T-SKM-Net: Trainable Neural Network Framework for Linear Constraint Satisfaction via Sampling Kaczmarz-Motzkin Method

Haoyu Zhu[1], **Yao Zhang**[1], **Jiashen Ren**[1], **Qingchun Hou**[1*]

[1]Zhejiang University

haoyu.22@intl.zju.edu.cn, eezhangyao@zju.edu.cn, jiashen.22@intl.zju.edu.cn, houqingchun@zju.edu.cn

Abstract

Neural network constraint satisfaction is crucial for safety-critical applications such as power system optimization, robotic path planning, and autonomous driving. However, existing constraint satisfaction methods face efficiency-applicability trade-offs, with hard constraint methods suffering from either high computational complexity or restrictive assumptions on constraint structures. The Sampling Kaczmarz-Motzkin (SKM) method is a randomized iterative algorithm for solving large-scale linear inequality systems with favorable convergence properties, but its argmax operations introduce non-differentiability, posing challenges for neural network applications. This work proposes the Trainable Sampling Kaczmarz-Motzkin Network (T-SKM-Net) framework and, for the first time, systematically integrates SKM-type methods into neural network constraint satisfaction. The framework transforms mixed constraint problems into pure inequality problems through null space transformation, employs SKM for iterative solving, and maps solutions back to the original constraint space, efficiently handling both equality and inequality constraints. We provide theoretical proof of post-processing effectiveness in expectation and end-to-end trainability guarantees based on unbiased gradient estimators, demonstrating that despite non-differentiable operations, the framework supports standard backpropagation. On the DCOPF case118 benchmark, our method achieves 4.27ms/item GPU serial forward inference with 0.0025% max optimality gap with post-processing mode and 5.25ms/item with 0.0008% max optimality gap with joint training mode, delivering over 25× speedup compared to the pandapower solver while maintaining zero constraint violations under given tolerance.

Code — https://github.com/IDO-Lab/T-SKM-Net
Extended version — https://arxiv.org/abs/2512.10461

1 Introduction

Constrained decision problems arise across various disciplines, such as the DC Optimal Power Flow (DCOPF) problem in power systems (Carpentier 1962), which requires minimizing operational costs while satisfying physical and security constraints of the power system. However, directly solving these problems using optimization solvers requires substantial computational time. Therefore, in scenarios demanding rapid or even real-time responses, traditional solvers often fail to meet timing requirements (Scutari and Sun 2018), motivating researchers to explore more efficient approximate solution methods.

In recent years, deep learning (Goodfellow et al. 2016) has demonstrated powerful function approximation capabilities across various complex tasks (Hornik, Stinchcombe, and White 1989; Cybenko 1989; LeCun, Bengio, and Hinton 2015), providing new insights for solving constrained optimization problems (Smith 1999; Cappart et al. 2023; Hou et al. 2023; Liu et al. 2024). Constrained decision problems can be transformed into single forward inference of neural networks by learning the mapping relationship from problem parameters to decision variables. However, neural network outputs often cannot guarantee strict satisfaction of the original constrained decision problem's constraints, which limits their use in safety-critical applications.

To address this challenge, researchers have proposed various methods for integrating constraints into neural networks. Existing methods can be mainly categorized into soft constraints and hard constraints. Soft constraint methods (Raissi, Perdikaris, and Karniadakis 2019) indirectly handle constraints by incorporating constraint violation terms as penalty terms in the loss function. While these methods are simple to implement and maintain network differentiability, they cannot strictly guarantee constraint satisfaction, posing potential risks in safety-critical applications. Hard constraint methods attempt to strictly satisfy constraints at network output, including differentiable optimization layers (Amos and Kolter 2017; Donti, Rolnick, and Kolter 2021; Min and Azizan 2025), parameterized feasible space methods (Tordesillas, How, and Hutter 2023; Zhang, Tabas, and Zhang 2024), and decision rule approaches (Constante-Flores, Chen, and Li 2025). However, these methods still face challenges such as high computational complexity, requirement for pre-computing feasible points, or limited expressiveness when handling input-dependent dynamic linear constraints.

To address these limitations, this work proposes the Trainable Sampling Kaczmarz-Motzkin Network (T-SKM-Net) framework. The main contributions include: (1) First integration of the Sampling Kaczmarz-Motzkin method to neu-

*Corresponding author.

ral network linear constraint satisfaction for input-dependent dynamic constraints; (2) Theoretical proof that this method can serve as an effective approximation of L_2 projection, providing theoretical foundation for its use as a post-processing strategy; (3) Theoretical demonstration of end-to-end trainability of the framework through unbiased gradient estimators (Robbins and Monro 1951; Bottou 2010; Goda and Kitade 2023), addressing training challenges caused by non-differentiable argmax operations.

2 Related Work

Neural network constraint satisfaction methods can be categorized into soft constraints and hard constraints based on constraint handling approaches, where hard constraint methods can be further subdivided into three main paradigms.

2.1 Soft Constraint Methods

Soft constraint strategies indirectly handle constraints by incorporating constraint violation terms as penalty terms in the loss function (Raissi, Perdikaris, and Karniadakis 2019; Yang, Zhang, and Karniadakis 2020). While these methods are simple to implement and preserve network differentiability, they cannot strictly guarantee constraint satisfaction, posing potential risks in safety-critical applications.

