposed COMONet with l hidden layers. Partition the input as $\mathbf{x} = (\mathbf{x}_m, \mathbf{x}_{\neg m}), \qquad \mathbf{x}_m \in \mathbb{R}^{|\mathcal{M}|}$. Then f is partially monotonic increasing with respect to $\mathbf{x}_m$.

Proof. Write

$$\mathbf{x}_m = (\mathbf{x}_{mn}, \mathbf{x}_{mv}, \mathbf{x}_{mc}).$$

We show that the output is increasing with respect to each component of $\mathbf{x}_m$, fixing all other variables.

Case 1: $x_i \in \mathbf{x}_{mn}$. The variable x_i is first processed by h_{mono}, which is coordinatewise increasing by Lemma 4.5. In subsequent layers, its information can pass through h_{mono}, h_{conv}, and h_{conc} units. By Lemmas 4.5, 4.1, and 4.3, all these units are coordinatewise increasing. Thus, by Proposition A.1, every hidden representation depending on x_i is increasing in x_i.

Case 2: $x_i \in \mathbf{x}_{mv}$. The variable x_i is processed through h_{conv} units. By Lemma 4.1, these units are coordinatewise increasing. Therefore, repeated application of Proposition A.1 shows that all hidden representations depending on x_i are increasing in x_i.

Case 3: $x_i \in \mathbf{x}_{mc}$. The variable x_i is processed through h_{conc} units. By Lemma 4.3, these units are coordinatewise increasing. Therefore, by Proposition A.1, all hidden representations depending on x_i are increasing in x_i.

Finally, the output layer uses elementwise positive weights, and hence preserves coordinatewise increasingness. Therefore, for every coordinate x_i in $\mathbf{x}_m$, fixing all other variables,

$$x_i \le x_i' \quad \Longrightarrow \quad f(x_i, \mathbf{x}_{\neg i}) \le f(x_i', \mathbf{x}_{\neg i}).$$

Hence, by Definition 3.3, f is partially monotonic increasing with respect to $\mathbf{x}_m$. □

Theorem 4.9. Let $f : \mathbb{R}^d \to \mathbb{R}$ satisfy the prescribed partial convexity and partial concavity constraints. For any $i \in \mathcal{CV}$ and $j \in \mathcal{CC}$, define

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + \alpha_{ij} x_i x_j, \qquad \alpha_{ij} \in \mathbb{R}.$$

Then, adding the bilinear term $\alpha_{ij} x_i x_j$ does not change the partial convexity of f with respect to x_i or the partial concavity of f with respect to x_j.

Proof. We first show that $\tilde{f}$ remains partially convex in x_i.

Fix all variables except x_i, including x_j. For any $x_i, x_i' \in \mathbb{R}$ and $\lambda \in [0, 1]$, we have

$$\alpha_{ij}\big(\lambda x_i + (1-\lambda)x_i'\big)x_j = \lambda \alpha_{ij} x_i x_j + (1-\lambda)\alpha_{ij} x_i' x_j.$$

Since f is partially convex in x_i,

$$f\big(\lambda x_i + (1-\lambda)x_i', \mathbf{x}_{\neg i}\big) \le \lambda f(x_i, \mathbf{x}_{\neg i}) + (1-\lambda) f(x_i', \mathbf{x}_{\neg i}).$$

Adding the linear equality above to both sides gives

$$\tilde{f}\big(\lambda x_i + (1-\lambda)x_i', \mathbf{x}_{\neg i}\big) \leq \lambda \tilde{f}(x_i, \mathbf{x}_{\neg i}) + (1-\lambda)\tilde{f}(x_i', \mathbf{x}_{\neg i}).$$

Therefore, by Definition 3.1, $\tilde{f}$ remains partially convex in x_i.

Similarly, fix all variables except x_j, including x_i. For any $x_j, x_j' \in \mathbb{R}$ and $\lambda \in [0,1]$,

$$\alpha_{ij} x_i\big(\lambda x_j + (1-\lambda)x_j'\big) = \lambda \alpha_{ij} x_i x_j + (1-\lambda)\alpha_{ij} x_i x_j'.$$

Since f is partially concave in x_j,

$$f\big(\lambda x_j + (1-\lambda)x_j', \mathbf{x}_{\neg j}\big) \geq \lambda f(x_j, \mathbf{x}_{\neg j}) + (1-\lambda) f(x_j', \mathbf{x}_{\neg j}).$$

Adding the linear equality above to both sides yields

$$\tilde{f}\big(\lambda x_j + (1-\lambda)x_j', \mathbf{x}_{\neg j}\big) \geq \lambda \tilde{f}(x_j, \mathbf{x}_{\neg j}) + (1-\lambda)\tilde{f}(x_j', \mathbf{x}_{\neg j}).$$

Therefore, by Definition 3.2, $\tilde{f}$ remains partially concave in x_j. This completes the proof. □

Theorem 4.10. Let $f : \mathbb{R}^d \to \mathbb{R}$ satisfy the prescribed partial convexity or partial concavity constraints. For any two distinct variables $i, j \in \mathcal{CV}$ or $i, j \in \mathcal{CC}$, define

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}) + \beta_{ij} x_i x_j, \qquad \beta_{ij} \in \mathbb{R}.$$

Then, adding the bilinear term $\beta_{ij} x_i x_j$ preserves the assigned variable-wise convexity or concavity. However, it does not necessarily preserve joint convexity or joint concavity over the pair (x_i, x_j).

Proof. We first consider the case $i, j \in \mathcal{CV}$. Fix all variables except x_i, including x_j. For any $x_i, x_i' \in \mathbb{R}$ and $\lambda \in [0,1]$,

$$\beta_{ij}\big(\lambda x_i + (1-\lambda)x_i'\big)x_j = \lambda \beta_{ij} x_i x_j + (1-\lambda)\beta_{ij} x_i' x_j.$$

Thus, the bilinear term is affine in x_i when x_j is fixed. Since affine functions satisfy the convexity inequality with equality, adding this term does not affect partial convexity in x_i. The same argument applies to x_j. Therefore, $\tilde{f}$ preserves variable-wise convexity for both x_i and x_j.

The case $i, j \in \mathcal{CC}$ follows analogously. When all variables except one are fixed, the bilinear term is affine in the remaining variable. Since affine functions also satisfy the concavity inequality with equality, adding $\beta_{ij} x_i x_j$ does not affect partial concavity in either x_i or x_j.

