Cauchy–Schwarz Regularizers

Anonymous authors
Paper under double-blind review

Abstract

We introduce a novel class of regularization functions, called Cauchy–Schwarz (CS) regularizers, which can be designed to induce a wide range of properties in solution vectors of optimization problems. To demonstrate the versatility of CS regularizers, we derive concrete regularization functions that promote discrete-valued vectors, eigenvectors of a given matrix, and orthogonal matrices. The resulting CS regularizers are simple, differentiable, and can be free of spurious critical points, making them suitable for gradient-based solvers and large-scale optimization problems. In addition, CS regularizers automatically adapt to the appropriate scale, which is, for example, beneficial when discretizing weights of neural networks. To demonstrate the efficacy of CS regularizers, we provide results for solving underdetermined systems of linear equations and weight quantization in neural networks. Furthermore, we discuss specializations, variations, and generalizations, which lead to an even broader class of new and possibly more powerful regularizers.

1 Introduction

We focus on the design of novel regularization functions $\ell\colon \mathbb{R}^N \to \mathbb{R}_{\geq 0}$ that promote certain pre-defined properties on the solution vector(s) $\hat{\mathbf{x}} \in \mathbb{R}^N$ of regularized optimization problems

$$\hat{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) + \lambda\ell(\mathbf{x}), \tag{1}$$

where $f\colon \mathbb{R}^N \to \mathbb{R}$ is an objective function and $\lambda \in \mathbb{R}_{\geq 0}$ a regularization parameter. One instance of such an optimization problem is the binarization of neural-network weights, where the solution(s) of (1) are the network's weights that should be binary-valued $\hat{\mathbf{x}} \in \{-\alpha, \alpha\}^N$, but with appropriate scale $\alpha \in \mathbb{R}$ chosen by the regularizer—the scale can then be absorbed into the activation function.

1.1 Contributions

We propose *Cauchy–Schwarz (CS) regularizers*, a novel class of regularization functions that can be designed to impose a wide range of properties. We derive concrete examples of CS regularization functions that promote discrete-valued vectors (e.g., binary- and ternary-valued vectors), eigenvectors of a given matrix, and matrices with orthogonal columns. The resulting regularizers are (i) simple, (ii) automatically determine the appropriate scale, (iii) free of any spurious critical points, and (iv) differentiable, which enables the use of (stochastic) gradient-based numerical solvers that make them suitable to be used in large-scale optimization problems. In addition, we discuss a variety of specializations, variations, and generalizations, which allow for the design of an even broader class of new and possibly more powerful regularization functions. Finally, we showcase the efficacy and versatility of CS regularizers for solving underdetermined systems of linear equations and neural network weight binarization and ternarization. All proofs and additional experimental results are relegated to the appendices, which can be found in the supplementary material.

1.2 Notation

Column vectors and matrices are written in boldface lowercase and uppercase letters, respectively. The entries of a vector $\mathbf{x} \in \mathbb{R}^N$ are $[\mathbf{x}]_n = x_n$, $n = 1, \ldots, N$, and transposition is $\mathbf{x}^{\mathrm{T}}$. The N-dimensional all-zeros vector is $\mathbf{0}_N$ and the all-ones vector $\mathbf{1}_N$; we omit the dimension N if it is clear from the context. The inner product between the vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$ is $\langle \mathbf{x}, \mathbf{y}\rangle = \mathbf{x}^{\mathrm{T}}\mathbf{y}$, and linear dependence is denoted by

$$\mathbf{x} \sim \mathbf{y} \iff \exists (a_1, a_2) \in \mathbb{R}^2 \setminus \{(0,0)\} : a_1 \mathbf{x} = a_2 \mathbf{y}. \tag{2}$$

For $p \geq 1$, the p-norm of a vector $\mathbf{x} \in \mathbb{R}^N$ is $\|\mathbf{x}\|_p \triangleq (\sum_{n=1}^{N} |x_n|^p)^{1/p}$, and we will frequently use the shorthand notation $[\![\mathbf{x}]\!]^p \triangleq \sum_{n=1}^{N} x_n^p$ (note the absence of absolute values). The entry on the nth row and kth column of a matrix $\mathbf{X}$ is $X_{n,k}$, the Frobenius norm is $\|\mathbf{X}\|_{\mathrm{F}}$, and columnwise vectorization is $\mathrm{vec}(\mathbf{X})$. The $N \times N$ identity matrix is $\mathbf{I}_N$ and the $M \times N$ all-zero matrix is $\mathbf{0}_{M \times N}$.

1.3 Relevant Prior Art

Semidefinite relaxation (SDR) can be used for solving optimization problems with binary-valued solutions (Luo et al., 2010). Since SDR requires lifting (i.e., increasing the dimension of the original problem size), solving such problems quickly results in prohibitive complexity, even for moderately-sized problems. As a remedy, non-lifting-based SDR approximations were proposed in (Shah et al., 2016; Castañeda et al., 2017). These methods utilize biconvex relaxation that scales better to larger optimization problems. Convex non-lifting-based approaches were also proposed for recovering binary-valued solutions from linear measurements using ℓ^∞-norm regularization (Mangasarian & Recht, 2011). In contrast to such methods, the proposed CS regularizers are (i) differentiable, which enables their use together with differentiable objective functions and any (stochastic) gradient-based numerical solver, and (ii) can be specialized to impose a wider range of different structures.

Vector discretization is widely used for neural network parameter quantization (Hubara et al., 2017). Regularization-free approaches, e.g., the method from Rastegari et al. (2016), perform neural network binarization by simply quantizing the weights and adapting their scale to their average absolute value. Approaches that utilize projections onto discrete sets within gradient-descent-based methods have been proposed in (Hou et al., 2016; Leng et al., 2018). In contrast, the proposed CS regularizers are differentiable and automatically adapt their scale to the magnitude of the solution vectors.

Prior art includes a plethora of references vector discretization methods that rely on regularization functions have been proposed: The methods in (Hung et al., 2015; Tang et al., 2017; Wess et al., 2018; Bai et al., 2019; Darabi et al., 2019; Choi et al., 2020; Yang et al., 2021; Razani et al., 2021; Xu et al., 2023) use regularization functions related[1] to the form $\ell(\mathbf{x}, \beta) = \sum_{n=1}^{N} (x_n^2 - \beta)^2$ for N-dimensional vectors $\mathbf{x} \in \mathbb{R}^N$ and either fix the magnitude β^2 (e.g., $\beta = 1$) or learn this additional parameter during gradient descent. Another strain regularization functions that utilizes trigonometric functions related[2] to the form $\ell(\mathbf{x}, \beta) = \sum_{n=1}^{N} \sin^2(\beta \pi x_n)$ have been proposed in (Naumov et al., 2018; Elthakeb et al., 2020; Solodskikh et al., 2022). In contrast to all of the above regularization functions, the proposed CS regularizers (i) do not introduce additional trainable parameters while still being able to automatically adapt their scale to the vectors' magnitude and (ii) can be designed to promote a wider range of structures. Finally, the proposed CS regularizers include the regularization functions of Tang et al. (2017); Darabi et al. (2019) as a special case; see App. B.4.

Recovering matrices with orthogonal columns finds, for example, use in the orthogonal Procrustes problem: $\hat{\mathbf{X}} = \arg\min_{\mathbf{X} \in \mathbb{R}^{N \times K}} \|\mathbf{A}\mathbf{X} - \mathbf{B}\|_{\mathrm{F}}$ subject to $\mathbf{X}^{\mathrm{T}}\mathbf{X} = \mathbf{I}_N$ A closed-form solution to this problem is given by $\hat{\mathbf{X}} = \mathbf{U}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are the left- and right-singular matrices of the matrix $\mathbf{A}^{\mathrm{T}}\mathbf{B}$ (Schönemann, 1966). In contrast, CS regularizers can be designed to promote matrices with orthogonal rows of arbitrary scale and without requiring a singular value decomposition.

We finally note that Tran et al. (2022) propose the use of the CS *divergence* to regularize autoencoders. In contrast, we use the CS *inequality* (Steele, 2004) to design new regularization functions that can—among many other structures—be used to promote discrete-valued vectors (e.g., binary or ternary), eigenvectors to a given matrix, and matrices with orthogonal rows.

[1] Some methods, e.g., (Darabi et al., 2019), use non-differentiable regularizers with $||x_n| - \beta|$ instead of $(x_n^2 - \beta)^2$, while others, e.g., (Xu et al., 2023), use regularizers of the form $\gamma\|\mathbf{x} - \alpha\,\mathrm{sign}(\mathbf{x})\|$ and introduce additional scaling factors.

[2] The method in (Naumov et al., 2018) fixes the scale, while Elthakeb et al. (2020) utilizes a trainable parameter; the regularizer in (Solodskikh et al., 2022) introduces an additional differentiable regularizer that imposes finite range.

2 CAUCHY–SCHWARZ REGULARIZERS

In this section, we introduce the general recipe for deriving CS regularizers. We then use this recipe to design specific CS regularization functions that promote discrete-valued (e.g., binary and ternary) vectors, eigenvectors of a given matrix, and matrices with orthogonal columns.

2.1 THE RECIPE

The following result is an immediate consequence of the CS inequality (Steele, 2004) and provides a recipe for the design of a wide range of regularization functions; a short proof is provided in App. A.1.

Proposition 1. *Fix two vector-valued functions* $\mathbf{g}, \mathbf{h} : \mathbb{R}^N \to \mathbb{R}^M$ *and define the set*

$$\mathcal{X} \triangleq \left\{ \tilde{\mathbf{x}} \in \mathbb{R}^N : \mathbf{g}(\tilde{\mathbf{x}}) \sim \mathbf{h}(\tilde{\mathbf{x}}) \right\}. \tag{3}$$

Then, the nonnegative regularization function

$$\ell(\mathbf{x}) \triangleq \|\mathbf{g}(\mathbf{x})\|_2^2 \|\mathbf{h}(\mathbf{x})\|_2^2 - |\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x}) \rangle|^2 \tag{4}$$

is zero if and only if (iff) $\mathbf{x} \in \mathcal{X}$.

We call regularization functions derived from Proposition 1 *CS regularizers*. While CS regularizers are guaranteed to be (i) nonnegative and (ii) zero iff $\mathbf{x} \in \mathcal{X}$, it is also desirable for gradient-based numerical solvers that these regularizers do not exhibit any *spurious critical points*.

Definition 1. *A* spurious critical point *is a vector* $\mathbf{x} \notin \mathcal{X}$ *for which* $\nabla \ell(\mathbf{x}) = \mathbf{0}$.

Whether or not a CS regularizer has spurious critical points depends on the specific choice of $\mathbf{g}$ and $\mathbf{h}$. Nonetheless, even if a CS regularizer has spurious critical points, it may still accomplish the desired goal. We conclude by noting that any vector $\mathbf{x}$ for which $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ or $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ will minimize (4).

The following result will be useful below when we analyze properties of specific CS regularizers; a short proof is given in App. A.2.

Lemma 1. *Fix two vector-valued functions* $\mathbf{g}, \mathbf{h} : \mathbb{R}^N \to \mathbb{R}^M$. *Then, the following equalities hold:*

$$\ell(\mathbf{x}) = \|\mathbf{g}(\mathbf{x})\|_2^2 \min_{\beta \in \mathbb{R}} \|\beta \mathbf{g}(\mathbf{x}) - \mathbf{h}(\mathbf{x})\|_2^2 = \|\mathbf{h}(\mathbf{x})\|_2^2 \min_{\beta \in \mathbb{R}} \|\mathbf{g}(\mathbf{x}) - \beta \mathbf{h}(\mathbf{x})\|_2^2. \tag{5}$$

Lemma 1 will be used to highlight the important *auto-scale property* CS regularizers, since setting β to its optimal value in (5) leads exactly to the regularization function in (4), as the optimization problem in β is continuous, quadratic, and has a closed-form solution.

2.2 RECOVERING DISCRETE-VALUED VECTORS

From Proposition 1, we can derive a range of *differentiable* CS regularizers that, when minimized, promote discrete-valued vectors. This can be accomplished by using entry-wise polynomials for the functions $\mathbf{g}$ and $\mathbf{h}$. We next show three concrete and practical useful examples.