2.2 Hard Constraint Methods

Differentiable Optimization Layers and Projection Methods These methods ensure constraint satisfaction by embedding optimization problems into neural networks or employing projection operations (Chen et al. 2021). OptNet (Amos and Kolter 2017) embeds quadratic programming layers (Butler and Kwon 2023) into neural networks but suffers from high computational complexity. DC3 (Donti, Rolnick, and Kolter 2021) adopts completion and correction strategies but cannot strictly guarantee equality constraint satisfaction due to linear approximations. HardNet (Min and Azizan 2025) provides closed-form projection expressions but relies on restrictive assumptions such as constraint matrix full rank. GLinSAT (Zeng et al. 2024) solves an entropy-regularized linear program with an accelerated first-order method to impose general linear constraints, but still requires an inner iterative solver per batch. Recent works also design feasibility-seeking or projection-like layers with guarantees, such as homeomorphic projection for sets homeomorphic to a unit ball (including all compact convex sets and some nonconvex sets) (Liang, Chen, and Low 2023), feasibility-seeking NNs via unrolled violation minimization (Nguyen and Donti 2025), and feasibility-restoration mappings for AC-OPF (Han et al. 2024), but they introduce additional optimization or invertible-network components inside the layer.

Parameterized Feasible Space Methods These methods parameterize neural network outputs to feasible regions. Zhang, Tabas, and Zhang (2024) uses gauge maps for polyhedra mapping but requires the origin to be a strict interior point, making equality constraint handling difficult. RAYEN (Tordesillas, How, and Hutter 2023) employs geometric transformations but requires offline computation of feasible points, limiting applicability to input-dependent constraints.

Decision Rule Methods Recent work introduces decision rule-based methods from stochastic optimization. (Constante-Flores, Chen, and Li 2025) et al. propose a framework combining task and safety networks through convex combinations. However, this approach has limited expressiveness and requires convex constraint sets. Preventive learning (Zhao et al. 2023) calibrates linear inequality constraints during training to anticipate DNN prediction errors and ensure feasibility without post-processing, but is tailored to convex linear constraints.

2.3 Sampling Kaczmarz-Motzkin Method

The Kaczmarz method (Karczmarz 1937) and Motzkin relaxation method (Motzkin and Schoenberg 1954) are classical iterative techniques for solving linear systems and inequalities, respectively. Strohmer and Vershynin (2009) proved exponential convergence for the randomized Kaczmarz algorithm. Loera, Haddock, and Needell (2017) unified these approaches in the Sampling Kaczmarz-Motzkin (SKM) method for large-scale linear inequality systems. Morshed, Islam, and Noor-E-Alam (2022) further improved SKM with global linear convergence guarantees. Despite favorable theoretical properties, SKM-type methods have not been integrated to neural network constraint satisfaction problems.

2.4 Positioning of Our Contributions

Existing methods have various limitations when handling input-dependent dynamic linear constraints: soft constraints cannot strictly guarantee constraint satisfaction; differentiable optimization layers and feasibility-seeking architectures (e.g., GLinSAT (Zeng et al. 2024), homeomorphic projection (Liang, Chen, and Low 2023), FSNet (Nguyen and Donti 2025), FRMNet (Han et al. 2024), DC3 (Donti, Rolnick, and Kolter 2021)) often require solving auxiliary optimization problems or running invertible-network subroutines in each forward pass; parameterized feasible space methods require prior knowledge or offline computation of feasible points, making them difficult to handle dynamic constraints; decision rule methods have limited expressiveness and typically require convex constraint sets.

The T-SKM-Net framework proposed in this work first introduces SKM methods to the neural network constraint satisfaction domain, leveraging their computational efficiency and theoretical convergence guarantees. Unlike existing methods, the T-SKM-Net framework provides two flexible usage modes:

1. **Post-processing Method**: Used solely as a post-processing method without incorporation into training steps. It can be directly applied to any pre-trained neural network to ensure satisfaction of linear constraints.
2. **Joint Training Mode**: Integrates the constraint satisfaction layer into the neural network framework, achieving end-to-end joint optimization that optimizes prediction performance while satisfying constraints.

3 Preliminaries

3.1 Problem Formulation

Consider the mixed linear constraint system:

$$A(x)z \leq b(x) \tag{1}$$
$$C(x)z = d(x) \tag{2}$$

where $x \in \mathbb{R}^{n_{\text{in}}}$ is the input, $z \in \mathbb{R}^n$ is the output variable, $A(x) \in \mathbb{R}^{p \times n}$, $C(x) \in \mathbb{R}^{q \times n}$ are constraint matrices, and $b(x) \in \mathbb{R}^p$, $d(x) \in \mathbb{R}^q$ are right-hand side vectors. The feasible region is:

$$\mathcal{F}(x) = \{z \in \mathbb{R}^n : A(x)z \leq b(x), C(x)z = d(x)\} \tag{3}$$

In neural network constraint satisfaction, an upstream network $f_\theta : \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}^n$ produces output $y_0 = f_\theta(x)$ that typically violates constraints ($y_0 \notin \mathcal{F}(x)$). A constraint satisfaction layer transforms y_0 into a feasible solution $z^* \in \mathcal{F}(x)$. The layer must satisfy: (1) constraint satisfaction, (2) solution quality, (3) computational efficiency, and (4) end-to-end trainability.

3.2 Sampling Kaczmarz-Motzkin Method

The SKM algorithm solves linear inequality systems $Az \leq b$ through the following iteration:

$$z_{k+1} = z_k - \delta \frac{\left(a_{i^*}^\top z_k - b_{i^*}\right)_+}{\|a_{i^*}\|^2} a_{i^*} \tag{4}$$

where $\delta > 0$ is the step size, $i^* = \arg\max_{i \in S_k} \left(a_i^\top z_k - b_i\right)_+$ is the most violated constraint in the sampled set $S_k \subseteq \{1, \ldots, p\}$ with $|S_k| = \beta$, $(\cdot)_+ = \max(\cdot, 0)$, and a_i is the i-th row of matrix A.