It remains to show that joint convexity or joint concavity over the pair (x_i, x_j) is not necessarily preserved. To see this, consider the base function $f_0(x_i, x_j) = 0$, which is both jointly convex and jointly concave. After adding the bilinear interaction, we obtain

$$\tilde{f}(x_i, x_j) = f_0(x_i, x_j) + \beta_{ij} x_i x_j = \phi(x_i, x_j), \qquad \phi(x_i, x_j) = \beta_{ij} x_i x_j.$$

We use the Jensen gap of ϕ, defined as the difference between the value of ϕ at a convex combination of two points and the corresponding convex combination of its function values. For any two points $u = (u_i, u_j)$, $v = (v_i, v_j)$, and $\lambda \in [0,1]$, this gap is

$$\begin{aligned}\Delta_\phi(u, v; \lambda) &\coloneqq \phi(\lambda u + (1-\lambda)v) - \lambda\phi(u) - (1-\lambda)\phi(v) \\ &= -\lambda(1-\lambda)\beta_{ij}(u_i - v_i)(u_j - v_j).\end{aligned}$$

Indeed, $(u_i - v_i)(u_j - v_j)$ can take either sign depending on whether the two coordinates move in the same or opposite directions. For any $\beta_{ij} \neq 0$, the Jensen gap can therefore be positive for some choices of u, v and negative for others. Hence, ϕ is neither jointly convex nor jointly concave in general. Therefore, adding the bilinear term does not necessarily preserve joint convexity or joint concavity over the pair (x_i, x_j). □

B Flow diagrams of COMONet

Fig. 6 present the flow structure of COMONet to demonstrate how each variable group ($\mathbf{x}_{cv}$, $\mathbf{x}_{mv}$, $\mathbf{x}_{cc}$, $\mathbf{x}_{mc}$, $\mathbf{x}_{mn}$, $\mathbf{x}_u$) contributes to the final prediction through their respective computational flows. Each subfigure highlights the specific path for a variable group, represented by bold dashed lines, showing how the input is processed through layers to produce the final output. This detailed visualization helps to clarify the role and influence of each group of variables in the model's overall architecture.

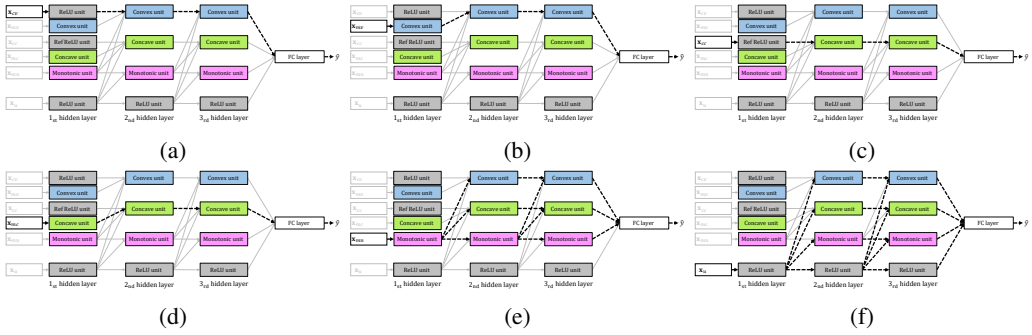
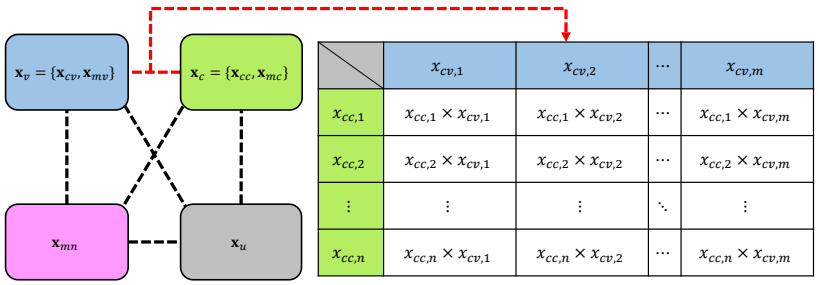


Figure 6: Flow diagrams representing the computational flows for each variable group in COMONet. Bold dashed lines indicate the paths followed by individual variable groups (a) $\mathbf{x}_{cv}$, (b) $\mathbf{x}_{mv}$, (c) $\mathbf{x}_{cc}$, (d) $\mathbf{x}_{mc}$, (e) $\mathbf{x}_{mn}$, (f) $\mathbf{x}_u$.

C Interaction layer of COMONet

Fig. 7 shows the all needed pairwise cross interactions between $\mathbf{x}_{cv}$ and $\mathbf{x}_{cc}$.



	$x_{cv,1}$	$x_{cv,2}$	$\cdots$	$x_{cv,m}$
$x_{cc,1}$	$x_{cc,1} \times x_{cv,1}$	$x_{cc,1} \times x_{cv,2}$	$\cdots$	$x_{cc,1} \times x_{cv,m}$
$x_{cc,2}$	$x_{cc,2} \times x_{cv,1}$	$x_{cc,2} \times x_{cv,2}$	$\cdots$	$x_{cc,2} \times x_{cv,m}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$x_{cc,n}$	$x_{cc,n} \times x_{cv,1}$	$x_{cc,n} \times x_{cv,2}$	$\cdots$	$x_{cc,n} \times x_{cv,m}$

Figure 7: Cross interaction terms between convex and concave variables could be captured by Interaction layer.

Fig. 8 shows the all possible pairwise intra interactions between $\mathbf{x}_{cc}$-$\mathbf{x}_{cc}$ or $\mathbf{x}_{cv}$-$\mathbf{x}_{cv}$.

	$x_{cc,1}$	$x_{cc,2}$	$x_{cc,3}$	$\cdots$	$x_{cc,n}$
$x_{cc,1}$		$x_{cc,1} \times x_{cc,2}$	$x_{cc,1} \times x_{cc,3}$	$\cdots$	$x_{cc,1} \times x_{cc,n}$
$x_{cc,2}$			$x_{cc,2} \times x_{cc,3}$	$\cdots$	$x_{cc,2} \times x_{cc,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
$x_{cc,n-1}$				$\ddots$	$x_{cc,n-1} \times x_{cc,n}$
$x_{cc,n}$				$\cdots$	

	$x_{cv,1}$	$x_{cv,2}$	$x_{cv,3}$	$\cdots$	$x_{cv,n}$
$x_{cv,1}$		$x_{cv,1} \times x_{cv,2}$	$x_{cv,1} \times x_{cv,3}$	$\cdots$	$x_{cv,1} \times x_{cv,n}$
$x_{cv,2}$			$x_{cv,2} \times x_{cv,3}$	$\cdots$	$x_{cv,2} \times x_{cv,n}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
$x_{cv,n-1}$				$\ddots$	$x_{cv,n-1} \times x_{cv,n}$
$x_{cv,n}$				$\cdots$	

Figure 8: Intra-interaction terms between Convex-convex and Concave-concave variables could be captured by Interaction layer.

D Configuration flexibility of COMONet

Fig. 9 and Fig. 10 shows various COMONet configurations, demonstrating the flexibility of the proposed method. Fig. 9 illustrates the $2^4 - 1 = 15$ possible configurations obtained by dividing the variable groups into four categories: $\mathbf{x}_v$, $\mathbf{x}_c$, $\mathbf{x}_{mn}$, $\mathbf{x}_u$. While, table in Fig. 10 shows all $2^6 - 1 = 63$ possible configurations of COMONet. These configurations highlight the ability of COMONet to handle a wide range of input scenarios while maintaining consistent processing through its computational layers. This adaptability ensures that the model can be tailored to specific tasks by including or excluding certain variable groups as needed.