2.2.1 SYMMETRIC BINARY

Define $\mathbf{g}(\mathbf{x}) \triangleq [x_1^2, \ldots, x_N^2]^{\mathrm{T}}$ and $\mathbf{h}(\mathbf{x}) \triangleq \mathbf{1}_N$. Then, Proposition 1 yields the following CS regularizer that promotes symmetric binary-valued vectors; see App. A.3 for the derivation.

Regularizer 1 (Symmetric Binary)**.** *Let* $\mathbf{x} \in \mathbb{R}^N$ *and define*

$$\ell_{\text{bin}}(\mathbf{x}) \triangleq N [\![\mathbf{x}]\!]^4 - \left([\![\mathbf{x}]\!]^2 \right)^2. \tag{6}$$

Then, the nonnegative function in (6) is only zero for symmetric binary-valued vectors, i.e., iff $\mathbf{x} \in \{-\alpha, \alpha\}^N$ *for any* $\alpha \in \mathbb{R}$. *Furthermore,* $\ell_{\text{bin}}(\mathbf{x})$ *does not have any spurious critical points.*

To gain insight into Regularizer 1, we invoke Lemma 1 and obtain

$$\ell_{\text{bin}}(\mathbf{x}) = N \min_{\beta \in \mathbb{R}_{\geq 0}} \textstyle\sum_{n=1}^N \left((x_n - \sqrt{\beta})(x_n + \sqrt{\beta}) \right)^2. \tag{7}$$

This equivalence implies that, for a given vector $\mathbf{x}$, Regularizer 1 is the right-hand-side total square error in (7), but with optimally chosen scale $\alpha \triangleq \sqrt{\beta}$; this is the *auto-scale property* of CS regularizers. In other words, the CS regularizer implicitly adapts its scale α to the scale of every argument $\mathbf{x}$. Furthermore, this CS regularizer is zero iff $\mathbf{x} \in \{-\alpha, \alpha\}^N$ for some $\alpha \in \mathbb{R}$, as only $x_n = \alpha$ or $x_n = -\alpha$ for $n = 1, \dots, N$ allows the right-hand-side of (7) to be zero.

We showcase the efficacy of ℓ_{bin} for the recovery of binary-valued solutions in Section 3.1 and compare its advantages to existing binarizing regularizers (cf. Section 1.3), e.g., being differentiable, scale-adaptive, and free of additional optimization parameters, in App. C.1.

2.2.2 ONE-SIDED BINARY

Define $\mathbf{g}(\mathbf{x}) \triangleq [x_1^2, \dots, x_N^2]^{\mathrm{T}}$ and $\mathbf{h}(\mathbf{x}) \triangleq [x_1, \dots, x_N]^{\mathrm{T}}$. Then, Proposition 1 yields the following CS regularizer that promotes one-sided binary-valued vectors; see App. A.4 for the derivation.

Regularizer 2 (One-Sided Binary). *Let $\mathbf{x} \in \mathbb{R}^N$ and define*

$$\ell_{\text{osb}}(\mathbf{x}) \triangleq [\![\mathbf{x}]\!]^2 [\![\mathbf{x}]\!]^4 - \left([\![\mathbf{x}]\!]^3\right)^2. \tag{8}$$

Then, the nonnegative function in (8) is only zero for one-sided binary-valued vectors, i.e., iff $\mathbf{x} \in \{0, \alpha\}^N$ for any $\alpha \in \mathbb{R}$. Furthermore, $\ell_{\text{osb}}(\mathbf{x})$ does not have any spurious critical points.

To gain insight into Regularizer 2, we invoke Lemma 1 and obtain

$$\ell_{\text{osb}}(\mathbf{x}) = [\![\mathbf{x}]\!]^2 \min_{\beta \in \mathbb{R}} \textstyle\sum_{n=1}^N \left(x_n(x_n - \beta)\right)^2. \tag{9}$$

Once again, we observe this CS regularizer's auto-scale property and only vectors of the form $\mathbf{x} \in \{0, \alpha\}^N$ for some $\alpha \in \mathbb{R}$ minimize (9).

2.2.3 SYMMETRIC TERNARY

Define $\mathbf{g}(\mathbf{x}) \triangleq [x_1^3, \dots, x_N^3]^{\mathrm{T}}$ and $\mathbf{h}(\mathbf{x}) \triangleq [x_1, \dots, x_N]^{\mathrm{T}}$. Then, Proposition 1 yields the following CS regularizer that promotes symmetric ternary-valued vectors; see App. A.5 for the derivation.

Regularizer 3 (Symmetric Ternary). *Let $\mathbf{x} \in \mathbb{R}^N$ and define*

$$\ell_{\text{ter}}(\mathbf{x}) \triangleq [\![\mathbf{x}]\!]^2 [\![\mathbf{x}]\!]^6 - \left([\![\mathbf{x}]\!]^4\right)^2. \tag{10}$$

Then, the nonnegative function in (10) is only zero for symmetric ternary-valued vectors, i.e., iff $\mathbf{x} \in \{-\alpha, 0, \alpha\}^N$ for any $\alpha \in \mathbb{R}$. Furthermore, $\ell_{\text{ter}}(\mathbf{x})$ does not have any spurious critical points.

To gain insight into Regularizer 3, we invoke Lemma 1 and obtain

$$\ell_{\text{ter}}(\mathbf{x}) = [\![\mathbf{x}]\!]^2 \min_{\beta \in \mathbb{R}_{\geq 0}} \textstyle\sum_{n=1}^N \left(x_n(x_n - \sqrt{\beta})(x_n + \sqrt{\beta})\right)^2. \tag{11}$$

As above, we observe this CS regularizer's auto-scale property and only vectors of the form $\mathbf{x} \in \{-\alpha, 0, \alpha\}^N$ for some $\alpha \in \mathbb{R}$ minimize (11).

The CS regularizers introduced so far promote binary- or ternary-valued vectors. In App. B.1, we detail an approach that generalizes CS regularizers to a symmetric, discrete-valued set with 2^B equispaced entries. In addition, al off the CS regularizers introduced above involve polynomials of higher (e.g. quartic) order, leading to potential numerical stability issues. In App. B.2, we propose alternative symmetric binarization regularizers that avoid such issues; similar alternative regularization functions can be derived for the other discretization regularizers.

2.3 RECOVERING EIGENVECTORS OF A GIVEN MATRIX

All CS regularizers introduced so far promote vectors with discrete-valued entries. In order to demonstrate the versatility of Proposition 1, we now propose a CS regularizer that promotes vectors that are eigenvectors of a given (and fixed) $\mathbf{C} \in \mathbb{R}^{N \times N}$ matrix.

Define $g(\mathbf{x}) \triangleq \mathbf{C}\mathbf{x}$ and $h(\mathbf{x}) = \mathbf{x}$. Then, Proposition 1 yields the following CS regularizer that promotes eigenvectors of $\mathbf{C}$; see App. A.6 for the derivation.

Regularizer 4 (Eigenvector Recovery). *Fix $\mathbf{C} \in \mathbb{R}^{N \times N}$, let $\mathbf{x} \in \mathbb{R}^N$, and define*

$$\ell_{\text{er}}(\mathbf{x}) \triangleq \|\mathbf{C}\mathbf{x}\|_2^2 \|\mathbf{x}\|_2^2 - (\mathbf{x}^T \mathbf{C}\mathbf{x})^2 \tag{12}$$

Then, the nonnegative function in (12) is only zero for eigenvectors of $\mathbf{C}$ and the all-zeros vector.

2.4 RECOVERING MATRICES WITH ORTHOGONAL COLUMNS

Finally, we demonstrate that Proposition 1 can also be used to promote structure in matrices. The following CS regularizer promotes matrices $\mathbf{X} \in \mathbb{R}^{N \times K}$ with $K \leq N$ to have orthogonal columns.

Define $\mathbf{g}(\mathbf{X}) \triangleq \mathrm{vec}(\mathbf{X}^{\mathrm{T}}\mathbf{X})$ and $\mathbf{h}(\mathbf{x}) \triangleq \mathrm{vec}(\mathbf{I}_M)$. Then, Proposition 1 yields the following CS regularizer that promotes matrices with orthogonal columns; see App. A.7 for the derivation.

Regularizer 5 (Orthogonal Matrix)**.** *Let $\mathbf{X} \in \mathbb{R}^{N \times K}$ with $K \leq N$ and define*

$$\ell_{\mathrm{om}}(\mathbf{X}) \triangleq K\|\mathbf{X}^{\mathrm{T}}\mathbf{X}\|_{\mathrm{F}}^2 - \|\mathbf{X}\|_{\mathrm{F}}^4. \tag{13}$$

Let $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$ be the singular value decomposition of $\mathbf{X}$. Then, we equivalently have,

$$\ell_{\mathrm{om}}(\mathbf{X}) \triangleq K\left(\textstyle\sum_{k=1}^{K} S_{k,k}^4\right) - \left(\textstyle\sum_{k=1}^{K} S_{k,k}^2\right)^2, \tag{14}$$

which is the symmetric binarizer from (6) applied to the singular values of $\mathbf{X}$. The nonnegative function in (13) is only zero for matrices $\mathbf{X}$ with pairwise orthogonal columns of equal length, i.e., iff $\mathbf{X}^{\mathrm{T}}\mathbf{X} = \alpha\mathbf{I}_N$ for $\alpha > 0$. Furthermore, $\ell_{\mathrm{om}}(\mathbf{X})$ does not have any spurious critical points.

2.5 GENERALIZATIONS AND VARIATIONS

Proposition 1 enables the design of a much broader range of CS regularizers, beyond those introduced so far. We now outline a range of possible generalizations and variations. In addition, we propose a range of non-differentiable variants of CS regularizers in App. B.5.

2.5.1 HÖLDER REGULARIZER

Proposition 1 can be generalized by replacing the CS inequality with Hölder's inequality (Hölder, 1889). This results in the following recipe; a short proof is given in App. A.8.

Proposition 2. *Fix two vector-valued functions $\mathbf{g}, \mathbf{h}\colon \mathbb{R}^N \to \mathbb{R}^N$ and define $\mathcal{X}$ as in (3). Let $p, q \geq 1$ so that $\frac{1}{p} + \frac{1}{q} = 1$ and let $r > 0$. Then, the nonnegative function*

$$\breve{\ell}(\mathbf{x}) \triangleq \|\mathbf{g}(\mathbf{x})\|_p^r \|\mathbf{h}(\mathbf{x})\|_q^r - |\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle|^r \tag{15}$$

is zero iff $\mathbf{x} \in \mathcal{X}$.

Proposition 1 is a special case of Proposition 2 by setting $p = q = r = 2$. A systematic investigation of the potency of regularizers resulting from Proposition 2 is left for future work.

2.5.2 SCALE-INVARIANT HÖLDER REGULARIZER

By slightly modifying the proof of Proposition 2, one can also develop Hölder regularizers that are *scale-invariant*, i.e., in which scaling the entire vector-valued function $\mathbf{g}(\mathbf{x})$ or $\mathbf{h}(\mathbf{x})$ with a nonzero constant has no impact on the regularizer's function value; see App. A.9 for the proof.

Proposition 3. *Fix two* vector-valued *functions $\mathbf{g}, \mathbf{h}\colon \mathbb{R}^N \to \mathbb{R}^N$ and define $\mathcal{X}$ as in (3). Let $p, q \geq 1$ so that $\frac{1}{p} + \frac{1}{q} = 1$ and let $r > 0$. Furthermore, set $\varepsilon \geq 0$. Then, the nonnegative function*

$$\bar{\ell}(\mathbf{x}) \triangleq \frac{\|\mathbf{g}(\mathbf{x})\|_p^r \|\mathbf{h}(\mathbf{x})\|_q^r + \varepsilon}{|\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle|^r + \varepsilon} - 1 \tag{16}$$

is zero iff $\mathbf{x} \in \mathcal{X}$.

Such scale-invariant regularization functions require special attention. First, while the parameter $\varepsilon > 0$ prevents the denominator in (16) from becoming zero, only $\varepsilon = 0$ leads to a scale-invariant regularizer. Second, regularizers derived from Proposition 3 may have significantly more spurious critical points than those obtained via Proposition 2. Third, evaluating the gradient of regularizers derived from (16) is typically more involved. Nonetheless, their (approximately) scale-invariant property might turn out useful in some applications and outweigh the above drawbacks. We note that scale-invariant versions of CS regularizers can be obtained as a special case of Proposition 3.