4 T-SKM-Net Framework

4.1 Framework Overview

T-SKM-Net (Trainable Sampling Kaczmarz-Motzkin Network) addresses the fundamental challenge of efficiently handling mixed linear constraint systems in neural networks. While SKM methods excel at solving pure inequality systems, directly applying them to mixed constraints faces significant geometric challenges: equality constraints define hyperplanes (zero-measure sets) while inequality constraints define half-spaces, creating a fundamental mismatch that leads to inefficient oscillations between constraint types during iterative processing.

To address this challenge, T-SKM-Net transforms mixed constraint problems into pure inequality problems before applying SKM iterations. The framework accepts upstream neural network output $y_0 \in \mathbb{R}^n$ and input $x \in \mathbb{R}^{n_{\text{in}}}$, producing output $z^* \in \mathcal{F}(x)$ that strictly satisfies all constraints through three key steps (Figure 1):

(1) **Constraint Transformation**: Convert mixed constraints into pure inequalities through SVD-based null space transformation (Golub and Van Loan 2013), which addresses the geometric mismatch by decomposing the problem into equality-satisfying subspace and inequality optimization; (2) **SKM Iteration**: Apply randomized constraint selection and projection in the transformed space, where iterations effectively move within the equality constraint manifold while seeking inequality constraint satisfaction; (3) **Solution Recovery**: Map the solution back to the original constraint space while preserving both equality and inequality constraint satisfaction.

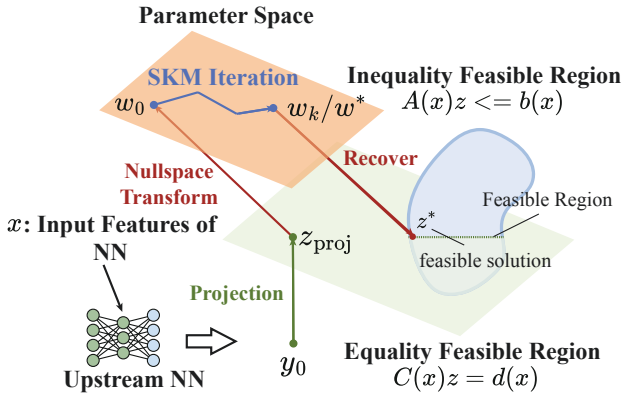

Figure 1: T-SKM-Net framework architecture.

4.2 Algorithm Pipeline

Algorithm 1: T-SKM-Net Constraint Satisfaction

Require: Upstream output y_0, constraints $A(x), b(x), C(x), d(x)$
Ensure: Feasible solution z^*

Constraint Transformation:
Compute SVD: $C = U\Sigma V^T$, obtain N (null space basis)
$z_{\text{proj}} = y_0 - C^\dagger(Cy_0 - d)$
$A_{\text{new}} = AN$, $b_{\text{new}} = b - Az_{\text{proj}}$
SKM Iteration:
Initialize $w_0 = 0$
for $k = 0, 1, \ldots, K-1$ **do**
  Sample constraint set $S_k \subseteq \{1, \ldots, p\}$ with $|S_k| = \beta$
  $i^* = \arg\max_{i \in S_k} (a_i^T w_k - b_i^{\text{new}})_+$
  $w_{k+1} = w_k - \delta \frac{(a_{i^*}^T w_k - b_{i^*}^{\text{new}})_+}{\|a_{i^*}\|^2} a_{i^*}$
  if termination condition met **then**
    break
  end if
end for
Solution Recovery:
$z^* = z_{\text{proj}} + Nw_K$
return z^*

The algorithm ensures constraint satisfaction: equality constraints are satisfied by construction since $Nw_K \in \text{null}(C)$, while inequality constraints are satisfied when SKM converges.

4.3 Theoretical Guarantees

We provide three theoretical results establishing the effectiveness and trainability of T-SKM-Net:

Theorem 1 (SKM L2 Projection Approximation)**.** *For inequality constraint $Az \leq b$ with feasible region $\mathcal{P}$, the SKM method starting from z_0 satisfies:*

$$\mathbb{E}[d(z_k, z_0)] \leq 2 \cdot d(z_0, \mathcal{P}) \tag{5}$$

where $d(z_0, \mathcal{P})$ is the distance from initial point to feasible region.

Proof Sketch: We construct an auxiliary function $V(z) = \|z - P(z_0)\|^2$ where $P(z_0)$ is the L2 projection onto $\mathcal{P}$. The key insight is that SKM updates exhibit Fejér monotonicity (Combettes 2008): $V(z_{k+1}) \leq V(z_k)$ for $0 < \delta < 2$. Applying Jensen's inequality and the triangle inequality yields the bound. *Complete proof in appendix.*

Theorem 2 (T-SKM-Net L2 Projection Approximation)**.** *For mixed constraint system with feasible region $\mathcal{F}(x)$ and null space basis matrix N, T-SKM-Net satisfies:*

$$\mathbb{E}[d(z_k, y_0)] \leq \sqrt{1 + 4\kappa(N)^2} \cdot d(y_0, P(y_0)) \tag{6}$$

where $\kappa(N)$ is the condition number of N and $P(y_0)$ is the L2 projection. For SVD-based N, $\kappa(N) = 1$, yielding $\mathbb{E}[d(z_k, y_0)] \leq \sqrt{5} \cdot d(y_0, P(y_0))$.