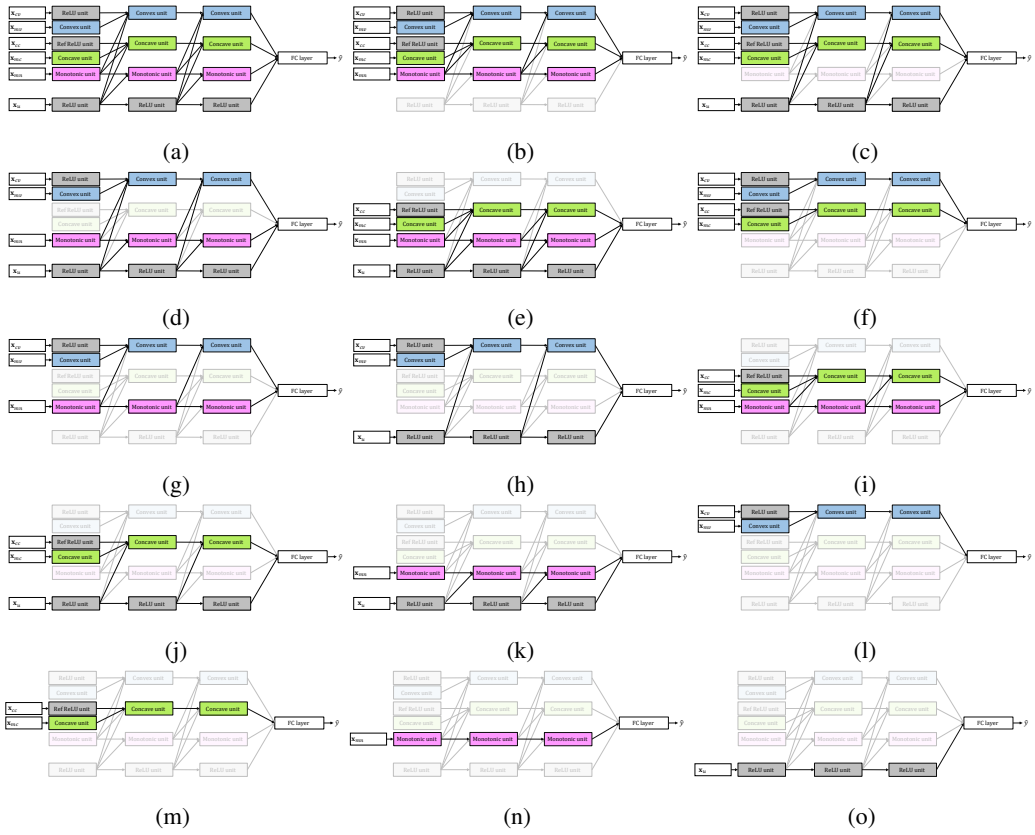


Figure 9: Examples structures of various configurations for COMONet.

	$\mathbf{x}_{cv}$	$\mathbf{x}_{mv}$	$\mathbf{x}_{cc}$	$\mathbf{x}_{mc}$	$\mathbf{x}_{mn}$	$\mathbf{x}_u$		$\mathbf{x}_{cv}$	$\mathbf{x}_{mv}$	$\mathbf{x}_{cc}$	$\mathbf{x}_{mc}$	$\mathbf{x}_{mn}$	$\mathbf{x}_u$		$\mathbf{x}_{cv}$	$\mathbf{x}_{mv}$	$\mathbf{x}_{cc}$	$\mathbf{x}_{mc}$	$\mathbf{x}_{mn}$	$\mathbf{x}_u$		$\mathbf{x}_{cv}$	$\mathbf{x}_{mv}$	$\mathbf{x}_{cc}$	$\mathbf{x}_{mc}$	$\mathbf{x}_{mn}$	$\mathbf{x}_u$
	6 types of features						16	✗	✓	✓	✓	✗	✓	33	✓	✓	✗	✗	✗	✓	50	✗	✓	✗	✗	✓	✗
1	✓	✓	✓	✓	✓	✓	17	✓	✓	✗	✗	✓	✓	34	✓	✗	✓	✗	✗	✓	51	✗	✗	✓	✗	✓	✗
	5 types of features						18	✓	✗	✓	✗	✓	✓	35	✗	✓	✓	✗	✗	✓	52	✗	✗	✗	✓	✓	✗
2	✓	✓	✓	✓	✓	✗	19	✗	✓	✓	✗	✓	✓	36	✓	✗	✗	✓	✗	✓	53	✓	✗	✗	✗	✗	✓
3	✓	✓	✓	✓	✗	✓	20	✓	✗	✗	✓	✓	✓	37	✗	✓	✗	✓	✗	✓	54	✗	✓	✗	✗	✗	✓
4	✓	✓	✓	✗	✓	✓	21	✗	✓	✗	✓	✓	✓	38	✗	✗	✓	✓	✗	✓	55	✗	✗	✓	✗	✗	✓
5	✓	✓	✗	✓	✓	✓	22	✗	✗	✓	✓	✓	✓	39	✓	✗	✗	✗	✓	✓	56	✗	✗	✗	✓	✗	✓
6	✓	✗	✓	✓	✓	✓		3 types of features						40	✗	✓	✗	✗	✓	✓	57	✗	✗	✗	✗	✓	✓
7	✗	✓	✓	✓	✓	✓	23	✓	✓	✓	✗	✗	✗	41	✗	✗	✓	✗	✓	✓		1 types of features					
	4 types of features						24	✓	✓	✗	✓	✗	✗	42	✗	✗	✗	✓	✓	✓	58	✗	✗	✗	✗	✗	✓
8	✓	✓	✓	✓	✗	✗	25	✓	✗	✓	✓	✗	✗		2 types of features						59	✗	✗	✗	✗	✓	✗
9	✓	✓	✓	✗	✓	✗	26	✗	✓	✓	✓	✗	✗	43	✓	✓	✗	✗	✗	✗	60	✗	✗	✗	✓	✗	✗
10	✓	✓	✗	✓	✓	✗	27	✓	✓	✗	✗	✓	✗	44	✓	✗	✓	✗	✗	✗	61	✗	✗	✓	✗	✗	✗
11	✓	✗	✓	✓	✓	✗	28	✓	✗	✓	✗	✓	✗	45	✗	✓	✓	✗	✗	✗	62	✗	✓	✗	✗	✗	✗
12	✗	✓	✓	✓	✓	✗	29	✗	✓	✓	✗	✓	✗	46	✓	✗	✗	✓	✗	✗	63	✓	✗	✗	✗	✗	✗
13	✓	✓	✓	✗	✗	✓	30	✓	✗	✗	✓	✓	✗	47	✗	✓	✗	✓	✗	✗							
14	✓	✓	✗	✓	✗	✓	31	✗	✓	✗	✓	✓	✗	48	✗	✗	✓	✓	✗	✗							
15	✓	✗	✓	✓	✗	✓	32	✗	✗	✓	✓	✓	✗	49	✓	✗	✗	✗	✓	✗							

Figure 10: All possible 63 configurations for COMONet.

E Detailed Experiment Descriptions

E.1 Training configurations

All experiments in this study were conducted on a system equipped with an Intel(R) Core(TM) i7-14700K 3.40 GHz processor, 64.0GB of DDR5 RAM, and running Microsoft Windows 11 Pro as the operating system. For GPU computations, we utilized an NVIDIA GeForce RTX 4070 Ti SUPER with 16.0GB of memory. The implementation of all models and experiments was carried out using Python (version 3.12.7) and the PyTorch [39] library (version 2.5.1) with CUDA (version 12.4). During model training, the ADAM [30] optimizer was employed as the stochastic optimization solver. Hyperparameters for each dataset were explored using a grid search strategy. For all datasets, the training process incorporated early stopping and, when necessary, exponentiated batch normalization (EBN) to enhance stability and efficiency. Table 4 summarizes the hyperparameter settings for the proposed method. Hyperparameters for both our proposed and the benchmark methods were selected via grid search over batch sizes of $128, 256, 512$, and learning rates of $0.05, 0.002, 0.005, 0.0005$. The number of epochs varied $[200, 3000]$ depending on the dataset. In general, experiments were conducted using 5-fold cross-validation repeated 5 times, and the mean and standard deviation (std) across 25 runs were reported. However, when the dataset was split into train/validation/test, experiments were conducted 5 times, and the mean and standard deviation from these runs were reported.