We conclude by noting that many other CS or Hölder regularizers can be derived when combining the above ideas. We also note that most of these results can be generalized to complex-valued vectors. A detailed investigation of such generalizations, variations, and specializations is left for future work.

3 Application Examples

We now showcase application examples for vector discretization, recovery of eigenvectors of a given matrix, recovering matrices with orthogonal columns, and quantization of neural network weights.

3.1 Recovering Discrete-Valued Vectors

The proposed CS regularizers enable the recovery of binary-, ternary- and two-bit-valued vectors from underdetermined linear systems of equations. To this end, we solve systems of linear equations $\mathbf{b} = \mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathbb{R}^{M\times N}$ has i.i.d. standard normal entries and $M < N$. We create vectors $\mathbf{x}^\star \in \mathbb{R}^N$, whose entries are chosen i.i.d. with uniform probability from $\{-1,+1\}$ for symmetric binary and from $\{0,+1\}$ for one-sided binary, and with probability $0.25, 0.5, 0.25$ from $\{-1,0,+1\}$, respectively, for ternary-valued vectors. For, two-bit-valued vectors, we create twp vectors $\mathbf{y}_1 \in \mathbb{R}^N$ and $\mathbf{y}_2 \in \mathbb{R}^N$ whose entries are chosen i.i.d. with uniform probability from $\{-1,+1\}$ and $\{-2,+2\}$, respectively, and calculate $\mathbf{x}^\star = \mathbf{y}_1 + \mathbf{y}_2$. Then, we calculate $\mathbf{b} = \mathbf{A}\mathbf{x}^\star$, and we try to recover the vector $\mathbf{x}^\star$ from $\mathbf{b}$ by solving optimization problems of the form

$$\hat{\mathbf{x}} \in \arg\min_{\tilde{\mathbf{x}}\in\mathbb{R}^N} \ell(\tilde{\mathbf{x}}) \quad \text{subject to } \mathbf{b} = \mathbf{A}\tilde{\mathbf{x}} \tag{17}$$

using a projected gradient descent algorithm—specifically, FISTA with backtracking (Beck & Teboulle, 2009; Goldstein)[3]. Here, $\ell(\tilde{\mathbf{x}})$ are the CS regularizers from Section 2.2. We fix $N = 100$ and vary M between 30 and 90.[4] We declare success for recovering $\mathbf{x}^\star$ if the returned solution $\hat{\mathbf{x}}$ satisfies $\|\hat{\mathbf{x}} - \mathbf{x}^\star\|_2/\|\mathbf{x}^\star\|_2 \leq 10^{-2}$.Fig. 1 shows the success probabilities with respect to the undersampling ratio γ along with error bars calculated from the standard error of the mean.

Symmetric Binary We first recover symmetric binary-valued solutions using Regularizer 1 with ℓ_{bin} from (6). For this scenario, Mangasarian & Recht (2011) showed that ℓ^∞-norm minimization recovers the binary-valued solution as long as $\gamma = M/N$ satisfies $\gamma > 0.5$ and N approaches infinity. Thus, our baseline is ℓ^∞-norm minimization, which we solve with Douglas–Rachford splitting as in (Studer et al., 2015). Fig. 1a shows the success rate for ℓ_{bin} and ℓ^∞-norm minimization with respect to the undersampling ratio γ. For ℓ_{bin} minimization, we observed that different initializations can have an impact on the success rate since the objective is non-convex. Thus, we allow at most 10 random initializations of projected gradient descent. We note that multiple initializations of Douglas-Rachford splitting does not affect its success rate as the ℓ^∞-norm is convex and the algorithm would converge to the same solution. We see from Fig. 1a that ℓ_{bin} minimization has, for any undersampling ratio γ, a higher probability of successfully recovering the true solution than ℓ^∞-norm minimization.

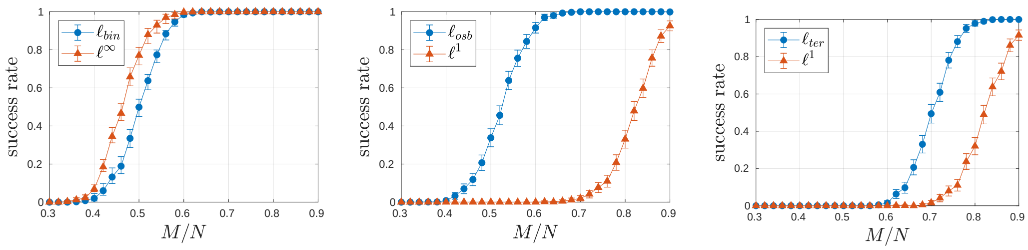
We discuss the advantages of ℓ_{bin} over existing binarizing regularizers in App. C.1. We also provide a comparison of the success rates of ℓ_{bin} and ℓ^∞-norm minimization with existing binarizing regularizers in App. C.2, and observe that ℓ_{bin} achievest the highest success rate. Moreover, in App. C.3, we provide the success rates of other binarizing CS regularizer variants (e.g., the Hölder, non-differentiable, and scale-invariant regularizers), where ℓ_{bin}, once again, achieves best performance.

One-Sided Binary We now recover vectors with one-sided binary-valued entries using Regularizer 2 with ℓ_{osb} from (8). In this experiment, we ran projected gradient descent for only one random initialization. Our baseline for this scenario is ℓ^1-norm minimization (Cai et al., 2009), as the generated vectors are sparse with half of the entries being nonzero (on average). We solve the ℓ^1-norm minimization problem with Douglas–Rachford splitting (Studer et al., 2015). Fig. 1b demonstrates that ℓ_{osb} minimization significantly outperforms ℓ^1-norm minimization for all undersampling ratios.

Symmetric Ternary We also recover vectors with symmetric ternary-valued entires using Regularizer 3 with ℓ_{ter} from (10). In this experiment, we ran projected gradient descent for only one random initialization. We use ℓ^1-norm minimization as our baseline as the generated vectors are sparse with half of the entries being nonzero (on average). Fig. 1c demonstrates that ℓ_{ter} minimization significantly outperforms ℓ^1-norm minimization for all undersampling ratios.

[3] We run projected gradient descent and the baseline algorithms for a maximum of 10^4 iterations.

[4] We also study the impact of the sparsity of $\mathbf{x}^\star$ while the number of measurements M is fixed in App. C.4. For each M, we randomly generate 1000 problem instances and report the average success probability.



(a) Symmetric binary (b) One-sided binary (c) Symmetric ternary

Figure 1: Probability of success for recovering vectors with (a) binary-, (b) one-sided binary-, and (c) symmetric ternary values dependent on the undersampling ratio M/N.

Symmetric Two-Bit We also recover vectors with symmetric two-bit valued entries using Regularizer 6 from (43). In this experiment, we ran projected gradient descent for 10 random initializations. Fig. 6 in App. C.5 demonstrates that ℓ_{equ} can recover symmetric two-bit vectors for all undersampling ratios with reasonable success probability.

3.2 Recovering Eigenvectors of a Matrix

The proposed CS regularizers also enable the recovery of eigenvectors of a given (and fixed) matrix. We once again consider a system of linear equations $\mathbf{b} = \mathbf{A}\mathbf{x}$, where $\mathbf{A} \in \mathbb{R}^{M\times N}$ has i.i.d. standard normal entries and $M < N$. We create vectors $\mathbf{x}^{\star} \in \mathbb{R}^{N}$ by uniform randomly choosing an eigenvector of a random matrix $\mathbf{C} \in \mathbb{R}^{N\times N}$ with i.i.d. standard normal entries. We calculate $\mathbf{b} = \mathbf{A}\mathbf{x}^{\star}$ and solve (17) using FISTA as in Section 3.1. As a baseline, we consider minimizing $\ell_{\mu}(\mathbf{x},\mu) \triangleq \|\mathbf{C}\mathbf{x} - \mu\mathbf{x}\|_2^2$, which includes an additional optimization parameter (i.e., μ) compared to minimizing $\ell(\tilde{\mathbf{x}}) = \ell_{\text{er}}(\mathbf{x})$ in (17). In App. C.6, Fig. 7, we observe that ℓ_{er} has significantly higher success rates than the baseline.

3.3 Recovering Matrices with Orthogonal Columns

We now demonstrate that the proposed regularizers can also impose structure to matrices. To this end, we consider a system of linear equations $\mathbf{A}\mathbf{X} = \mathbf{B}$, where $\mathbf{A} \in \mathbb{R}^{M\times N}$ has i.i.d. standard normal entries with $M < N$, and $\mathbf{X} \in \mathbb{R}^{N\times K}$ with $\mathbf{X}^{\text{T}}\mathbf{X} = \mathbf{I}_K$. We solve

$$\hat{\mathbf{X}} \in \arg\min_{\tilde{\mathbf{X}}\in\mathbb{R}^{N\times K}} \ \ell(\tilde{\mathbf{X}}) \quad \text{subject to } \mathbf{A}\tilde{\mathbf{X}} = \mathbf{B} \tag{18}$$

using FISTA as in Section 3.1. We have observed that for $N = K = 10$ and $N = K = 100$ and for values of M such that $M/N \in [0.1, 1]$, the output of FISTA was *always* an orthogonal matrix.

3.4 Quantizing Neural Network Weights

We now provide another application example for binarizing and ternarizing neural network weights. Our goal is to highlight the simplicity, versatility, and effectiveness of CS regularizers.

3.4.1 Method

Our weight quantization procedure consists of three steps: (i) training with CS regularizers, (ii) weight quantization, and (iii) continued training of remaining parameters. We detail these steps below. In order to demonstrate solely the impact of CS regularizers, we neither modify the neural network architecture (e.g., we do not alter the layers or activations) nor the forward-backward propagation stages, since we do not introduce any non-differentiable operations during training.

Step 1: Regularized Training Let $\boldsymbol{\theta}$ denote the set of all parameters of a neural network and $L(\boldsymbol{\theta})$ the loss function for learning the network's task. We solve the following optimization problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \ L(\boldsymbol{\theta}) + \lambda \textstyle\sum_{k=1}^{K} \eta_k \, \ell(\boldsymbol{\theta}_k). \tag{19}$$

Here, $\lambda \in \mathbb{R}_{\geq 0}$ is a regularization parameter, $\boldsymbol{\theta}_k$ is a vector consisting of the network weights which should share a common scaling factor (this can, for example, be an entire layer, one convolution kernel, or any other subset of network parameters), η_k is the associated normalization factor (e.g., the reciprocal value of the dimension of $\boldsymbol{\theta}_k$), and ℓ can be any CS regularizer (e.g., ℓ_{bin} or ℓ_{ter}).

After training the network parameters with the CS regularizers for a given number of epochs, the weights contained in the vectors $\boldsymbol{\theta}_k$ will be *concentrated* around $\{-\alpha_k, \alpha_k\}$ for symmetric binary-valued regularization or $\{-\alpha_k, 0, \alpha_k\}$ for symmetric ternary-valued regularization for some $\alpha_k > 0$.

Step 2: Quantization The goal is to quantize the regularized weights $\{\boldsymbol{\theta}_k\}_{k=1}^K$ from the previous step. In what follows, we describe the quantization procedure for one weight vector $\mathbf{w} = \boldsymbol{\theta}_k$.