Proof Sketch: The framework decomposes the original problem through null space transformation: $z = z_{\text{proj}} + Nw$. We exploit orthogonality between the projection error $e = y_0 - z_{\text{proj}}$ and the null space component Nw. Combined with Theorem 1 applied to the transformed subproblem and matrix singular value bounds, the Cauchy-Schwarz inequality yields the final bound where the condition number $\kappa(N)$ controls the approximation quality. *Complete proof in appendix.*

Assumption 1 (Regularity Conditions)**.** *The parameterized SKM algorithm with constraints $A(x)w \leq b(x)$ satisfies:*

1. ***Non-degeneracy**: $\|a_i(x)\| \geq c > 0$ for all i, x*
2. ***Differentiability**: $A(x), b(x)$ are differentiable with $\|\nabla_x A(x)\| \leq L_A$, $\|\nabla_x b(x)\| \leq L_b$*
3. ***Boundedness**: $\|A(x)\|, \|b(x)\| \leq M$ for all relevant x*
4. ***Step size**: $0 < \delta < 2$*
5. ***Non-degeneracy of tie events**: Constraint violation differences do not vanish identically on open subsets of input space*

Let W_K be a random function mapping x to the result after K steps of random SKM iterate with sampling path ω.

Theorem 3 (End-to-End Trainability)**.** *Under Assumption 1, the SKM algorithm supports end-to-end training with: (i) well-defined expected gradients $\nabla_x \mathbb{E}_\omega[W_K(x, \omega)]$, (ii) finite gradient variance Var$(\nabla_x W_K(x, \omega)) < \infty$, and (iii) unbiased gradient estimation $\mathbb{E}_\omega[\nabla_x W_K(x, \omega)] = \nabla_x \mathbb{E}_\omega[W_K(x, \omega)]$.*

Proof Sketch: The main challenge is handling non-differentiable $\arg\max$ operations in SKM. We establish that tie events (where multiple constraints achieve the maximum violation) have zero Lebesgue measure under Assumption 1(5). On the complement, SKM is piecewise differentiable with bounded Jacobians. The gradient bound $\|\nabla_x W_k\| \leq kC$ ensures dominated convergence, enabling interchange of expectation and differentiation. Since sampling probabilities are independent of x, gradients are unbiased. *Complete proof in appendix.*

These results establish that T-SKM-Net produces high-quality solutions in expectation (Theorems 1 and 2) while supporting standard backpropagation despite non-differentiable operations (Theorem 3).

4.4 Usage Modes

T-SKM-Net supports two flexible usage modes:

Post-processing Mode: Applied to pre-trained networks without modifying parameters. Given network f_θ and input x, compute $z^* = \mathcal{T}\text{-SKM}(f_\theta(x), x, \omega)$, ensuring constraint satisfaction while preserving network accuracy.

End-to-End Training Mode: Integrated into network architecture for joint optimization. The training objective becomes:

$$\min_\theta \mathbb{E}[\mathcal{L}(\mathcal{T}\text{-SKM}(f_\theta(x), x, \omega), y_{\text{true}})] \tag{7}$$

Despite non-differentiable $\arg\max$ operations, Theorem 3 guarantees unbiased gradient estimation by treating the sampling sequence ω as a deterministic path during backpropagation.

4.5 Implementation and Strategy

Computational Optimization **Batch tensorization**: T-SKM-Net is implemented as a batched tensor operator, so all steps (null space transform, SKM iteration, recovery) can be easily parallelized on GPUs for large-scale problems.

SVD precomputation: In applications where the equality matrix C is fixed (e.g., DCOPF with fixed topology), we precompute $C = U\Sigma V^\top$ once offline and reuse the null-space basis N, which removes this cost from the online path.

Parameter Selection Guidelines **Step size and sampling**: In all experiments we set $\delta = 1.0$ and observe a good trade-off between convergence speed and projection accuracy. For the sampling size β, we use small fixed values (e.g., 5–10) on small problems and scale it roughly like $\beta = \mathcal{O}(\sqrt{m})$ on large problems, where m is the number of inequality constraints.

Training Strategy Optimization **Violation-aware loss**: For joint training, we add simple penalty terms on the equality/inequality violations of $f_\theta(x)$ to the task loss, which empirically shortens SKM convergence and stabilizes training.

Delayed activation: We often freeze the T-SKM-Net layer in the early epochs and enable it only in the last 10%–20% of training, letting the backbone first learn a reasonable initialization before enforcing hard constraints.

Algorithm Variant Support T-SKM-Net is compatible with SKM variants (e.g., momentum-accelerated updates).

5 Experiments

We evaluate T-SKM-Net on synthetic constraint satisfaction problems and real-world DC Optimal Power Flow (DCOPF) applications. All experiments are conducted on Intel Core i5-13500 CPU and AMD Radeon Instinct MI50 GPU (if not

Dim	Method	Time (s)	Speedup	Max Vio.	Rel. Err.
500	CVXPY	0.164	1.0×	1.78e-14	-
	OSQP	0.435	0.38×	9.18e-06	0.74
	Gurobi	0.073	2.24×	1.69e-14	0.0
	T-SKM-Net	**0.017**	**9.65**×	4.30e-06	12.1
2000	CVXPY	10.03	1.0×	4.62e-14	-
	OSQP	29.80	0.34×	5.90e-05	0.36
	Gurobi	1.658	6.05×	3.20e-14	0.0
	T-SKM-Net	**0.237**	**42.3**×	1.19e-06	7.53
3000	CVXPY	32.59	1.0×	5.51e-14	-
	OSQP	96.24	0.34×	1.13e-04	0.44
	Gurobi	4.217	7.73×	6.04e-14	0.0
	T-SKM-Net	**0.856**	**38.1**×	6.20e-07	6.19

Table 1: L2 Projection Performance (Medium Scale, Speedup vs CVXPY)

Dim	Method	Time (s)	Speedup	Max Vio.
4000	Gurobi	9.184	1.0×	7.82e-14
	T-SKM-Net	**2.001**	**4.59**×	8.71e-06
8000	Gurobi	60.76	1.0×	1.07e-13
	T-SKM-Net	**13.30**	**4.57**×	3.41e-07
10000	Gurobi	103.6	1.0×	1.72e-13
	T-SKM-Net	**22.82**	**4.54**×	1.31e-06

Table 2: L2 Projection Performance (Large Scale, Speedup vs Gurobi)

specified). We compare against state-of-the-art optimization solvers and neural constraint satisfaction methods, focusing on computational efficiency, solution quality, and constraint satisfaction.