Table 4: Hyperparameters of COMONet for real-world datasets

Dataset	Number of parameters	Learning rate	Batch size
COMPAS	1457	0.005	128
Heart Disease	19649	0.002	128
Loan Defaulter	1489	0.0005	512
Blog Feedback	5137	0.0005	256
Auto-MPG	19265	0.005	128
Car sales	1195	0.005	109
Puzzle sales	1819	0.005	155
Wine Ratings	6753	0.005	512

Table 5: Hyperparameters of benchmark methods for real-world datasets

Methods	Dataset	Number of parameters	Learning rate	Batch size
SCNN	Car sales	1450	0.005	109
SCNN	Puzzle sales	5460	0.005	155
SCNN	Wine Ratings	9094	0.005	512
PenDer	Car sales	6401	0.005	109
PenDer	Puzzle sales	6529	0.005	155
PenDer	Wine Ratings	10241	0.005	512

E.2 Strategies for training stabilization

In this study, one of the key components of the proposed method, the exponentiated weight $(\exp(w))$, has the potential to explode as the weight value increases due to the nature of the exponential function. To address this issue, we adopted the weight initialization strategy introduced in the appendix of SMNN [29]. Specifically, the initial values of the exponentiated weight w were sampled from a uniform distribution within the range $[-20, 2]$, effectively mitigating exploding weights. Furthermore, the scaling parameter γ of the Exponentiated Batch Normalization was initialized to 0 to ensure stable training, by setting $\exp(\gamma)$ to 1. In addition, for activation functions such as ReLU and ReLU-n, we introduced a Leaky ReLU modification with $\alpha = 0.01$ in their off regions. This adjustment preserves

the intended properties of each activation function while improving training stability.

$$\text{Leaky ReLU-}n(x) = \begin{cases} \alpha x, & \text{if } x < 0, \\ x, & \text{if } 0 \le x \le n, \\ \alpha(x-n)+n, & \text{if } x > n. \end{cases} \tag{23}$$

E.3 Exponentiated batch normalization

In COMONet, different activation functions (e.g., ReLU and ReLU-n) are employed across layers, and the weights in certain layers are exponentiated. These differences in activation functions and weight transformations can lead to substantial shifts in the distributions of layer outputs. In some cases, such discrepancies may result in unstable learning dynamics, motivating the use of Batch Normalization [27]. However, standard Batch Normalization introduces a learnable scaling factor γ, which can become negative during training. If this scaling factor is negative, the normalized output is sign-reversed, which can violate shape constraints such as monotonicity, convexity, and concavity. In particular, such a sign reversal can change the sign of sensitivities with respect to constrained variables and thereby disrupt the structural guarantees. To avoid this issue, we use Exponentiated Batch Normalization (EBN), defined as

$$y = \frac{x - \mathbb{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} \cdot \exp(\gamma) + \beta. \tag{24}$$

Here, x denotes the layer output before normalization and y denotes the normalized output. The scaling factor is parameterized as $\exp(\gamma)$, which is strictly positive for all $\gamma \in \mathbb{R}$. Therefore, EBN applies an affine transformation with positive scale, preserving the sign of sensitivities propagated from the previous layer. Consequently, if the pre-normalized layer output is coordinatewise increasing with respect to a constrained variable, then the normalized output remains coordinatewise increasing. Similarly, positive scaling preserves convexity and concavity of the pre-normalized layer output.

E.4 Descriptions for real-world datasets

This section provides an overview of the real-world datasets used in the experiments. These datasets were derived from previously published benchmarks frequently cited in literature on monotonic and convex neural networks. The criteria for applying shape constraints followed the methodologies outlined in prior benchmark studies. While precisely defining shape constraints poses challenges, as mentioned in the conclusion, future research that focuses on identifying these constraints for specific variables could yield valuable insights. Additionally, some datasets contain instances with relatively small sample sizes, reflecting realistic challenges often encountered in practical applications. Effectively addressing such constraints is critical for developing robust and widely applicable models. Table 6 provides a summarized overview of each dataset, with detailed descriptions presented below (**Bold** text indicates monotonic decrease, while *italic* text denotes concavity.):

AutoMPG: The Auto-MPG dataset is a regression dataset with 7 variables and approximately 398 instances, used to predict a car's miles per gallon (mpg). It includes monotonic decreasing relationships between mpg and the variables **weight**, **displacement** and **horse power**.

Heart Disease: The Heart Disease dataset is a classification dataset with 13 variables, used to predict the presence or absence of heart disease in individuals. Among the variables, trestbps (resting blood pressure) and chol (cholesterol level) are known to have monotonic increasing relationships with the risk of heart disease.

COMPAS*: The COMPAS dataset is a binary classification dataset that predicts whether offenders in Florida will reoffend within two years based on criminal history data. It includes 13 variables, of which 4 (number of juvenile misdemeanor, number of other convictions, number of prior adult convictions, and number of juvenile felony) are known to have monotonic increasing relationships with recidivism risk. This dataset raises ethical concerns; however, it has been used in recent publications for comparison studies and remains relevant in research fields focused on fairness.

Blog Feedback: The BlogFeedback dataset is a regression dataset used to predict the number of comments a blog post will receive within 24 hours. It includes 276 variables, of which 8 variables (A51, A52, A53, A54, A56, A57, A58, A59) are known to have monotonic increasing relationships with the number of comments.

Loan Defaulter: The Loan Defaulter dataset is a classification dataset used to predict whether a customer will default on a loan. It includes loan data from 2007 to 2015 and consists of 28 variables, among which 5 variables have shape constraints. Number of public record bankruptcies and Debt to income ratio have monotonic increasing relationships with default risk, while **Credit score**, **Length of employment**, and **Annual income** have monotonic decreasing relationships with default risk.

Car Sales: The Car Sales dataset is a one-dimensional regression problem aimed at predicting monthly car sales (in thousands) based on the car price (in thousands). In this problem, the **price** variable is constrained to have a convex and monotonically decreasing relationship with car sales. The dataset consists of a total of 155 entries, with 109 used for training, 32 for testing, and 14 for validation.

Puzzle Sales: The Puzzle Sales dataset is a regression dataset designed to predict six-month sales of wooden jigsaw puzzles using variables derived from Amazon reviews. Three variables are used for prediction: (1) the average star rating, which is expected to have a monotonically increasing relationship with sales; (2) *the number of reviews*; and (3) *the word count of reviews*, both of which are expected to exhibit a monotonically increasing and concave relationship with sales. The dataset includes 156 training examples, 169 validation examples, and 200 test examples.

Wine Ratings: The Wine Ratings dataset is a regression dataset designed to predict wine scores on an 80–100 scale using various wine attributes. The dataset consists of 61 variables in total: 21 binary variables representing the country of production, 39 boolean variables derived from wine descriptions published by the Wine Enthusiast Magazine, and a continuous variable representing the wine's price. Among the 120,919 data entries, 84,642 were used for training, 12,092 for validation, and 24,185 for testing. The variable *price* was included in the training process with the expectation that it has a concave and monotonically increasing relationship with Wine Ratings.