We binarize the regularized weight vector $\mathbf{w}$ according to $\hat{\mathbf{w}} \triangleq \arg\min_{\mathbf{x} \in \mathcal{X}_{\text{bin}}} \|\mathbf{w} - \mathbf{x}\|_2^2$ with $\mathcal{X}_{\text{bin}} \triangleq \{\tilde{\mathbf{x}} \in \{-\alpha, \alpha\}^N : \alpha \in \mathbb{R}\}$ as in (Rastegari et al., 2016), which is given by

$$\hat{w}_n = \alpha^\star \operatorname{sign}(w_n),\ n = 1, \ldots, N,\ \text{with}\ \alpha^\star = \|\mathbf{w}\|_1 / N. \tag{20}$$

We ternarize the regularized weight vector $\mathbf{w} \neq \mathbf{0}_N$ according to $\hat{\mathbf{w}} \triangleq \arg\min_{\mathbf{x} \in \mathcal{X}_{\text{ter}}} \|\mathbf{w} - \mathbf{x}\|_2^2$ with $\mathcal{X}_{\text{ter}} \triangleq \{\tilde{\mathbf{x}} \in \{-\alpha, 0, \alpha\}^N : \alpha \in \mathbb{R}\}$ as in (Li et al., 2016), which is accomplished as follows: Let $\mathcal{I}_\tau = \{i : |w_i| \geq \tau\}$. Then, find the threshold that determines which entries of $\hat{\mathbf{w}}$ are nonzero as

$$\tau^\star = \arg \max_{\tau \in \{|w_i| : i \in \mathcal{I}\}} \tfrac{1}{|\mathcal{I}_\tau|} \left(\textstyle\sum_{i \in \mathcal{I}_\tau} |w_i|\right)^2. \tag{21}$$

Finally, compute the ternarized vector as

$$\hat{w}_n = \begin{cases} \alpha^\star \operatorname{sign}(w_n), & |w_n| \geq \tau^\star \\ 0, & \text{otherwise,} \end{cases} n = 1, \ldots, N,\ \text{with}\ \alpha^\star = \tfrac{1}{|\mathcal{I}_\tau|} \textstyle\sum_{i \in \mathcal{I}_\tau} |w_i|. \tag{22}$$

Step 3: Training with Quantized Weights After weight quantization, the number of trainable parameters is significantly reduced since we now have only one scale factor for a vector of quantized weights. Hence, we fix the signs of the weights and continue training only their shared scale factors alongside other tunable network parameters (e.g., biases, batch normalization parameters, etc.) without the use of CS regularizers and for a small number of epochs.

3.4.2 Experimental Results

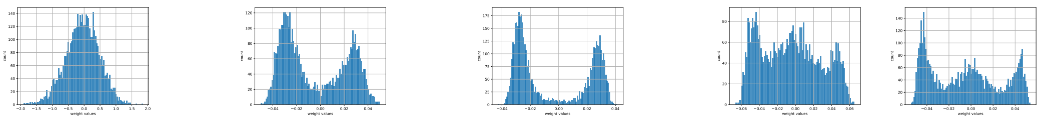
We conduct experiments on the benchmark datasets ImageNet (ILSVRC12) (Deng et al., 2009) and CIFAR-10 (Krizhevsky, 2009) for image classification. We follow classical data augmentation strategies as detailed in App. D.1.

Implementation As in (Rastegari et al., 2016; Qin et al., 2020; He et al., 2020), we regularize and quantize all network layers except for the first convolutional layer and the last fully-connected layer. For convolutional layers, we calculate the CS regularizer for vectors consisting of the weights in all kernels that produce one output channel; this leads to one scaling factor for each output channel following the approach from Rastegari et al. (2016). For fully-connected layers, we utilize one CS regularizer for each row of the weight matrix; this leads to one scaling factor for each output feature. We set the weights η_k in (19) to the reciprocal of the dimension of the elements in the corresponding weight vector $\boldsymbol{\theta}_k$.

For ImageNet, we use ResNet-18 (He et al., 2016) and initialize the weights with a pretrained full-precision model from PyTorch (pyt). In Step 1, we set the regularization parameter $\lambda = 10^3$ for binarization and $\lambda = 10^7$ for ternarization[5] We train the network for 40 epochs each in Steps 1 and 3, with a batch size of 1024. We use Adam optimizer (Kingma & Ba, 2017) with learning rate initialized by 0.001 and cosine annealing learning rate scheduler (Loshchilov & Hutter, 2016).

For CIFAR-10, we use ResNet-20 and initialize the weights with a pretrained full-precision model from Idelbayev, similarly to (Qin et al., 2020). We set $\lambda = 10$ for binarization and $\lambda = 10^5$ for ternarization. We train the network for 400 and 20 epochs in Steps 1 and 2, respectively, with a batch size of 128. The optimizer and learning rate schedule is the same as ImageNet.

[5]We chose the regularization parameter λ empirically based on using $1/10$th of the training sets for validation. We have observed that changes by a factor of 10 in λ has insignificant effect on the resulting accuracies. Please see Tables 4-7 in App. D.2 for an ablation study for varying λ.



(a) Pretrained (b) ℓ_{bin}, 5 epochs (c) ℓ_{bin}, 20 epochs (d) ℓ_{ter}, 5 epochs (e) ℓ_{ter}, 20 epochs

Figure 2: Neural network weight histograms of one output channel of a convolutional layer in the pretrained ResNet-18 model and the model after regularized training with ℓ_{bin} or ℓ_{ter} over ImageNet.

Weight Distribution Fig. 2 illustrates the impact of CS regularizers on the network weights with histograms for one output channel of one convolutional layer based on training ResNet-18 on ImageNet. We observe in Fig. 2(a) the weight distribution of the pretrained network resembles that of a zero-mean Gaussian distribution. Figures 2(b) and (c) reveal, as intended, that the weights are becoming more concentrated around binary values after 5 and 20 epochs of training with the regularizer ℓ_{bin}, respectively. Figures 2(d) and (e) reveal a similar behavior when using ℓ_{ter}.

Performance Evaluation In App. D.3, Tables 8-10, we provide the top-1 accuracy for our methods along with the full-precision baseline and various SOTA baselines that binarize or ternarize the weights of the network while the activations are left in full precision. All SOTA top-1 accuracy results in Tables 8-10 are taken from the corresponding papers.

For ImageNet, we observe from Tables 8 and Table 9 that for binary-valued weights, our approach outperforms SQ-BWN (Dong et al., 2017), BWN (Rastegari et al., 2016), HWGQ (Cai et al., 2017), PCCN (Gu et al., 2019), and the ternary TWN (Li et al., 2016), while the accuracy we achieve is 4.9% lower than the best SOTA method ProxyBNN (He et al., 2020). For ternary-valued weights, our approach outperforms TWN and SQ-TWN (Dong et al., 2017), while the accuracy we achieve is 2.8% lower than the SOTA method QIL (Jung et al., 2019).

For CIFAR-10, we observe from Table 10 that for binary-valued weights, our approach outperforms DoReFa-Net by a small margin and achieve the same performance as LQ-Net (Zhang et al., 2018), while the accuracy of our approach is 0.9% lower than the SOTA methods DAQ (Kim et al., 2021) and LCR-BNN (Shang et al., 2022); these two methods are also the only methods outperforming our ternary-valued approach by 0.2%.

While some of the SOTA methods achieve better accuracy than our approach, our results (i) require a simpler training procedure[6] and (ii) showcase the potential of CS regularizers: We only have one step of regularized training with full-precision weights, a quantization step, and a second step of training with fewer parameters; this procedure does not require any additional storage at any stage of training. In contrast, all of the baseline methods retain both the quantized and full-precision values for the weights during training, and use the quantized weights in forward and backward propagation while the full-precision values are updated with the gradients calculated with respect to the quantized values. This results in additional storage. Moreover, to reduce the quantization error or to alleviate the mismatch between forward and backward propagation, references (Gu et al., 2019; Hu et al., 2018; Kim et al., 2021; He et al., 2020; Yang et al., 2019; Zhang et al., 2018; Jung et al., 2019) introduce more trainable parameters and Shang et al. (2022) proposes a regularization method that requires the construction of matrices that scale with the square of the number of features in one layer. Please see Table 3 for a detailed comparison of the

We conclude by noting that the proposed CS regularizers could be combined with any of these existing approaches for possible further accuracy improvements.

4 Limitations

While the proposed CS regularizers provide a recipe for designing regularization functions with a wide variety of properties, they suffer from a range of limitations summarized next.

First and foremost, CS regularizers are typically nonconvex. While we have been able to prove that some of the proposed nonconvex CS regularizers are free of any spurious critical points, convergence

[6]Please see Table 3 for the advantages/disadvantages of our training strategy compared to the SOTA methods.

to a global minimum depends on the combination of the objective, regularizer, and optimization algorithm. Thus, multiple random restarts of the optimizer might be necessary in practice.

Furthermore, some of the CS regularizers involve higher-order polynomials (e.g., the ternarization regularizer in (10) involves eighth-order polynomials), which can result in poor convergence behavior, especially around their minimum. To counteract this issue, one can either resort to non-differentiable variants outlined in App. B.5 or to scale-invariant variants outlined in Section 2.5.2. In addition, utilizing adaptive step-size selection methods and schedules that adapt (e.g., increase) the regularization parameter λ over iterations could also be used to improve convergence.

Finally, we have only scratched the surface of many of the specializations, variations, and generalizations put forward in Section 2.5. Besides that, we have only investigated the efficacy of CS regularizers with two example applications, i.e., solving underdetermined systems of linear equations and neural network weight quantization with a simple training recipe, in Section 3. A thorough theoretical analysis and simulative study of alternative CS regularizers in a broader range of applications, as well as combining CS regularizers with sophisticated SOTA quantized neural network training procedures, such as the ones in (He et al., 2020; Jung et al., 2019), are left for future work.

5 Conclusions

We have proposed Cauchy–Schwarz (CS) regularizers, a novel class of regularization functions that can be designed to promote a wide range of properties. We have derived example regularization functions that promote discrete-valued vectors, eigenvectors to matrices, or matrices with orthogonal columns, and we have outlined a range of specializations, variations, and generalizations that lead to an even broader class of new and possibly more powerful regularizers. For solving underdetermined systems of linear equations, we have shown that CS regularizers can outperform well-established baseline methods, such as ℓ^∞-norm or ℓ^1-norm minimization. For weight quantization of neural networks, we have shown that utilizing CS regularizers enables one to achieve competitive accuracy to existing quantization methods while using a simple training procedure.

References

https://pytorch.org/vision/0.13/models/generated/torchvision.models.resnet18.html.

Yu Bai, Yu-Xiang Wang, and Edo Liberty. ProxQuant: Quantized neural networks via proximal operators. In *International Conference on Learning Representations, ICLR*, 2019.

Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *IEEE Transactions on Image Processing*, 18(11):2419–2434, January 2009.

T. Tony Cai, Guangwu Xu, and Jun Zhang. On recovery of sparse signals via ℓ_1-norm minimization. *IEEE Transactions on Information Theory*, 55(7):3388–3397, June 2009.

Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave Gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Oscar Castañeda, Sven Jacobsson, Giuseppe Durisi, Mikael Coldrey, Tom Goldstein, and Christoph Studer. 1-bit massive MU-MIMO precoding in VLSI. *IEEE J. Emerging Sel. Topics Circuits Syst.*, 7(4):508–522, December 2017.

Yoojin Choi, Mostafa El-Khamy, and Jungwon Lee. Learning sparse low-precision neural networks with learnable regularization. *IEEE Access*, 8:96963–96974, 2020.

Clayton W. Commander. *Maximum cut problem, MAX-CUTMaximum Cut Problem, MAX-CUT*, pp. 1991–1999. Springer US, Boston, MA, 2009. ISBN 978-0-387-74759-0. doi: 10.1007/978-0-387-74759-0_358. URL https://doi.org/10.1007/978-0-387-74759-0_358.

Sajad Darabi, Mouloud Belbahri, Matthieu Courbariaux, and Vahid Partovi Nia. Regularized binary network training. *arXiv preprint arXiv:1812.11800*, 2019.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comp. Vision and Patt. Recog.*, pp. 248–255, May 2009. doi: 10.1109/CVPR.2009.5206848.

Yinpeng Dong, Jianguo Li, and Renkun Ni. Learning accurate low-bit deep neural networks with stochastic quantization. In *BMVC*, 01 2017. doi: 10.5244/C.31.189.

Ahmed T Elthakeb, Prannoy Pilligundla, Fatemehsadat Mireshghallah, Tarek Elgindi, Charles-Alban Deledalle, and Hadi Esmaeilzadeh. WaveQ: Gradient-based deep quantization of neural networks through sinusoidal adaptive regularization. *arXiv preprint arXiv:2003.00146*, 2020.

Tom Goldstein. fasta-matlab. `https://github.com/tomgoldstein/fasta-matlab`. Accessed: 2024-03-30.

Ruihao Gong, Xianglong Liu, Shenghu Jiang, Tianxiang Li, Peng Hu, Jiazhen Lin, Fengwei Yu, and Junjie Yan. Differentiable soft quantization: Bridging full-precision and low-bit neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4852–4861, 2019.