5.1 L2 Projection Efficiency

We first evaluate T-SKM-Net as a post-processing method for L2 projection onto feasible regions. We construct random linear constraint systems with equal numbers of equality and inequality constraints (both $n/2$ where n is the variable dimension), starting from infeasible points with initial maximum violations on the order of 10^2.

We compare against three optimization approaches: CVXPY (using OSQP algorithm (Stellato et al. 2020)), OSQP library directly, and Gurobi commercial solver. Tables 1 and 2 show results where "Speedup" indicates acceleration relative to the baseline method, "Max Vio." denotes maximum constraint violation, and "Rel. Err." measures the L2 distance to CVXPY's solution.

T-SKM-Net achieves significant speedups while maintaining approximation errors orders of magnitude smaller than initial violations. For large-scale problems where open-source solvers become impractical, T-SKM-Net consistently outperforms the commercial Gurobi solver with 4-5× speedups.

5.2 Constrained Optimization Problems

We evaluate T-SKM-Net on constrained quadratic programming problems to assess both post-processing effectiveness and end-to-end trainability. We consider the optimization problem (Nocedal and Wright 2006):

$$\arg\min_{y} \quad \frac{1}{2} y^T Q y + p^T y \tag{8}$$

$$\text{s.t.} \quad Ay = x \tag{9}$$

$$Gy \leq h \tag{10}$$

where $y \in \mathbb{R}^{100}$ is the decision variable, $x \in \mathbb{R}^{50}$ is the input parameter, and we vary the number of inequality constraints from 10 to 150.

We compare T-SKM-Net against optimization solvers (OSQP, qpth (Amos and Kolter 2017)), differentiable optimization methods (OptNet), and neural constraint satisfaction methods (DC3, gradient-based post-processing). Neural networks are trained using gradient descent with constraint violation penalties in the loss function to approximate the optimization problem solutions. The neural network baseline uses gradient-based correction at test time to reduce constraint violations (Donti, Rolnick, and Kolter 2021). We evaluate both end-to-end training and post-processing modes of T-SKM-Net.

Table 3 shows results as the number of inequality constraints ("Ineq.") varies from 30 to 150, where "Obj. Val." denotes the objective value, "Max Eq./Ineq." the maximum equality/inequality violations, and † indicates constraint violations.

The results reveal distinct performance characteristics across different approaches. Traditional optimization solvers (OSQP, qpth) provide exact solutions but with very different costs: qpth's differentiable implementation incurs substantial overhead (86–652ms). OptNet, which uses qpth as its underlying differentiable optimizer, maintains strict constraint satisfaction, but its high cost makes it impractical for real-time applications.

Among neural constraint satisfaction methods, T-SKM-Net consistently achieves zero constraint violations in both training and post-processing modes, while DC3 exhibits inequality violations in high-constraint scenarios. The neural network baseline with gradient-based correction, despite achieving competitive objective values, shows persistent equality violations, rendering it unsuitable for safety-critical applications.

In terms of solution quality, T-SKM-Net is competitive across all constraint densities. For moderate problems (30–50 inequalities), our method attains objective values comparable to NN and OptNet while maintaining zero violations. In high-constraint scenarios (130–150 inequalities), T-SKM-Net matches OptNet's solution quality while achieving over 200× speedup, and clearly outperforms DC3, which suffers from both constraint violations and deteriorated objectives.

The computational efficiency analysis shows that T-SKM-Net achieves large speedups compared to differentiable optimization methods: up to 450× faster than OptNet and

Ineq.	Method	Obj. Val.	Max Eq.†	Max Ineq.†	Time (ms)
30	OSQP	-16.33	0.000	0.000	**0.3**
	qpth	-16.33	0.000	0.000	86.4
	OptNet	**-14.03**	0.000	0.000	71.5
	DC3	**-14.02**	0.000	0.000	3.3
	NN, $\leq$ test	**-14.09**	0.350†	0.000	2.5
	Ours (Train)	**-14.04**	0.000	0.000	1.0
	Ours (Post)	**-14.05**	0.000	0.000	1.0
50	OSQP	-15.05	0.000	0.001†	**0.8**
	qpth	-15.05	0.000	0.000	106.3
	OptNet	-12.46	0.000	0.000	101.0
	DC3	**-13.44**	0.000	0.000	5.7
	NN, $\leq$ test	-12.55	0.351†	0.000	2.5
	Ours (Train)	-12.53	0.000	0.000	**0.9**
	Ours (Post)	-12.52	0.000	0.000	**0.9**
130	OSQP	-12.73	0.000	0.000	**1.1**
	qpth	-12.73	0.000	0.000	492.8
	OptNet	**-10.25**	0.000	0.000	450.6
	DC3	-9.45	0.000	0.002†	109.0
	NN, $\leq$ test	**-10.32**	0.352†	0.000	2.6
	Ours (Train)	**-10.23**	0.000	0.000	**1.0**
	Ours (Post)	**-10.24**	0.000	0.000	2.1
150	OSQP	-12.67	0.000	0.001†	**1.2**
	qpth	-12.67	0.000	0.000	652.1
	OptNet	**-10.22**	0.000	0.000	560.0
	DC3	-9.06	0.000	0.006†	111.2
	NN, $\leq$ test	**-10.29**	0.351†	0.000	2.5
	Ours (Train)	**-10.24**	0.000	0.000	1.8
	Ours (Post)	**-10.24**	0.000	0.000	**1.4**

Table 3: Constrained QP Performance (100 variables, 50 equality constraints). † indicates constraint violations.