Table 6: Descriptions for Real-world Benchmark Datasets

Dataset	Task	# Instances	# Features	# Constrained features	Monotonic-Convex (Concave) features	Monotonic features
Auto-MPG	Regression	398	7	3	– – – – – –	**weights, displacement, horse power**
Blog Feedback	Regression	54270	276	8	– – – – – –	A51,A52,A53,A54,A56,A57,A58,A59
COMPAS	Classification	6172	13	4	– – – – – –	number of prior adult convictions, number of juvenile felony, number of juvenile misdemeanor, number of other convictions
Heart Disease	Classification	303	13	2	– – – – – –	trestbps, chol
Loan Defaulter	Classification	488909	28	5	– – – – – –	number of public record bankruptcies, debt-to-income ratio, **credit score**, **length of employment**, **annual income**
Car Sales	Regression	155	1	1	**price**	– – – – – –
Puzzle Sales	Regression	525	3	3	*number of reviews*, *word count*	star rating
Wine Ratings	Regression	120919	61	1	*price*	– – – – – –

E.5 COMONet Configuration for hyperbolic paraboloid function (21)

The hyperbolic paraboloid function in (21) was learned using the structure in Fig. 11.

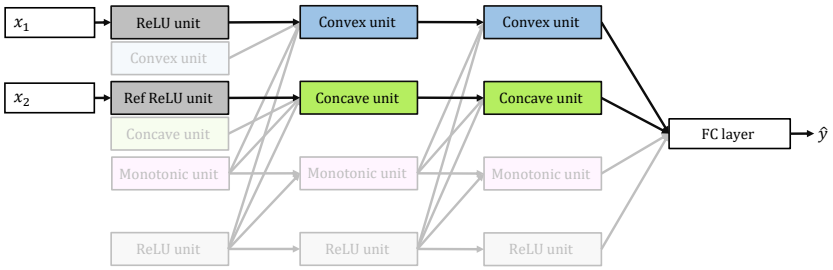


Figure 11: COMONet Configuration for hyperbolic paraboloid function (21).

F Supplemental report of experimental results

F.1 Tabulated numerical results for Fig. 5

Fig. 5 was created using the data in the following table.

Table 7: Results of Generalization test with (22)

Network	Training MSE	Test MSE	Denoised test MSE
COMONet ($\lambda = 0$)	0.01 ± 0.00	0.01 ± 0.00	0.01 ± 0.00
COMONet ($\lambda = 1$)	0.92 ± 0.14	0.99 ± 0.03	0.02 ± 0.01
COMONet ($\lambda = 2$)	3.90 ± 0.57	4.03 ± 0.28	0.06 ± 0.02
COMONet ($\lambda = 5$)	23.74 ± 3.41	24.88 ± 0.89	0.31 ± 0.10
COMONet ($\lambda = 10$)	96.84 ± 14.66	98.07 ± 6.97	0.79 ± 0.36
COMONet ($\lambda = 20$)	400.29 ± 55.23	402.42 ± 5.93	3.13 ± 1.32
Same Structure ($\lambda = 0$)	0.01 ± 0.01	0.01 ± 0.00	0.01 ± 0.00
Same Structure ($\lambda = 1$)	0.77 ± 0.14	1.28 ± 0.10	0.29 ± 0.04
Same Structure ($\lambda = 2$)	3.15 ± 0.46	5.31 ± 0.32	1.30 ± 0.22
Same Structure ($\lambda = 5$)	19.47 ± 3.21	32.77 ± 1.76	7.21 ± 1.15
Same Structure ($\lambda = 10$)	75.09 ± 10.27	128.25 ± 6.85	30.17 ± 5.60
Same Structure ($\lambda = 20$)	302.68 ± 44.74	510.34 ± 21.90	106.57 ± 11.04
MLP ($\lambda = 0$)	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
MLP ($\lambda = 1$)	0.28 ± 0.08	1.93 ± 0.12	0.97 ± 0.10
MLP ($\lambda = 2$)	1.03 ± 0.27	8.15 ± 0.59	4.29 ± 0.49
MLP ($\lambda = 5$)	8.23 ± 2.66	48.64 ± 4.03	23.19 ± 3.57
MLP ($\lambda = 10$)	26.95 ± 8.68	195.11 ± 14.12	97.22 ± 8.66
MLP ($\lambda = 20$)	148.87 ± 42.43	722.66 ± 61.73	334.91 ± 71.79

Fig. 12 illustrates the same structure model employed in experiments on (22).

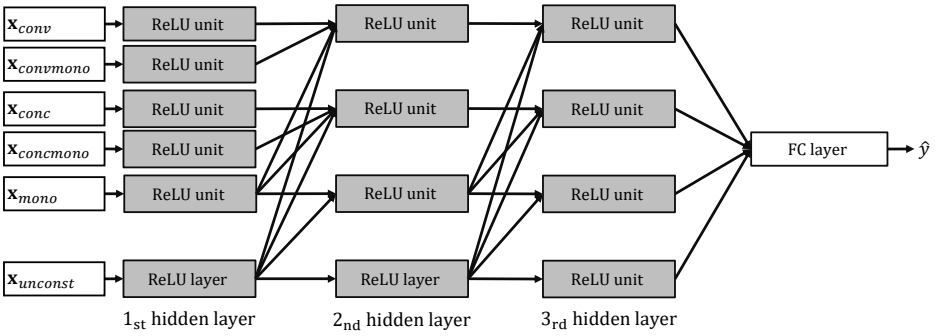


Figure 12: Same structure network

F.2 Results on PenDer

Table 8 shows Test MSE and Convexity Score $\mathcal{C}_k$ and Monotonic Score $\mathcal{M}_k$ of PenDer [22].

Table 8: Test MSE, Convexity Score and Monotonic Score of PenDer on Real-world Dataset.

Dataset	Test MSE ↓	$\mathcal{M}_k$	$\mathcal{C}_k$
Car Sales (conv)	10411 ± 107	1	1
Car Sales (conv, decr)	10415 ± 104	1	1
Puzzle Sales (conc)	9428 ± 113	1	0.98 ± 0.008
Puzzle Sales (conc, incr)	9519 ± 92	1	0.99 ± 0.004
Wine Ratings (conc)	5.19 ± 0.11	1	0.99 ± 0.000
Wine Ratings (conc, incr)	5.27 ± 0.20	0.99 ± 0.000	0.99 ± 0.000

F.3 Hyperbolic paraboloid with interaction term

We tested whether COMONet can capture convex–concave interactions using the synthetic function

$$f(x_1, x_2) = (x_1 - 0.5)^2 - (x_2 - 0.5)^2 + \lambda\, x_1 x_2, \quad \lambda \in \{0, 1, 2, 5, 10\}.$$

Models with the interaction layer enabled (ON) were compared against models without it (OFF) using the same 1,000 samples, 80/20 split, and cross-validation protocol as in the main experiments. As shown in Table 9, the ON model maintains low error for increasing λ, while the OFF model fails to capture the interaction term.

Table 9: Interaction-layer ablation on the convex–concave synthetic function.

Interaction	λ	Train MSE	Test MSE
ON	0	0.0001 ± 0.0000	0.0001 ± 0.0001
ON	1	0.0003 ± 0.0011	0.0004 ± 0.0012
ON	2	0.0019 ± 0.0045	0.0016 ± 0.0045
ON	5	0.0053 ± 0.0098	0.0056 ± 0.0109
ON	10	0.0209 ± 0.0491	0.0188 ± 0.0355
OFF	0	0.0001 ± 0.0000	0.0001 ± 0.0001
OFF	1	0.0080 ± 0.0020	0.0086 ± 0.0017
OFF	2	0.0318 ± 0.0086	0.0357 ± 0.0085
OFF	5	0.1951 ± 0.0738	0.2295 ± 0.0928
OFF	10	0.7576 ± 0.2535	0.8151 ± 0.1596

F.4 Trustworthy test using LIME

Fig. 13 shows the LIME [41] values for x_1 and x_3 at the different noise levels. These results validate that embedding shape constraints yields more reliable interpretations.