Jiaxin Gu, Ce Li, Baochang Zhang, Jungong Han, Xianbin Cao, Jianzhuang Liu, and David Doermann. Projection convolutional neural networks for 1-bit CNNs via discrete back propagation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8344–8351, July 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. in IEEE Conf. on Comp. Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016. doi: 10.1109/CVPR.2016.90.

Xiangyu He, Zitao Mo, Ke Cheng, Weixiang Xu, Qinghao Hu, Peisong Wang, Qingshan Liu, and Jian Cheng. ProxyBNN: Learning binarized neural networks via proxy matrices. In *European Conference on Computer Vision*, pp. 223–241. Springer, August 2020.

Otto Hölder. Ueber einen Mittelwerthssatz. *Nachrichten von der Königlichen Gesellschaft der Wissenschaften und der Georg-Augusts-Universität zu Göttingen*, (2):38–47, 1889.

Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

Qinghao Hu, Peisong Wang, and Jian Cheng. From hashing to CNNs: Training binary weight networks via hashing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, April 2018.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.

Pei-Hen Hung, Chia-Han Lee, Shao-Wen Yang, V Srinivasa Somayazulu, Yen-Kuang Chen, and Shao-Yi Chien. Bridge deep learning to the physical world: An efficient method to quantize network. In *IEEE Workshop on Signal Processing Systems (SiPS)*. IEEE, 2015.

Yerlan Idelbayev. Proper ResNet implementation for CIFAR10/CIFAR100 in PyTorch. `https://github.com/akamaster/pytorch_resnet_cifar10`. Accessed: 2024-04-30.

Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, 2019.

Dohyung Kim, Junghyup Lee, and Bumsub Ham. Distance-aware quantization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5251–5260, October 2021. doi: 10.1109/ICCV48922.2021.00522.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, January 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Cong Leng, Zesheng Dou, Hao Li, Shenghuo Zhu, and Rong Jin. Extremely low bit neural network: Squeeze the last bit out with admm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Fengfu Li, Bo Zhang, and Bin Liu. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Zhi-quan Luo, Wing-kin Ma, Anthony Man-cho So, Yinyu Ye, and Shuzhong Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, May 2010.

Olvi L. Mangasarian and Benjamin Recht. Probability of unique integer solution to a system of linear equations. *Eur. J. Oper. Res.*, 214(1):27–30, October 2011.

Yoshiki Matsuda. Benchmarking the MAX-CUT problem on the simulated bifurcation machine, 209. URL https://medium.com/toshiba-sbm/benchmarking-the-max-cut-problem-on-the-simulated-bifurcation-machine-e26e1127c0b0 Accessed: 2024-06-27.

Maxim Naumov, Utku Diril, Jongsoo Park, Benjamin Ray, Jedrzej Jablonski, and Andrew Tulloch. On periodic functions as regularizers for quantization of neural networks. *arXiv preprint arXiv:1811.09862*, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. 2019.

Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2247–2256, June 2020. doi: 10.1109/CVPR42600.2020.00232.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: ImageNet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Ryan Razani, Gregoire Morin, Eyyub Sari, and Vahid Partovi Nia. Adaptive binary-ternary quantization. In *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4613–4618, June 2021.

Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

Sohil Shah, Abhay Kumar Yadav, Carlos D Castillo, David W Jacobs, Christoph Studer, and Tom Goldstein. Biconvex relaxation for semidefinite programming in computer vision. pp. 717–735, September 2016.

Yuzhang Shang, Dan Xu, Bin Duan, Ziliang Zong, Liqiang Nie, and Yan Yan. Lipschitz continuity retained binary neural network. In *European Conference on Computer Vision*, pp. 603–619. Springer, October 2022.

Kirill Solodskikh, Vladimir Chikin, Ruslan Aydarkhanov, Dehua Song, Irina Zhelavskaya, and Jiansheng Wei. Towards accurate network quantization with equivalent smooth regularizer. In *European Conference on Computer Vision*, pp. 727–742. Springer, 2022.

J. Michael Steele. *The Cauchy–Schwarz Master Class: an Introduction to the Art of Mathematical Inequalities*. Cambridge University Press, 2004.

Christoph Studer, Tom Goldstein, Wotao Yin, and Richard G. Baraniuk. Democratic representations. *arXiv:1401.3420*, April 2015.

Sueda Taner and Christoph Studer. ℓ^p–ℓ^q-norm minimization for joint precoding and peak-to-average-power ratio reduction. In *Proc. Asilomar Conf. Signals, Syst., Comput.*, pp. 437–442, October 2021.

Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Linh Tran, Maja Pantic, and Marc Peter Deisenroth. Cauchy–Schwarz regularized autoencoder. *Journal of Machine Learning Research*, 23(115):1–37, 2022.

Matthias Wess, Sai Manoj Pudukotai Dinakarrao, and Axel Jantsch. Weighted quantization-regularization in DNNs for weight memory minimization toward HW implementation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2929–2939, 2018.

Sheng Xu, Yanjing Li, Teli Ma, Mingbao Lin, Hao Dong, Baochang Zhang, Peng Gao, and Jinhu Lu. Resilient binary neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 10620–10628, 2023.

Huanrui Yang, Lin Duan, Yiran Chen, and Hai Li. BSQ: Exploring bit-level sparsity for mixed-precision neural network quantization. *arXiv preprint arXiv:2102.10462*, 2021.

Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7308–7316, 2019.

Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. LQ-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 365–382, 2018.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Appendix: Cauchy–Schwarz Regularizers

A Proofs and Derivations

A.1 Proof of Proposition 1

From the Cauchy–Schwarz inequality (Steele, 2004) follows that

$$|\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle| \leq \|\mathbf{g}(\mathbf{x})\|_2 \|\mathbf{h}(\mathbf{x})\|_2. \tag{23}$$

Squaring both sides of (23) and rearranging terms leads to

$$0 \leq \|\mathbf{g}(\mathbf{x})\|_2^2 \|\mathbf{h}(\mathbf{x})\|_2^2 - |\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle|^2 \triangleq \ell(\mathbf{x}). \tag{24}$$

Equality in (23) holds iff $\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x})$, for which $\ell(\mathbf{x}) = 0$.

A.2 Proof of Lemma 1

Assume that $\mathbf{h}(\mathbf{x}) \neq \mathbf{0}$. Then,

$$\frac{\partial \|\mathbf{g}(\mathbf{x}) - \beta \mathbf{h}(\mathbf{x})\|_2^2}{\partial \beta} = 0 \implies \beta = \frac{\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle}{\|\mathbf{h}(\mathbf{x})\|_2^2}. \tag{25}$$

Plugging the right-hand-side into $\|\mathbf{g}(\mathbf{x}) - \beta \mathbf{h}(\mathbf{x})\|_2^2$ yields

$$\min_{\beta \in \mathbb{R}} \|\mathbf{g}(\mathbf{x}) - \beta \mathbf{h}(\mathbf{x})\|_2^2 = \|\mathbf{g}(\mathbf{x})\|_2^2 - \frac{|\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle|^2}{\|\mathbf{h}(\mathbf{x})\|_2^2}. \tag{26}$$

Multiplying both sides by $\|\mathbf{h}(\mathbf{x})\|_2^2$ results in

$$\ell(\mathbf{x}) = \|\mathbf{h}(\mathbf{x})\|_2^2 \min_{\beta \in \mathbb{R}} \|\mathbf{g}(\mathbf{x}) - \beta \mathbf{h}(\mathbf{x})\|_2^2. \tag{27}$$

If $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, then (27) still holds. By swapping $\mathbf{g}(\mathbf{x})$ with $\mathbf{h}(\mathbf{x})$, the equalities in (5) follow.

A.3 Derivation of Regularizer 1

Regularizer 1 is minimized by vectors $\mathbf{x} \in \mathbb{R}^N$ that satisfy the linear dependence condition $\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x})$ for the specific choices $\mathbf{g}(\mathbf{x}) = [x_1^2, \ldots, x_N^2]^{\mathrm{T}}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{1}_N$. We have

$$\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x}) \iff \exists (a_1, a_2) \in \mathbb{R}^2 \backslash \{(0,0)\} : a_1 x_n^2 = a_2, \quad n = 1, \ldots, N. \tag{28}$$

If $a_1 \neq 0$, then $x_n^2 = a_2/a_1$ which implies $x_n = \pm\alpha$, $n = 1, \ldots, N$, for some $\alpha \in \mathbb{R}$. If $a_1 = 0$ then $a_2 \neq 0$, so the condition $a_1 x_n^2 = a_2$ cannot be satisfied; this implies that the only vectors that satisfy $\ell_{\text{bin}}(\mathbf{x}) = 0$ from (6) are in the following set:

$$\mathcal{X}_{\text{bin}} = \left\{ \tilde{\mathbf{x}} \in \{-\alpha, \alpha\}^N : \alpha \in \mathbb{R} \right\}. \tag{29}$$

The same result would also follow directly from inspection of (7).

To establish the fact that Regularizer 1 does not have any spurious critical points, we need to show that $\nabla \ell_{\text{bin}}(\mathbf{x}) = \mathbf{0}$ iff $\mathbf{x} \in \mathcal{X}_{\text{bin}}$. To this end, we inspect

$$\tfrac{\partial \ell_{\text{bin}}(\mathbf{x})}{\partial x_n} = 4N x_n^3 - 4[\![\mathbf{x}]\!]^2 x_n = 4x_n(N x_n^2 - [\![\mathbf{x}]\!]^2) = 0, \quad n = 1, \ldots, N. \tag{30}$$

Clearly, every vector $\mathbf{x} \in \mathcal{X}_{\text{bin}}$ satisfies (30). For any other vector, the gradient is nonzero. To prove this, it is sufficient to show that the derivative is nonzero for a two-dimensional, non-binary-valued vector, because any vector with a non-binary-valued subvector is non-binary-valued (and, any non-binary-valued vector has a non-binary-valued subvector). To this end, let $\mathbf{x} = [\alpha, \beta]^{\mathrm{T}}$ for $\alpha \neq \beta$, $\alpha \neq -\beta$, and $\alpha, \beta \neq 0$. Then, we have

$$\tfrac{\partial \ell_{\text{bin}}(\mathbf{x})}{\partial x_1} = 4\alpha(2\alpha^2 - (\alpha^2 + \beta^2)) = 4\alpha(\alpha^2 - \beta^2) \neq 0. \tag{31}$$

A.4 DERIVATION OF REGULARIZER 2

Regularizer 2 is minimized by vectors $\mathbf{x} \in \mathbb{R}^N$ that satisfy the linear dependence condition $\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x})$ for the specific choices $\mathbf{g}(\mathbf{x}) = [x_1^2, \ldots, x_N^2]^\mathrm{T}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{x}$. We have

$$\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x}) \iff \exists (a_1, a_2) \in \mathbb{R}^2 \backslash \{(0,0)\} : a_1 x_n^2 = a_2 x_n, \quad n = 1, \ldots, N. \tag{32}$$

If $x_n = 0$, then the condition $a_1 x_n^2 = a_2 x_n$ is trivially satisfied. If $x_n \neq 0$, then we inspect $a_1 x_n = a_2$. If $a_1 \neq 0$, then $x_n = a_2/a_1$, which implies $x_n = \alpha$ for some $\alpha \in \mathbb{R}$. If $a_1 = 0$ then $a_2 \neq 0$, so the condition $a_1 x_n = a_2$ cannot be satisfied; this implies that the only vectors that satisfy $\ell_{\text{osb}}(\mathbf{x}) = 0$ from (8) are in the following set:

$$\mathcal{X}_{\text{osb}} = \big\{\tilde{\mathbf{x}} \in \{0, \alpha\}^N : \alpha \in \mathbb{R}\big\}. \tag{33}$$

The same result would also follow directly from inspection of (9).