109× faster than DC3's reported times, while keeping runtime comparable to the highly optimized OSQP solver. Most importantly, unlike baseline neural networks that sacrifice constraint satisfaction for speed, T-SKM-Net enforces strict feasibility without compromising computational efficiency. These results demonstrate T-SKM-Net's effectiveness in both usage modes: as a post-processing method, it transforms infeasible neural network outputs into high-quality feasible solutions; in end-to-end training, it enables joint optimization of prediction accuracy and constraint satisfaction, making it particularly suitable for real-time constrained optimization.

5.3 DC Optimal Power Flow

We evaluate T-SKM-Net on the IEEE 118-bus DC Optimal Power Flow (DCOPF) problem (Hou et al. 2018; Anderson et al. 2022), a critical real-world application in power system operations. This experiment is run on NVIDIA RTX 4090. DCOPF involves minimizing generation costs while satisfying power balance equations and transmission line capacity constraints.

This problem represents a mixed linear constraint system that is well-suited for evaluating our constraint satisfaction approach. The DCOPF problem can be formulated as:

$$\begin{aligned} \min_{P_G,\,\theta} \quad & \sum_{g\in G}\Big(\tfrac{1}{2}\,c_g\,P_{G,g}^2 + b_g\,P_{G,g}\Big) \qquad (11)\\ \text{s.t.} \quad & P_{G,i} - P_{D,i} = \sum_{j\in\mathcal{N}(i)} B_{ij}\,(\theta_i-\theta_j), \quad \forall\, i\in\mathcal{B},\\ & P_{G,i}^{\min} \le P_{G,i} \le P_{G,i}^{\max}, \quad \forall\, i\in G,\\ & -P_{ij}^{\max} \le B_{ij}\,(\theta_i-\theta_j) \le P_{ij}^{\max}, \quad \forall\,(i,j)\in\mathcal{L}. \end{aligned}$$

where G, $\mathcal{B}$, and $\mathcal{L}$ denote the sets of generators, buses, and transmission lines, respectively, B_{ij} are line susceptances, and the input load demands P_D determine the optimal generation dispatch P_G and voltage angles θ.

We evaluate T-SKM-Net in post-processing mode, joint-training mode, and joint-training mode with MSE loss punishment to the upstream output to verify constraint satisfaction for neural network predictions and compare speed and solution quality under different setups.

We compare against the pandapower (Thurner et al. 2018) optimization solver as the ground truth baseline, a neural network baseline without constraint satisfaction, and DC3 (Donti, Rolnick, and Kolter 2021). All experiments use single-instance processing (batch size = 1) to reflect real-time operational requirements. Our method under all 3 setups achieves zero constraint violations under 10^{-3} tolerance while significantly outperforming existing approaches in both speed and solution quality.

Method	Time (ms)	Opt. Gap (%)	Max Eq. (e-4)	Max Ineq. (e-4)
pandapower	141.08	0.00 (0.00)	0	0
NN	**0.29**	4e-4 (3.5e-2)	2997†	6†
DC3	52.15	5.6e-5 (2.6e-3)	0	4†
Ours (Post)	4.27	1.0e-4 (2.5e-3)	**0**	**0**
Ours (Train)	5.64	7.0e-5 (2.0e-3)	**0**	**0**
Ours (Train-p)	5.25	**5.4e-5 (8.0e-4)**	**0**	**0**

Table 4: DCOPF Performance on IEEE 118-bus System. † indicates constraint violations.

Table 4 shows that T-SKM-Net achieves remarkable computational efficiency, delivering over 25× speedup compared to the traditional pandapower solver and about 10× speedup compared to DC3, while maintaining superior solution quality. The order-of-magnitude difference in computational time (∼5ms vs 52.15ms) demonstrates the significant efficiency advantage of our approach. The neural network baseline, while fastest, exhibits obvious constraint violations, making it unsuitable for practical deployment. DC3 achieves equality constraint satisfaction but violates inequality constraints. Most importantly, our method under all 3 setups ensures strict constraint satisfaction with zero violations while maintaining competitive computational efficiency.

Method	Sharpe	Avg Ineq/Eq (10^{-2})	Proj. time (s)
NN (no proj)	2.16	49.6 / 0.0	3.0e-4
qpth	2.42	0.0 / 0.2	1.41
DC3	2.97	8.4 / 0.0	1.1e-1
GLinSAT	2.28	**0.0 / 0.0**	2.6e-1
Ours (Post)	2.93	**0.0 / 0.0**	2.5e-1
Ours (Train)	**3.02**	**0.0 / 0.0**	2.5e-1

Table 5: Predictive portfolio allocation on the S&P 500 dataset. "Avg Ineq/Eq" reports mean absolute inequality/equality violations. All methods share the same backbone; only the constraint satisfaction layer and training mode differ.

The results demonstrate T-SKM-Net's practical viability for real-time power system applications, where both computational speed and constraint satisfaction are critical requirements. The sub-10ms inference time enables rapid decision-making in dynamic grid operations, while zero constraint violations ensure system safety and operational feasibility.