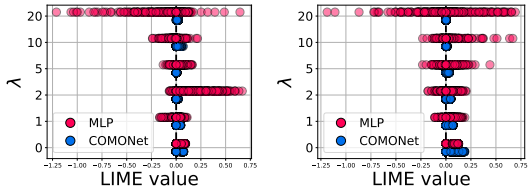


Figure 13: LIME values for x_1 (Left) and x_3 (Right), with data sampled at x_2 and x_4 fixed at their mean values in the test dataset. The values are shown for COMONet (red) and MLP (blue) across different λ values.

F.5 Figures with the experimental results with (22)

Fig. 14 presents contour plots obtained by varying the noise parameter λ from 0 to 5. Among the two displayed variables, x_1 is monotonically increasing and x_2 is convex with respect to y.

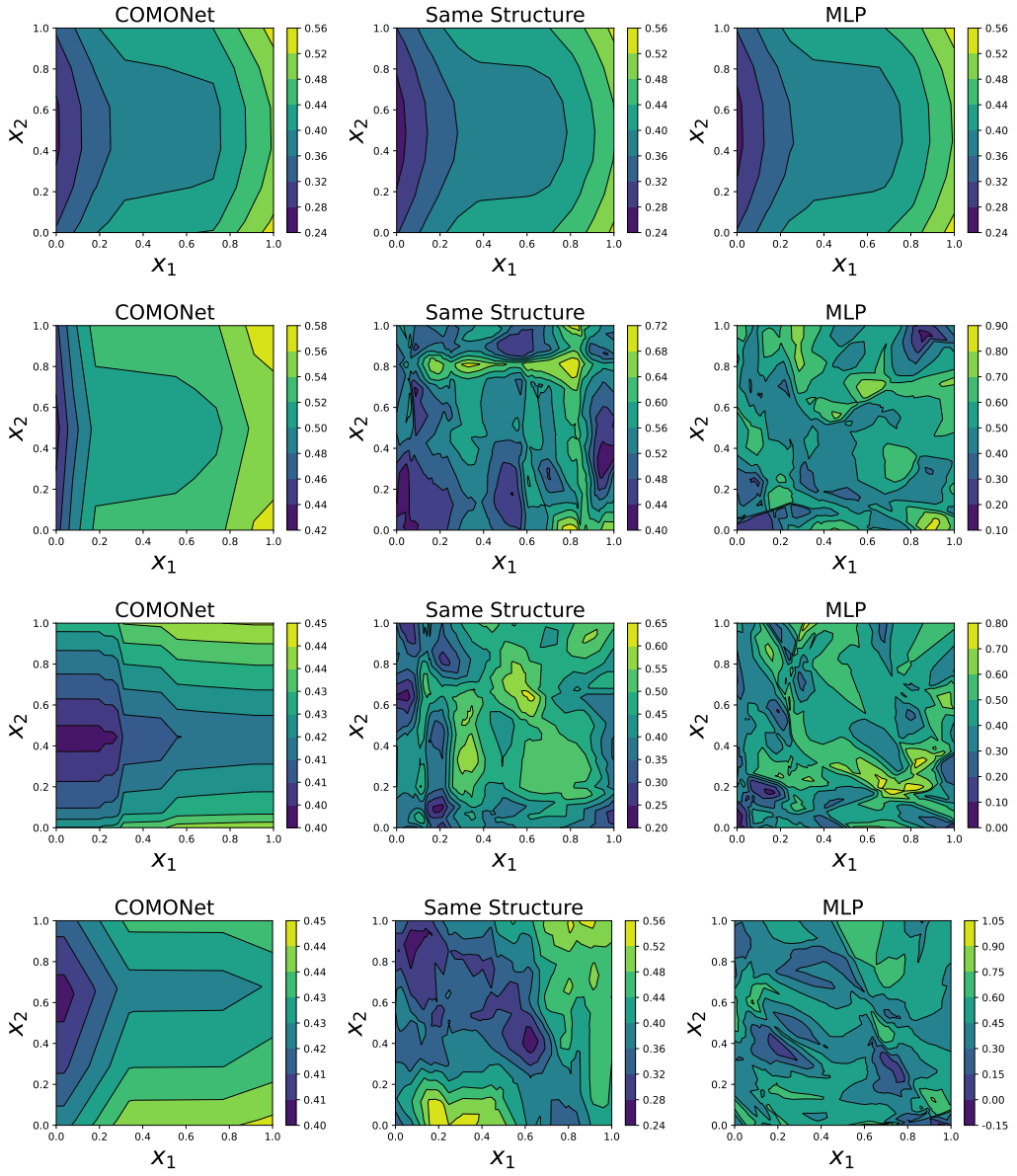


Figure 14: Contours of x_1 and x_2 varying within [0,1] when $\lambda = \{0, 1, 2, 5\}$, with x_3 and x_4 fixed at 0.5. **Left**: COMONet, **Center**: Same structure, **Right**: MLP.

Fig. 15 presents contour plots obtained by varying the noise parameter λ from 0 to 5. Among the two displayed variables, x_2 is convex and x_3 is monotonic-convex with respect to y.