To establish the fact that Regularizer 2 does not have any spurious critical points, we need to show that $\nabla \ell_{\text{osb}}(\mathbf{x}) = \mathbf{0}$ iff $\mathbf{x} \in \mathcal{X}_{\text{osb}}$. To this end, we inspect

$$\tfrac{\partial \ell_{\text{osb}}(\mathbf{x})}{\partial x_n} = 2x_n\big([\![\mathbf{x}]\!]^4 + 2x_n^2[\![\mathbf{x}]\!]^2 - 3x_n[\![\mathbf{x}]\!]^3\big) = 0, \quad n = 1, \ldots, N. \tag{34}$$

Clearly, every vector $\mathbf{x} \in \mathcal{X}_{\text{osb}}$ satisfies (34). For any other vector, the gradient is nonzero. To prove this, it is sufficient to show that the derivative is nonzero for a two-dimensional, non-one-sided-binary-valued (non-OSB) vector, because any vector with a non-OSB subvector is non-OSB (and, any non-OSB vector has a non-OSB subvector). Assume $\mathbf{x} = [\alpha, \beta]^\mathrm{T}$ for $\alpha \neq \beta$ and $\alpha, \beta \neq 0$. Then $\frac{\partial \ell_{\text{osb}}(\mathbf{x})}{\partial x_1} = 2\alpha\beta^2(2\alpha - \beta)(\alpha - \beta)$, and by symmetry, $\frac{\partial \ell_{\text{osb}}(\mathbf{x})}{\partial x_2} = 2\alpha^2\beta(\alpha - 2\beta)(\alpha - \beta)$; this implies that $\frac{\partial \ell_{\text{osb}}(\mathbf{x})}{\partial x_1}$ and $\frac{\partial \ell_{\text{osb}}(\mathbf{x})}{\partial x_2}$ cannot be zero simultaneously.

A.5 DERIVATION OF REGULARIZER 3

Regularizer 3 is minimized by vectors $\mathbf{x} \in \mathbb{R}^N$ that satisfy the linear dependence condition $\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x})$ for the specific choices $\mathbf{g}(\mathbf{x}) = [x_1^3, \ldots, x_N^3]^\mathrm{T}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{x}$. We have

$$\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x}) \iff \exists (a_1, a_2) \in \mathbb{R}^2 \backslash \{(0,0)\} : a_1 x_n^3 = a_2 x_n, \quad n = 1, \ldots, N. \tag{35}$$

If $x_n = 0$, then the condition $a_1 x_n^2 = a_2 x_n$ is trivially satisfied. If $x_n \neq 0$, then we inspect $a_1 x_n^2 = a_2$. If $a_1 \neq 0$, then $x_n^2 = a_2/a_1$, which implies $x_n = \pm\alpha$ for some $\alpha \in \mathbb{R}$. If $a_1 = 0$ then $a_2 \neq 0$, so the condition $a_1 x_n^2 = a_2$ cannot be satisfied; this implies that the only vectors that satisfy $\ell_{\text{osb}}(\mathbf{x}) = 0$ from (10) are in the following set:

$$\mathcal{X}_{\text{ter}} = \big\{\tilde{\mathbf{x}} \in \{-\alpha, 0, \alpha\}^N : \alpha \in \mathbb{R}\big\}. \tag{36}$$

The same result would also follow directly from inspection of (11).

To establish the fact that Regularizer 3 does not have any spurious critical points, we need to show that $\nabla \ell_{\text{ter}}(\mathbf{x}) = \mathbf{0}$ iff $\mathbf{x} \in \mathcal{X}_{\text{ter}}$. To this end, we inspect

$$\tfrac{\partial \ell_{\text{ter}}(\mathbf{x})}{\partial x_n} = 2x_n\big([\![\mathbf{x}]\!]^6 + 3[\![\mathbf{x}]\!]^2 x_n^4 - 4[\![\mathbf{x}]\!]^4 x_n^2\big) = 0, \quad n = 1, \ldots, N. \tag{37}$$

Clearly, every vector $\mathbf{x} \in \mathcal{X}_{\text{ter}}$ satisfies (37). For any other vector, the derivative is nonzero. To prove this, it is sufficient to show that the derivative is nonzero for a two-dimensional, non-ternary-valued vector, because any vector with a non-ternary-valued subvector is non-ternary-valued (and, any non-ternary-valued vector has a non-ternary-valued subvector). Assume $\mathbf{x} = [\alpha, \beta]$ for $\alpha \neq \beta$, $\alpha \neq -\beta$ and $\alpha, \beta \neq 0$. Then $\frac{\partial \ell_{\text{ter}}(\mathbf{x})}{\partial x_1} = 2\alpha\beta^2(3\alpha^2 - \beta^2)(\alpha^2 - \beta^2)$, and, by symmetry, $\frac{\partial \ell_{\text{ter}}(\mathbf{x})}{\partial x_2} = 2\alpha^2\beta(\alpha^2 - 3\beta^2)(\alpha^2 - \beta^2)$; this implies that $\frac{\partial \ell_{\text{ter}}(\mathbf{x})}{\partial x_1}$ and $\frac{\partial \ell_{\text{ter}}(\mathbf{x})}{\partial x_2}$ cannot be zero simultaneously.

A.6 DERIVATION OF REGULARIZER 4

Regularizer 4 is minimized by vectors $\mathbf{x} \in \mathbb{R}^N$ that satisfy the linear dependence condition $\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x})$ for the specific choices $\mathbf{g}(\mathbf{x}) = \mathbf{C}\mathbf{x}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{x}$. By definition, we have that $\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x})$ if and only if $\mathbf{x}$ is an eigenvector of $\mathbf{C}$.

A.7 Derivation of Regularizer 5

Regularizer 5 is minimized by matrices $\mathbf{X} \in \mathbb{R}^N$ that satisfy the linear dependence condition $\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x})$ for the specific choices $\mathbf{g}(\mathbf{X}) \triangleq \mathrm{vec}(\mathbf{X}^\mathrm{T}\mathbf{X})$ and $\mathbf{h}(\mathbf{x}) \triangleq \mathrm{vec}(\mathbf{I}_M)$.

To establish the fact that Regularizer 5 does not have any spurious critical points, we need to show that $\nabla \ell_{\text{om}}(\mathbf{x}) = \mathbf{0}$ iff $\mathbf{x} \in \mathcal{X}_{\text{om}}$. To this end, we inspect

$$\begin{aligned}\nabla {\ell_{\text{om}}}_{\mathbf{X}}(\mathbf{X}) &= 4N\mathbf{X}\mathbf{X}^\mathrm{T}\mathbf{X} - 4\|\mathbf{X}\|_\mathrm{F}^2\mathbf{X} = \mathbf{0} &(38)\\ &\Rightarrow 4\mathbf{X}(N\mathbf{X}^\mathrm{T}\mathbf{X} - \|\mathbf{X}\|_\mathrm{F}^2\mathbf{I}_N) = \mathbf{0} &(39)\\ &\Rightarrow N\mathbf{X}^\mathrm{T}\mathbf{X} = \|\mathbf{X}\|_\mathrm{F}^2\mathbf{I}_N, &(40)\end{aligned}$$

which implies that $\mathbf{X}$ must have orthogonal columns.

A.8 Proof of Proposition 2

From the triangle inequality and Hölder's inequality (Hölder, 1889) follows that

$$|\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle| \leq \textstyle\sum_{n=1}^N \big|[\mathbf{g}(\mathbf{x})]_n[\mathbf{h}(\mathbf{x})]_n\big| \leq \|\mathbf{g}(\mathbf{x})\|_p\|\mathbf{h}(\mathbf{x})\|_q, \tag{41}$$

where $p, q \geq 1$ so that $\frac{1}{p} + \frac{1}{q} = 1$. Raising the left-hand and the right-hand sides of (41) to the power of $r > 0$ and rearranging terms leads to

$$0 \leq \|\mathbf{g}(\mathbf{x})\|_p^r\|\mathbf{h}(\mathbf{x})\|_q^r - |\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle|^r \triangleq \breve{\ell}(\mathbf{x}). \tag{42}$$

Both inequalities in (41) hold iff $\mathbf{g}(\mathbf{x}) \sim \mathbf{h}(\mathbf{x})$, for which $\breve{\ell}(\mathbf{x}) = 0$.

A.9 Proof of Proposition 3

The proof follows that of App. A.8, but where we first add $\varepsilon \geq 0$ to the left-hand and right-hand sides of (41), followed by a division by $|\langle \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x})\rangle| + \varepsilon$ and rearranging terms to arrive at $\bar{\ell}(\mathbf{x})$ in (16).

B Alternative Cauchy–Schwarz Regularizers

B.1 Beyond Vector Ternarization

We now show one approach that generalizes CS regularizers to a symmetric, discrete-valued set with 2^B equispaced entries. The idea behind this approach is as follows: (i) decompose $\mathbf{x} \in \mathbb{R}^N$ into a sum of B auxiliary vectors $\mathbf{x} = \sum_{b=1}^B \mathbf{y}_b$ with $\mathbf{y}_b \in \mathbb{R}^N$ and (ii) apply one regularization function to the auxiliary vectors $\mathbf{y}_b$, $b = 1, \ldots, B$.

Define $\mathbf{g}\big(\{\mathbf{y}_b\}_{b=1}^B\big) = \big[\tilde{\mathbf{g}}(\mathbf{y}_1)^\mathrm{T}, \ldots, \tilde{\mathbf{g}}(\mathbf{y}_B)^\mathrm{T}\big]^\mathrm{T}$ using $\tilde{\mathbf{g}}(\mathbf{y}) = [y_1^2, \ldots, y_N^2]^\mathrm{T}$ and $\mathbf{h}\big(\{\mathbf{y}_b\}_{b=1}^B\big) = \big[\tilde{\mathbf{h}}_1(\mathbf{y}_1)^\mathrm{T}, \ldots, \tilde{\mathbf{h}}_B(\mathbf{y}_B)^\mathrm{T}\big]^\mathrm{T}$ using $\tilde{\mathbf{h}}_b(\mathbf{y}) = 4^{b-1}\mathbf{1}_N$. Then, Proposition 1 yields the following CS regularizer that promotes symmetric equispaced-valued vectors; see App. B.1.1 for the derivation.

Regularizer 6 (Symmetric Equispaced). *Let $\mathbf{y}_b \in \mathbb{R}^N$ for $b = 1, \ldots, B$ and define*

$$\ell_{\text{equ}}\big(\{\mathbf{y}_b\}_{b=1}^B\big) \triangleq KN\Big(\textstyle\sum_{b=1}^B [\![\mathbf{y}_b]\!]^4\Big) - \Big(\textstyle\sum_{b=1}^B 4^{b-1}[\![\mathbf{y}_b]\!]^2\Big)^2 \tag{43}$$

with $K \triangleq \sum_{b=1}^B 4^{2(b-1)}$. Then, the nonnegative function (43) is only zero for vectors $\mathbf{y}_b \in \{-2^{b-1}\alpha, 2^{b-1}\alpha\}^N \cup \mathbf{0}_N, b = 1, \ldots, B$, for any $\alpha \in \mathbb{R}$; this implies that the sum of these vectors $\mathbf{x} \triangleq \sum_{b=1}^B \mathbf{y}_b$ is in the set $\mathcal{X}_{\text{equ}} \triangleq \{\pm(2^b - 1)\alpha\}_{b=1}^B$ with $|\mathcal{X}_{\text{equ}}| = 2B$. Furthermore, ℓ_{equ} does not have any spurious critical points.

To gain insight into Regularizer 6, we invoke Lemma 1 and obtain

$$\ell_{\text{equ}}(\{\mathbf{y}_b\}_{b=1}^B) = KN \min_{\beta \in \mathbb{R}} \sum_{b=1}^B \big((y_b[n] - 2^{b-1}\sqrt{\beta})(y_b[n] + 2^{b-1}\sqrt{\beta})\big)^2 \tag{44}$$

We also observe this CS regularizer's auto-scale property and only vectors of the form $\mathbf{y}_b \in \{-2^{b-1}\alpha, 2^{b-1}\alpha\}^N$ for some $\alpha \in \mathbb{R}$ minimize (44). This implies that the vectors $\mathbf{x}$ are of the form $\mathbf{x} \in \{-(2^B-1)\alpha, \ldots, -\alpha, \alpha, \ldots, (2^B-1)\alpha\}^N$ for some $\alpha \in \mathbb{R}$.

In contrast to the initially introduced binarization and ternarization regularizers, Regularizer 6 introduces additional optimization parameters, i.e., increases the dimension of the optimization problem by a factor of B.