5.4 Predictive Portfolio Allocation

To further evaluate the generality of T-SKM-Net beyond synthetic and power system benchmarks, we consider the predictive portfolio allocation task which maximizes the Sharpe ratio (Sharpe 1998). Denoting x_i as the predicted portfolio decision variable of asset i, the portfolio is required to satisfy linear constraints

$$\sum_{i=1}^{n} x_i = 1, \quad \sum_{i \in S} x_i \geq q, \quad 0 \leq x_i \leq 1,$$

where S is a set of preferred large-cap technology stocks (e.g., AAPL, MSFT, AMZN, TSLA, GOOGL) and $q = 0.5$, as in (Zeng et al. 2024). Performance is measured by the average Sharpe ratio over the test horizon.

We compare six methods that share the same network backbone and differ only in the constraint satisfaction layer and training strategy: NN (unconstrained network without any projection layer), qpth (Amos and Kolter 2017) (differentiable QP layer), DC3 (Donti, Rolnick, and Kolter 2021), GLinSAT (Zeng et al. 2024), Our method with both post-processing and joint-training mode. For all methods we use the same training and evaluation protocol as in (Zeng et al. 2024). We report the average Sharpe ratio, the mean absolute inequality/equality violations ("Avg Ineq/Eq"), and the average projection time per test epoch.

Table 5 shows that the unconstrained NN baseline attains a moderate Sharpe ratio but exhibits large inequality violations, rendering many portfolios infeasible. qpth and DC3 significantly reduce one type of violation but still leave a non-negligible number of infeasible portfolios. GLinSAT achieves zero infeasibility under a 1e-3 tolerance, but at the cost of a noticeably lower Sharpe ratio. In contrast, T-SKM-Net attains both strict feasibility and superior portfolio performance: even in post-processing mode it matches DC3 in Sharpe ratio while eliminating all infeasible portfolios, and in joint training mode it further improves the Sharpe ratio to 3.02 with essentially zero average violations, at a projection cost comparable to GLinSAT and much lower than qpth. This experiment demonstrates that T-SKM-Net provides state-of-the-art performance on this task.

5.5 Ablation Study

Due to space limitations, we defer detailed ablation studies on null space transformation, sampling strategies, and SKM variants to the technical appendix, where we provide additional efficiency and accuracy comparisons.

6 Conclusion

This work presents T-SKM-Net, the first neural network framework to leverage Sampling Kaczmarz-Motzkin methods for linear constraint satisfaction. The proposed framework addresses the fundamental challenge of ensuring strict constraint satisfaction in neural networks while maintaining computational efficiency and end-to-end trainability.

The key contributions include: (1) integrating SKM methods to neural network constraint satisfaction for handling input-dependent dynamic linear constraints; (2) theoretical guarantees demonstrating L2 projection approximation in expectation and end-to-end trainability despite non-differentiable argmax operations; and (3) a flexible framework supporting both post-processing and joint training modes.

Experimental validation demonstrates significant computational advantages: 4-5$\times$ speedups compared to commercial Gurobi solver on L2 projection tasks, over 25$\times$ acceleration compared to pandapower solver on DC Optimal Power Flow with zero constraint violations, and superior performance compared to existing neural constraint satisfaction methods.

Several limitations warrant future investigation. The framework is currently restricted to linear constraints, and the selection of sampling size β lacks theoretical guidance, relying on empirical rules. Additionally, backpropagation memory consumption scales linearly with iteration count. Future research directions include developing theoretical frameworks for optimal β selection, exploring nonlinear constraint extensions (requiring fundamentally different approaches), and investigating memory-efficient optimization techniques such as implicit differentiation or selective computational graph preservation.

The T-SKM-Net framework demonstrates that classical iterative methods can be effectively integrated into modern deep learning pipelines, opening new possibilities for constraint-aware neural network design in safety-critical applications.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (No. 2024YFB2409300).

References

Amos, B.; and Kolter, J. Z. 2017. OptNet: Differentiable Optimization as a Layer in Neural Networks. In Precup, D.; and

Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, 136–145. PMLR.

Anderson, A. A.; Kincic, S.; Jefferson, B. A.; Mcgary, B. J.; Fallon, C. K.; Ciesielski, D. K.; Wenskovitch, J. E.; and Chen, Y. 2022. A real-time operations manual for the IEEE 118 bus transmission model. Technical report, Pacific Northwest National Laboratory (PNNL), Richland, WA (United States).

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*, 177–186. Springer.

Butler, A.; and Kwon, R. H. 2023. Efficient Differentiable Quadratic Programming Layers: An ADMM Approach. *Computational Optimization and Applications*, 84(2): 449–476.

Cappart, Q.; Chételat, D.; Khalil, E. B.; Lodi, A.; Morris, C.; and Veličković, P. 2023. Combinatorial Optimization and Reasoning with Graph Neural Networks. *Journal of Machine Learning Research*, 24(130): 1–61.

Carpentier, J. 1962. Contribution a l'etude du dispatching economique. *Bull. Soc. Fr. Elec. Ser.*, 3: 431.

Chen, B.; Donti, P. L.; Baker, K.; Kolter, J. Z.; and Bergés, M. 2021. Enforcing Policy Feasibility Constraints through Differentiable Projection for Energy Optimization. In *Proceedings of the Twelfth ACM International Conference on Future Energy Systems*, 199–210. Virtual Event Italy: ACM. ISBN 978-1-4503-8333-2.

Combettes, P. L. 2008. Fejér monotonicity in convex optimization. In *Encyclopedia of optimization*, 1016–1024. Springer.

Constante-Flores, G. E.; Chen, H.; and Li, C. 2025. Enforcing Hard Linear Constraints in Deep Learning Models with Decision Rules. arXiv:2505.13858.

Cybenko, G. 1989. Approximation by Superpositions of a Sigmoidal Function. *Mathematics of Control, Signals, and Systems*, 2(4): 303–314.