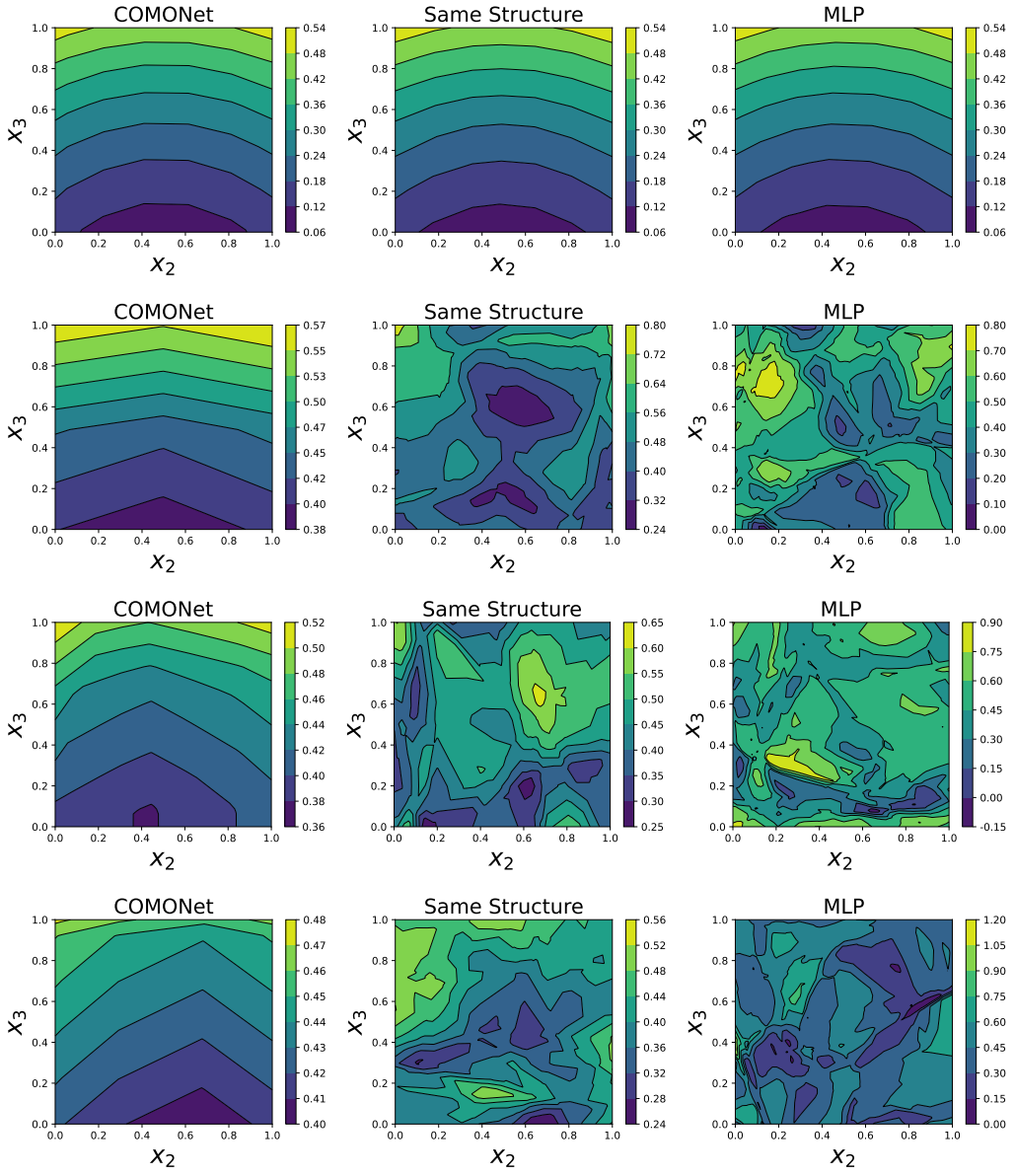
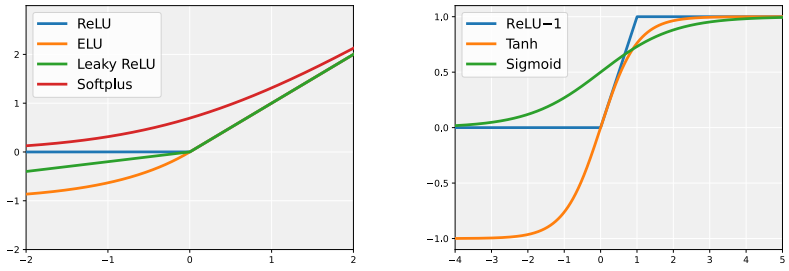


Figure 15: Contours of x_2 and x_3 varying within [0,1] when $\lambda = \{0, 1, 2, 5\}$, with x_1 and x_4 fixed at 0.5. **Left**: COMONet, **Center**: Same structure, **Right**: MLP.

G Ablation studies

G.1 Various activation settings

Fig. 16 and Table 10 represent various activation functions that satisfies conditions of Monotonicity and convexity. Among them Monotonic-convex activations can alternate ReLU and convex activations can alternate ReLU-n. Table 11 shows that no performance differences across various activations.



(a) (b)

Figure 16: **Visualization of various activation functions:** activation functions in (a) is monotonic-convex function that can be used for $h_{\text{conv}}, h_{\text{relu}}, h_{\text{conc}}, h_{\text{ref-relu}}$ and (b) shows monotonic-wavy function that can be used for h_{mono}

Table 10: Various Activation functions

Activation functions	Formula	Monotonicity	Convexity	Remark
ReLU	$\sigma(x) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{if } x < 0 \end{cases}$	✓	✓	
Leaky ReLU	$\sigma(x) = \begin{cases} x & \text{if } x \geq 0, \\ \alpha x & \text{if } x < 0 \end{cases}$	✓*	✓*	if $0 < \alpha < 1$
ELU	$\sigma(x) = \begin{cases} x & \text{if } x \geq 0, \\ \alpha(e^x - 1) & \text{if } x < 0 \end{cases}$	✓*	✓*	if $0 < \alpha \leq 1$
Softplus	$\sigma(x) = \log(1 + \exp(x))$	✓	✓	
Absolute	$\sigma(x) = \lvert x \rvert$	✗	✓	unsuitable
ReLU-n	$\sigma(x) = \begin{cases} n & \text{if } x \geq n, \\ x & \text{if } 0 \leq x < n, \\ 0 & \text{if } x < 0 \end{cases}$	✓	✗	
Tanh	$\sigma(x) = \tanh(x)$	✓	✗	
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	✓	✗	

* Indicates conditional satisfaction under specific parameter settings.

Table 11: Performance comparison among various activations settings († Indicates statistical ties.)

Activation		Auto MPG	Heart Disease	Remark
Monotonic activation	Convex activation	MSE ↓	Test Acc ↑	
ReLU-1	ReLU	7.38 ± 1.32†	0.85 ± 0.04†	
ReLU-6	ReLU	7.38 ± 1.32†	0.85 ± 0.04†	
Sigmoid	ReLU	9.04 ± 2.20†	0.84 ± 0.05†	
Leaky relu-1	Leaky ReLU	$\mathbf{7.03 \pm 1.54}$	$\mathbf{0.87 \pm 0.04}$	$\alpha = 0.01$
Sigmoid	ELU	8.67 ± 2.36†	$\mathbf{0.87 \pm 0.04}$	$\alpha = 0.01$
Tanh	ELU	7.20 ± 1.59†	0.86 ± 0.05†	$\alpha = 0.01$
Sigmoid	Softplus	10.09 ± 2.62	0.84 ± 0.06†	
Tanh	Softplus	7.46 ± 1.57†	$\mathbf{0.87 \pm 0.04}$	

G.2 The effect of upward directional connections

To verify whether COMONet can effectively learn interactions between variable groups, we conducted an ablation study. The function was designed as in (25), where the output y is determined by three variables x_1, x_2 and x_3. Among these, x_1 belongs to $\mathbf{x}_{cv}$, which has a convex relationship with y, while x_2 belongs to $\mathbf{x}_{mn}$, which has a monotonic relationship with y. To examine interaction effects, we varied the coefficient of interaction term α from 0 to 20. We generated a dataset of 1,000 samples from (25), using 800 for training and 200 for testing, and evaluated the models using 5-fold cross-validation repeated five times. We compared four models with different levels of connectivity in COMONet, as shown in Fig. 18: (a) Not Connected, where variable groups are completely separated; (b) Sparse-to-Specific, which allows connections only between $\mathbf{x}_{cv}$ and $\mathbf{x}_{mn}$ to facilitate learning the known interaction betweenx_1 and x_2 ; (c) Dense-to-Specific, where only the connection between $\mathbf{x}_{cv}$ and $\mathbf{x}_{mn}$ is removed; and (d) Fully Connected, where all groups are interconnected. As shown in Table 12 and Fig. 17, the Not Connected and Dense-to-Specific models exhibited increasing Test MSE as α increased, indicating a failure to capture interaction effects. In contrast, the Sparse-to-Specific and Fully Connected models maintained relatively low Test MSE despite increasing α, demonstrating their ability to effectively learn the interaction between x_1 and x_2. These results empirically validate that COMONet can capture interactions between separated variable groups.

$$
\begin{aligned}
y = (x_1 - 1)^2 + \sqrt{x_2} + \alpha x_1 x_2^2 + \sin(2\pi x_3), \\
x_i \in [0, 2], \forall i \in \{1, 2, 3\}, \\
\alpha, \in \{0, 1, 2, 5, 10, 20\}.
\end{aligned} \tag{25}
$$

Table 12: Comparison of Test MSE Performance.