B.1.1 Derivation of Regularizer 6

Regularizer 6 is minimized by vectors $\{\mathbf{y}_b\}_{b=1}^B$ that satisfy the linear dependence condition $\mathbf{g}(\{\mathbf{y}_b\}_{b=1}^B) \sim \mathbf{h}(\{\mathbf{y}_b\}_{b=1}^B)$ for the specific choices $\mathbf{g}(\{\mathbf{y}_b\}_{b=1}^B) = [\tilde{\mathbf{g}}(\mathbf{y}_1), \ldots, \tilde{\mathbf{g}}(\mathbf{y}_B)]^\mathrm{T}$ using $\tilde{\mathbf{g}}(\mathbf{y}) = [y_1^2, \ldots, y_N^2]^\mathrm{T}$ and $\mathbf{h}(\{\mathbf{y}_b\}_{b=1}^B) = [\tilde{\mathbf{h}}(\mathbf{y}_1), \ldots, \tilde{\mathbf{h}}(\mathbf{y}_B)]^\mathrm{T}$ using $\tilde{\mathbf{h}}(\mathbf{y}_b) = 4^{b-1}\mathbf{1}_N$. We have

$$\begin{aligned}
&\mathbf{g}(\{\mathbf{y}_b\}_{b=1}^B) \sim \mathbf{h}(\{\mathbf{y}_b\}_{b=1}^B) \\
&\iff \exists (a_1, a_2) \in \mathbb{R}^2 \backslash \{(0,0)\} : a_1 y_{b,n}^2 = a_2 4^{b-1}, \quad b = 1, \ldots, B,\ n = 1, \ldots, N.
\end{aligned} \tag{45}$$

If $a_1 \neq 0$, then $y_{n,b}^2 = \frac{a_2}{a_1} 4^{b-1}$ which implies $y_{n,b} = 0$ or $y_{b,n} = \pm\alpha 2^{b-1}$, $n = 1, \ldots, N$, $b = 1, \ldots, B$, for some $\alpha \in \mathbb{R}$. If $a_1 = 0$ then $a_2 \neq 0$, so the condition $a_1 y_{b,n}^2 = a_2 4^{b-1}$ cannot be satisfied; this implies that the only vectors $\mathbf{y}_b$ that satisfy $\ell_\text{equ}(\{\mathbf{y}_b\}_{b=1}^B) = 0$ from (43) are in the following set:

$$\mathcal{Y}_{b,\alpha} = \{-2^{b-1}\alpha, 2^{b-1}\alpha\}^N \cup \mathbf{0}_N, \tag{46}$$

with $\mathbf{y}_b \in \mathcal{Y}_{b,\alpha}, b = 1, \ldots, B$ for any $\alpha \in \mathbb{R}$. The same result would also follow directly from inspection of (44). Then, the vectors $\mathbf{x} = \sum_{b=1}^B \mathbf{y}_b$ are in the following set:

$$\mathcal{X}_\text{equ} = \left\{\mathbf{x} \in \{-2^{B-1}\alpha, \ldots, -\alpha, \alpha, \ldots, 2^{B-1}\alpha\}^N : \alpha \in \mathbb{R}\right\}. \tag{47}$$

To establish the fact that Regularizer 6 does not have any spurious critical points, we need to show that $\nabla \ell_\text{equ}(\mathbf{y}_b) = \mathbf{0}, b = 1, \ldots, B$ iff $\mathbf{y}_b \in \mathcal{Y}_{b,\alpha}, b = 1, \ldots, B$ for any $\alpha \in \mathbb{R}$. To this end, we inspect

$$\frac{\partial \ell_\text{equ}(\{\mathbf{y}_b\}_{b=1}^B)}{\partial y_b[n]} = 4 y_b[n] \left(KN (y_b[n])^2 - 4^{b-1} \textstyle\sum_{\tilde{b}=1}^B 4^{\tilde{b}-1} [\![\mathbf{y}_{\tilde{b}}]\!]^2 \right) = 0 \tag{48}$$

for $n = 1, \ldots, N$ and $b = 1, \ldots, B$. Clearly, $\mathbf{y}_b \in \mathcal{Y}_{b,\alpha}, b = 1, \ldots, B$ satisfies (48). For a set of vectors in any other form, the derivative is nonzero. To prove this, it is sufficient to show that the derivative is nonzero for a pair of scalars (i.e., $N = 1$) $(y_b, y_{b'})$ for $y_b \neq 0, |y_b| \neq 2^{b-1}|\alpha|$ and $|y_{b'}| = 2^{b'-1}|\alpha|$, because any pair of vectors including these entries would not satisfy (48) (and, any set of vectors that do not satisfy (48) must have such a pair of entries). We have,

$$\frac{\partial \ell_\text{equ}(y_b, y_{b'})}{\partial y_b} = 4 y_b 4^{2(b'-1)} \left(y_b^2 - 4^{b-1}\alpha^2\right) \neq 0. \tag{49}$$

B.2 Bounded CS Regularizers for Vector Binarization

We now propose alternative binarization regularizers that avoids potential numerical issues caused by higher-order polynomials.

Define $b(x) \triangleq (1 + x^2)^{-1}$ and $\mathbf{g}(\mathbf{x}) \triangleq [b(x_1), \ldots, b(x_N)]^\mathrm{T}$. Furthermore, let $\mathbf{h}(\mathbf{x}) \triangleq \mathbf{1}_N$. Then, Proposition 1 yields the following CS regularizer that promotes symmetric binary-valued vectors.

Regularizer 7 (Bounded Symmetric Binarizer)**.** *Let $\mathbf{x} \in \mathbb{R}^N$ and define*

$$\ell_\text{bbin}(\mathbf{x}) \triangleq N \textstyle\sum_{n=1}^N \frac{1}{(1+x_n^2)^2} - \left(\sum_{n=1}^N \frac{1}{1+x_n^2}\right)^2 \tag{50}$$

Then, the nonnegative function in (50) is only zero for one-sided binary-valued vectors, i.e., iff $\mathbf{x} \in \{0, \alpha\}^N$ for any $\alpha \in \mathbb{R}$. Furthermore, $\ell_\text{bbin}(\mathbf{x})$ does not have any spurious critical points.

An alternative binarization regularizer can be obtained as follows. Define $b(x) \triangleq e^{-x^2}$ and $\mathbf{g}(\mathbf{x}) \triangleq [b(x_1), \ldots, b(x_N)]^{\mathrm{T}}$. Furthermore, let $\mathbf{h}(\mathbf{x}) \triangleq \mathbf{1}_N$. Then, Proposition 1 yields the following CS regularizer that promotes symmetric binary-valued vectors.

Regularizer 8 (Alternative Bounded Symmetric Binarizer)**.** *Let* $\mathbf{x} \in \mathbb{R}^N$ *and define*

$$\ell_{\text{bin,exp}}(\mathbf{x}) \triangleq N \sum_{n=1}^{N} e^{-2x_n^2} - \left(\sum_{n=1}^{N} e^{-2x_n^2}\right)^2 \tag{51}$$

Then, the nonnegative function in (51) is only zero for one-sided binary-valued vectors, i.e., iff $\mathbf{x} \in \{0, \alpha\}^N$ *for any* $\alpha \in \mathbb{R}$.

Note that by normalizing (50) and (51) with $1/N^2$, the maximum value of the resulting CS regularizer is bounded from above by 1.

B.3 Vectors in Nullspace of a Given Matrix

The following regularizer promotes unit-norm vectors in the nullspace of a given (and fixed) matrix $\mathbf{C} \in \mathbb{R}^{M \times N}$. Define $\mathbf{g}(\mathbf{x}) \triangleq [(\mathbf{C}\mathbf{x})^{\mathrm{T}}, \|\mathbf{x}\|_2^2 - 1, 1]^{\mathrm{T}}$ and $\mathbf{h}(\mathbf{x}) \triangleq [\mathbf{0}_{M \times 1}^{\mathrm{T}}, 0, 1]^{\mathrm{T}}$. Then, Proposition 1 yields the following CS regularizer that promotes unit-norm vectors in nullspace of $\mathbf{C}$.

Regularizer 9 (Nullspace Vector)**.** *Fix* $\mathbf{C} \in \mathbb{R}^{M \times N}$ *with* $M \geq N$ *and let* $\mathbf{x} \in \mathbb{R}^N$*. Define*

$$\ell_{\text{ns}}(\mathbf{x}) \triangleq \|\mathbf{C}\mathbf{x}\|_2^2 + (\|\mathbf{x}\|_2^2 - 1)^2 \tag{52}$$

Then, the nonnegative function in Regularizer 52 is zero for only unit-norm vectors $\mathbf{x}$ *in the nullspace of* $\mathbf{C}$*, i.e., iff* $\mathbf{C}\mathbf{x} = \mathbf{0}_{M \times 1}$ *with* $\|\mathbf{x}\|_2^2 = 1$.

B.4 CS Regularizers with Fixed Scale

If one is, for example, interested in promoting binary-valued vectors with predefined scale, i.e., $\mathbf{x} \in \{-\alpha, \alpha\}^N$ but for a given *fixed* value of α, then one can use $\mathbf{g}(\mathbf{x}) \triangleq [x_1^2, \ldots, x_N^2, \alpha^2]^{\mathrm{T}}$ and $\mathbf{h}(\mathbf{x}) \triangleq \mathbf{1}_{N+1}$ in (4). We note, however, that this particular binarization regularizer with $\alpha = 1$ has been utilized before in (Tang et al., 2017); similar regularizers can be found in (Hung et al., 2015; Darabi et al., 2019). In general, the idea of augmenting the functions $\mathbf{g}$ and $\mathbf{h}$ with constants removes the auto-scale property of CS regularizers.

B.5 Non-Differentiable Variants

One can also develop non-differentiable variants of CS regularizers. For example, by defining $\mathbf{g}(\mathbf{x}) \triangleq \left[|x_1|, \ldots, |x_N|\right]^{\mathrm{T}}$ and $\mathbf{h}(\mathbf{x}) \triangleq \mathbf{1}_N$ in Proposition 1, one obtains the CS regularizer

$$\tilde{\ell}_{\text{bin}}(\mathbf{x}) \triangleq N [\![\mathbf{x}]\!]^2 - \|\mathbf{x}\|_1^2, \tag{53}$$

which also promotes symmetric binary-valued entries. Intriguingly, this regularizer is equal to the scaled empirical variance of the entry-wise absolute values of $\mathbf{x} \in \mathbb{R}^N$, i.e., $\tilde{\ell}_{\text{bin}}(\mathbf{x}) = N^2 \mathrm{Var}(|\mathbf{x}|)$. One could also combine the idea of (53) with Proposition 2 using $p = q = 2$ and $r = 1$ to obtain

$$\breve{\ell}_{\text{bin}}(\mathbf{x}) \triangleq \sqrt{N} \|\mathbf{x}\|_2 - \|\mathbf{x}\|_1, \tag{54}$$

which also promotes symmetric binary-valued entries. Such alternative versions might result in better empirical convergence if, for example, used within auto-differentiation frameworks that allow for non-differentiable functions. We conclude by noting that the specific regularizer in (53) has been used in (Taner & Studer, 2021) for dynamic-range reduction of complex-valued data in wireless systems.

C Additional Results

C.1 Comparison of the Binarizing CS Regularizer with Existing Regularizers

Table 1 summarizes the key properties of existing regularizers from Section 1.3 and how our regularizer can be superior to those, i.e., by being differentiable, scale-adaptive, and avoiding additional optimization parameters.

Table 1: Comparison of regularizers for vector binarization. Advantages are designated by (+) and disadvantages by (-).