Donti, P. L.; Rolnick, D.; and Kolter, J. Z. 2021. DC3: A Learning Method for Optimization with Hard Constraints. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Goda, T.; and Kitade, W. 2023. Constructing unbiased gradient estimators with finite variance for conditional stochastic optimization. *Mathematics and Computers in Simulation*, 204: 743–763.

Golub, G. H.; and Van Loan, C. F. 2013. *Matrix computations*. JHU press.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*. MIT press Cambridge.

Han, J.; Wang, W.; Yang, C.; Niu, M.; Yang, C.; Yan, L.; and Li, Z. 2024. FRMNet: A Feasibility Restoration Mapping Deep Neural Network for AC Optimal Power Flow. *IEEE Transactions on Power Systems*, 39(5): 6566–6577.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks*, 2(5): 359–366.

Hou, Q.; Yang, J.; Su, Y.; Wang, X.; and Deng, Y. 2023. Generalize Learned Heuristics to Solve Large-Scale Vehicle Routing Problems in Real-Time. In *The Eleventh International Conference on Learning Representations*.

Hou, Q.; Zhang, N.; Yang, J.; Kang, C.; Xia, Q.; and Miao, M. 2018. Linearized Model for Active and Reactive LMP Considering Bus Voltage Constraints. In *2018 IEEE Power & Energy Society General Meeting (PESGM)*, 1–5. Portland, OR: IEEE. ISBN 978-1-5386-7703-2.

Karczmarz, S. 1937. Angenaherte auflosung von systemen linearer glei-chungen. *Bull. Int. Acad. Pol. Sic. Let., Cl. Sci. Math. Nat.*, 355–357.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep Learning. *Nature*, 521(7553): 436–444.

Liang, E.; Chen, M.; and Low, S. H. 2023. Low Complexity Homeomorphic Projection to Ensure Neural-Network Solution Feasibility for Optimization over (Non-)Convex Set. In Krause, A.; Brunskill, E.; Cho, K.; Engelhardt, B.; Sabato, S.; and Scarlett, J., eds., *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, 20623–20649. PMLR.

Liu, X.; Lu, Y.; Abbasi, A.; Li, M.; Mohammadi, J.; and Kolouri, S. 2024. Teaching networks to solve optimization problems. *IEEE Access*, 12: 17102–17113.

Loera, J. A. D.; Haddock, J.; and Needell, D. 2017. A Sampling Kaczmarz-Motzkin Algorithm for Linear Feasibility. *SIAM J. Sci. Comput.*, 39(5).

Min, Y.; and Azizan, N. 2025. HardNet: Hard-Constrained Neural Networks with Universal Approximation Guarantees. arXiv:2410.10807.

Morshed, M. S.; Islam, M. S.; and Noor-E-Alam, M. 2022. Sampling Kaczmarz Motzkin Method for Linear Feasibility Problems: Generalization & Acceleration. *Mathematical Programming*, 194(1-2): 719–779.

Motzkin, T. S.; and Schoenberg, I. J. 1954. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6: 393–404.

Nguyen, H. T.; and Donti, P. L. 2025. FSNet: Feasibility-seeking Neural Network for Constrained Optimization with Guarantees. *arXiv preprint arXiv:2506.00362*.

Nocedal, J.; and Wright, S. J. 2006. *Numerical optimization*. Springer.

Raissi, M.; Perdikaris, P.; and Karniadakis, G. 2019. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *Journal of Computational Physics*, 378: 686–707.

Robbins, H.; and Monro, S. 1951. A stochastic approximation method. *The annals of mathematical statistics*, 400–407.

Scutari, G.; and Sun, Y. 2018. Parallel and distributed successive convex approximation methods for big-data optimization. In *Multi-Agent Optimization: Cetraro, Italy 2014*, 141–308. Springer.

Sharpe, W. F. 1998. The Sharpe Ratio (Fall 1994). In Bernstein, P. L.; and Fabozzi, F. J., eds., *Streetwise*, 169–178. Princeton University Press. ISBN 978-1-4008-2940-8.

Smith, K. A. 1999. Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research. *INFORMS Journal on Computing*, 11(1): 15–34.

Stellato, B.; Banjac, G.; Goulart, P.; Bemporad, A.; and Boyd, S. 2020. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4): 637–672.

Strohmer, T.; and Vershynin, R. 2009. A Randomized Kaczmarz Algorithm with Exponential Convergence. *Journal of Fourier Analysis and Applications*, 15(2): 262–278.

Thurner, L.; Scheidler, A.; Schafer, F.; Menke, J.-H.; Dollichon, J.; Meier, F.; Meinecke, S.; and Braun, M. 2018. Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems. *IEEE Transactions on Power Systems*, 33(6): 6510–6521.

Tordesillas, J.; How, J. P.; and Hutter, M. 2023. RAYEN: Imposition of Hard Convex Constraints on Neural Networks. arXiv:2307.08336.

Yang, L.; Zhang, D.; and Karniadakis, G. E. 2020. Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations. *SIAM Journal on Scientific Computing*, 42(1): A292–A317.

Zeng, H.; Yang, C.; Zhou, Y.; Yang, C.; and Guo, Q. 2024. GLinSAT: The General Linear Satisfiability Neural Network Layer By Accelerated Gradient Descent. In Globersons, A.; Mackey, L.; Belgrave, D.; Fan, A.; Paquet, U.; Tomczak, J. M.; and Zhang, C., eds., *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*. arXiv.

Zhang, L.; Tabas, D.; and Zhang, B. 2024. An Efficient Learning-Based Solver for Two-Stage DC Optimal Power Flow with Feasibility Guarantees. arXiv:2304.01409.

Zhao, T.; Pan, X.; Chen, M.; and Low, S. H. 2023. Ensuring DNN Solution Feasibility for Optimization Problems with Linear Constraints. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.