α	Not connected	Sparse-to-Specific	Dense-to-Specific	Fully connected
$\alpha = 0$	0.01 ± 0.01	0.01 ± 0.01	0.02 ± 0.00	0.03 ± 0.05
$\alpha = 1$	0.48 ± 0.03	0.04 ± 0.02	0.59 ± 0.22	0.49 ± 0.20
$\alpha = 2$	1.87 ± 0.11	0.21 ± 0.26	1.94 ± 0.29	0.48 ± 0.79
$\alpha = 5$	11.75 ± 0.98	0.33 ± 0.08	12.14 ± 0.78	7.76 ± 6.09
$\alpha = 10$	47.39 ± 6.76	1.79 ± 0.64	47.20 ± 6.56	3.24 ± 1.99
$\alpha = 20$	193.79 ± 40.07	11.15 ± 12.95	191.13 ± 32.80	15.41 ± 12.44

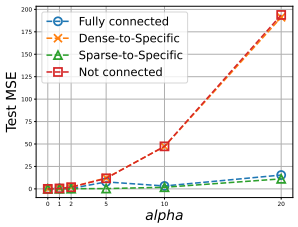


Figure 17: Visualization of The results in Table 12.

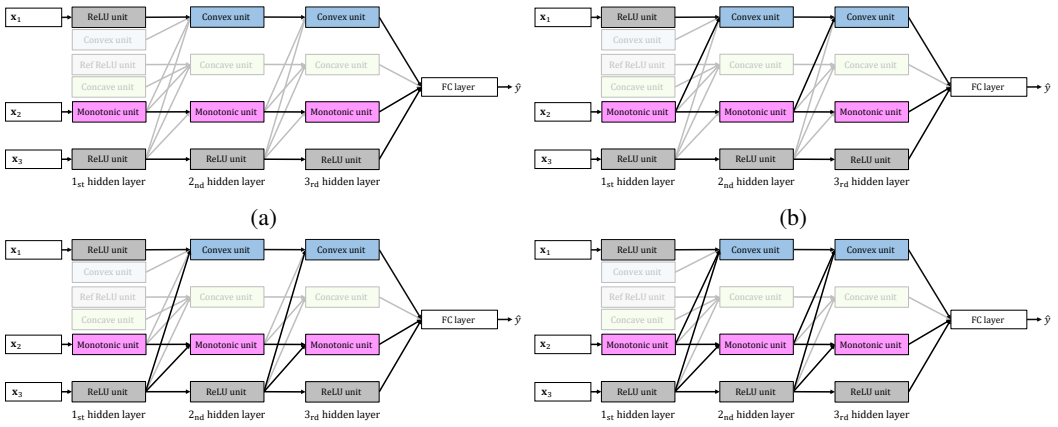


Figure 18: COMONet: (a) Not connected, (b) Sparse-to-Specific, (c) Dense-to-Specific, and (d) Fully connected.

G.3 Ablation study on Misassigned Constraints and Single-variable Learning Capability

To evaluate the importance of assigning each variable from (22) to the correct constraint branch, we conducted an ablation where the cosine-shaped term $cos(2\pi x_4)$ was intentionally routed to incorrect branches (monotonic, convex, or monotonic–convex) instead of the intended unconstrained branch. As shown in Table 13, misassigning x_4 leads to substantially higher Train/Test MSE, demonstrating that each constraint branch serves a distinct purpose and that improper constraint–variable alignment severely degrades accuracy. When x_4 is correctly placed in the unconstrained branch, the model achieves the lowest error, confirming the necessity of an unconstrained pathway for non-monotonic and non-convex components.

Table 13: Ablation study on the assignment of x_4 to different constraint branches.

Case	x_4 **branch**	**Constraint**	**Train MSE (mean $\pm$ std)**	**Test MSE (mean $\pm$ std)**
Case 1	$x_4 \to x_m$	monotonically increasing	0.472 ± 0.069	0.483 ± 0.061
Case 2	$x_4 \to x_{mv}$	convex & monotonically increasing	0.468 ± 0.094	0.470 ± 0.081
Case 3	$x_4 \to x_{cv}$	convex	0.067 ± 0.116	0.063 ± 0.108
Case 4	$x_4 \to x_u$	unconstrained	$\mathbf{0.004 \pm 0.003}$	$\mathbf{0.005 \pm 0.004}$

We also evaluated whether each constrained component can independently model the corresponding single-variable component from (22). The target function was decomposed into four single-variable terms, each associated with a distinct shape constraint (monotonic, convex, monotonic–convex, and unconstrained). As shown in Table 14, training each branch on its respective term resulted in very small MSE, demonstrating that all constrained subnetworks accurately capture their intended shape behavior and reliably express the corresponding single-variable functions.

Table 14: Ablation study showing each branch's ability to independently learn its single-variable component in (22).

Feature	**Constraint**	**Train MSE (mean $\pm$ std)**	**Test MSE (mean $\pm$ std)**
x_1	monotonically increasing	0.002 ± 0.003	0.002 ± 0.002
x_2	convex	0.001 ± 0.001	0.001 ± 0.001
x_3	convex & monotonically increasing	0.001 ± 0.001	0.001 ± 0.001
x_4	unconstrained	0.002 ± 0.002	0.003 ± 0.001

H Impact Statements

Imposing shape constraints in neural networks can substantially improve model reliability, resilience, and interpretability—qualities that are especially valuable in sectors like manufacturing, finance, and healthcare where data may be limited or noisy. Nevertheless, if constraints inadvertently encode stereotypes or adverse assumptions about protected attributes (such as age, gender, or ethnicity), they risk perpetuating unfair outcomes. To guard against this, practitioners should systematically evaluate constraint behavior across different demographic groups and embed fairness checks at every stage of model development and deployment. Moreover, promoting transparency by publishing constraint definitions and associated validation tools under an open-source license fosters accountability and helps ensure these methods serve broad societal interests.

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: We clearly state the main claims in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discussed our limitations in Section 6 (Conclusion).

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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: We prove all theorems from the main text in full detail, in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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: [Yes]

   Justification: We fully disclose all information required to reproduce the paper's main experimental results through the main text, Appendix, and supplemental material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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: We provide the data and code in supplemental material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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: [Yes]

   Justification: We provide detailed experiment settings in the Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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: We report both the mean and the standard deviation over repeated runs, providing suitably defined error bars and clear measures of variability.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: [Yes]

   Justification: We provide detailed information on computer resources in the Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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: We used the COMPAS dataset to compare our method against existing benchmarks but explicitly disclosed its known ethical issues in Appendix E.4. While it raises ethical concerns for some, it is considered acceptable by the majority.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss both the potential positive and negative societal impacts of our work within the paper, especially at section 6 Conclusion and Impact Statement in Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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: Our paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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: [Yes]

    Justification: We properly credit the creators or original owners of all assets used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We thoroughly document all newly introduced assets and provide the documentation alongside them.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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: Our paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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: This research does not involve LLMs as any important, original, or non standard components. We used LLMs solely for editing (grammar, spelling, word choice).

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.