Regularizer	Differentiable (+)	Scale-adaptive (+)	Requires additional optimization variables (-)
$\sum_n(\|x_n\|-1)^2$	No	No	No
$\sum_n(\|x_n\|-\beta)^2$	No	Yes	Yes
$\sum_n(x_n^2-1)^2$	Yes	No	No
$\sum_n(x_n^2-\beta)^2$	Yes	Yes	Yes
Ours (ℓ_{bin})	Yes	Yes	No

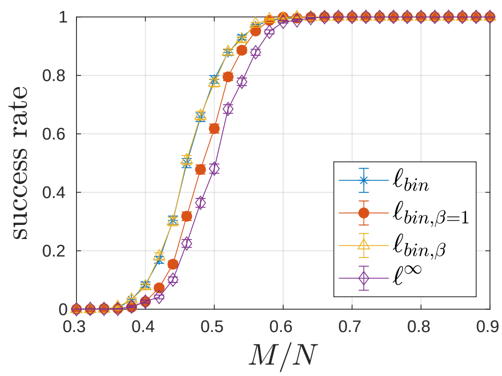


Figure 3: Probability of success in binary solution recovery with ℓ_{bin} and three baselines

C.2 Comparisons with Baselines for Binary Recovery

We follow the same experimental setup as in Section 3.1 and provide experiments for two additional baselines: (i) assuming that the scale β is known and fixed as a constant and (ii) letting β be a separate (and explicit) optimization parameter (that is learned together with the entries of the vector). As mentioned in Section 1.3, we use $\ell_{\text{bin},\beta=1} \triangleq \sum_{n=1}^{N}(x_n^2-\beta)^2$ with known and fixed $\beta=1$, and $\ell_{\text{bin},\beta} \triangleq \sum_{n=1}^{N}(x_n^2-\beta)^2$ with additional optimization parameter β.

In Fig. 3, we observe that both of these baseline methods achieve comparable recovery performance, but CS regularizers have the advantages of (i) not requiring to know the scale a-priori and (ii) not introducing additional optimization parameters.

C.3 Additional Results for More CS Regularizers for Binary Recovery

We follow the same experimental setup as Section 3.1 and provide experiments for four additional variants of CS regularizers: Here, $\ell_{\text{bin,H}}$ refers to the Hölder CS regularizer from (15) with $p=q=2, r=1$, $\ell_{\text{bin,si}}$ to the scale-invariant Hölder CS regularizer from (16) with $p=q=2, r=1$, $\tilde{\ell}_{\text{bin}}$ to the non-differentiable CS regularizer from (53), and $\tilde{\ell}_{\text{bin,exp}}$ to the bounded CS regularizer from (51).

In Fig. 4, we observe that while $\ell_{\text{bin,H}}$ has comparable success rate to ℓ_{bin}, the remaining variants are unfortunately outperformed by the baseline ℓ^{∞}-norm.

C.4 Additional Results for Sparse Recovery

In this subsection, we slightly modify our experimental setup from Section 3.1 in order to compare the solution recovery performance of ℓ_{osb}- and ℓ_{ter}-minimization to that of ℓ^1-norm minimization with respect to the sparsity of $\mathbf{x}^\star$. We fix $N=100$ and $M=75$. We create vectors $\mathbf{x}^\star \in \mathbb{R}^N$ with a fixed number of K uniform randomly chosen nonzero entries; these nonzero entries are $+1$ for one-sided binary vectors, and are chosen i.i.d. with uniform probability from $\{-1,+1\}$ for ternary vectors. We vary K from 20 to 80. For each K, we randomly generate 1000 problem instances and report the

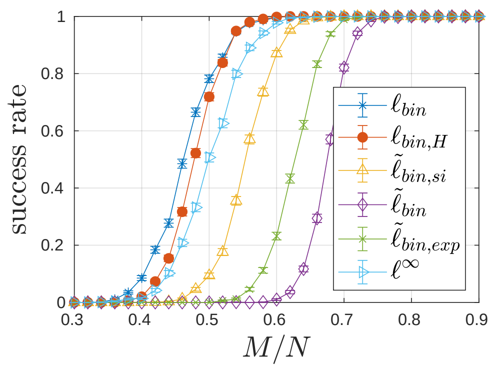
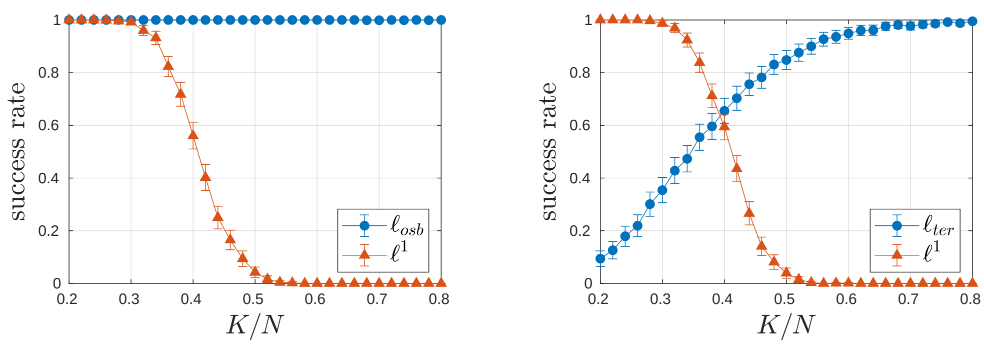


Figure 4: Probability of success in binary solution recovery with five CS regularizer variants and ℓ^{∞}-norm-based recovery.



(a) One-sided binary

(b) Symmetric ternary

Figure 5: Probability of success for recovering vectors with (a) one-sided binary-, and (b) symmetric ternary values dependent on the density ratio K/N.

average success probability and the standard deviation from the mean. We only allow one random initialization, and the remaining details of the setup are the same as those presented in Section 3.1.

Fig. 5 demonstrates the success rate of (a) ℓ_{osb}-minimization and (b) ℓ_{ter}-minimization compared to ℓ^1-norm minimization with respect to the density ratio of $\mathbf{x}^{\star}$ given by $\delta = K/N$. In Fig. 5 (a), we observe that while the success rate of ℓ^1-norm minimization reduces with $\delta > 0.3$ and almost reaches 0 at $\delta = 0.5$, the success rate of ℓ_{osb}-minimization is almost always 1 for any density ratio. In Fig. 5 (b), we observe that ℓ^1-norm minimization follows the same trend as in Fig. 5 (a) as expected, while the success rate of ℓ_{ter}-minimization increases with density. The success rate of ℓ_{ter}-minimization surpasses that of ℓ^1-norm minimization for $\delta > 0.4$. For small density ratios δ, CS regularization with ℓ_{ter} does perform poorly as projected gradient descent seems to get stuck in local minima. To counteract this issue, one could perform multiple restarts with random initialization.

C.5 Simulation Results for Two-Bit Solution Recovery from Section 3.1

Please see Fig. 6.

C.6 Simulation Results for Eigenvector Recovery from Section 3.2

Please see Fig. 7.

C.7 Approximating Maximum-Cut Problems with CS Regularizers

@sue

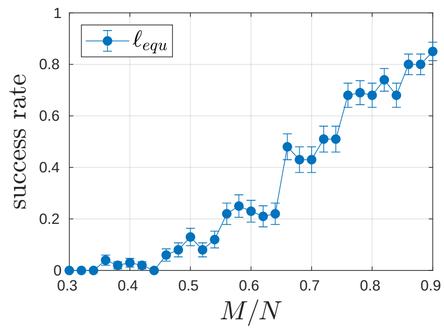


Figure 6: Two-bit solution recovery for $N = 10$ with CS regularizer ℓ_{equ}.

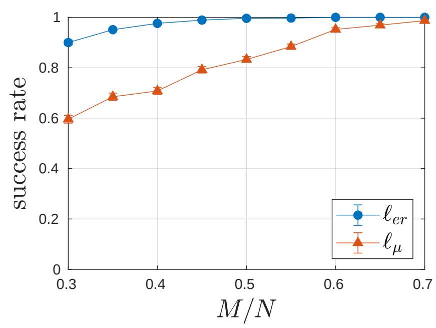


Figure 7: Eigenvector recovery for $N = 100$ with ℓ_{er} and with ℓ_{μ}, where μ must also be learned, as a baseline.

We now showcase another application in which CS regularizers can be utilized. Specifically, CS regualzers can be used to find approximate solutions to the well-known *weighted maximum cut* (MAX-CUT) problem (Commander, 2009). MAX-CUT of a graph is the partition of a graph's vertices into two disjoint sets such that the total weight of the edges between these two sets is maximized. For an undirected weighted graph $G = (V, E)$, this maximization problem can be formulated as the following integer quadratic programming problem:

$$\underset{\mathbf{x}\in\{-1,+1\}^N}{\text{maximize}} \quad \underbrace{\frac{1}{2}\sum_{1\leq i<j\leq N} w_{ij}(1 - x_i x_j)}_{\triangleq\, \ell_{\text{MC}}(\mathbf{x})}. \tag{55}$$

Here, $x_i \in \{-1, +1\}$, $i = 1, \dots, N$, denotes the binary set label for the ith vertex of the graph, N is the number of noces, and $w_{ij} \in \mathbb{R}$ denotes the weight of the edge between the ith and jth vertices.

The MAX-CUT problem is NP-hard and many approximations have been proposed in the literature. Classical approximations base on semidefinite and continuous relaxation; see, e.g, Commander (2009) and the references therein. Here, we propose a continuous reformulation that utilizes CS regularizers:

$$\hat{\mathbf{x}} \in \arg\min_{\mathbf{x}\in\mathbb{R}^N} -\ell_{\text{MC}}(\mathbf{x}) + \lambda\ell_{\text{bin}}(\mathbf{x}) \quad \text{subject to } |x_i| \leq 1, \tag{56}$$

which we solve by using a projected gradient descent algorithm, similarly to Section 2.2 with a fixed maximum number of iterations.

To evaluate our approach, we ran our projected gradient descent algorithm with random initializations for 10 trials. First, we considered small graphs; here, we set $\lambda = 1$.

- For the $N = 5, E = 7$ graph from Matsuda (209), we recovered the MAX-CUT in less than 300 iterations across 10 trials with random initializations.

- For a graph with $N = 4$ vertices, $E = 5$ edges, and weights $w_{12} = 10$, $w_{13} = 20$, $w_{14} = 30$, $w_{24} = 40$, and $w_{34} = 50$, we recovered the MAX-CUT in less than 40 iterations across 10 trials.

Second, we considered the larger graphs given in Matsuda (209) for benchmarking; here, we set $\lambda = 10^{-7}$. The table below demonstrates the graph ID and the maximum cut values from Matsuda (209) along with the average cut values of our random initializations and our recovered solutions across 10 trials. Unfortunately, it seems that our approach struggles to recover the MAX-CUT for large graphs; however, it significantly improves the objective value compared to the initialization, demonstrating its practical applicability. Moreover, with notable computational advantages over the method in Matsuda (209), our approach shows promise and could inspire valuable directions for future research.

Table 2: Comparison of Maximum Known Cut and Ours for Different Graphs

Graph ID	Target	Initial	Ours
G10	2000	67.4±50.5	1769.1±26.8
G11	564	7.2±14.9	482.0±10.46
G12	556	-0.2±24.3	470.8±13.7
G13	582	22.6±25.1	494.2±8.1

D Details of Neural Network Quantization Experiments from Section 3.4

D.1 Datasets and Preprocessing

ImageNet has over 1.2 M training images and 50 k validation images from 1000 object classes. We train and evaluate our network on the training and validation splits, respectively, and report the top-1 accuracy for performance evaluation. We adopt a typical data augmentation strategy on the training images as resizing the shorter side of the images to 256 pixels, taking a random crop of size 224×224, and applying a random horizontal flip. For validation, we apply the same resizing and take the center 224×224 crop.

CIFAR-10 (Krizhevsky, 2009) consists of over 50 k training images and 10 k testing images from 10 object classes. We adopt a typical data-augmentation strategy on the training images as padding by 4 pixels, taking a random 32×32 crop, and applying a random horizontal flip. For testing, we use the original images.

D.2 The Impact of Varying the Regularization Parameter on the Classification Accuracies

Please see Tables 4-7. These accuracies are based on the average of 10 runs.

D.3 Performance Comparison with SOTA Binarized and Ternarized Neural Networks from Section 3.4.2

Please see Table 3 for the advantages/disadvantages of our training strategy compared to the SOTA methods.

Please see Tables 8-10 for accuracy comparisons. Here, we report the average accuracy and standard deviation for 10 random initializations of training.

E Computational Resources

For our underdetermined linear systems experiments in Section 3.1, we used MATLAB. For the maximum number of 10^4 iterations, projected gradient descent and Douglas-Rachford splitting algorithms each took approximately one second at most.

Table 3: Comparison of variables that are required by SOTA neural network quantization methods and CS regularizers (ours) for training. Each column represents variables that are required *in addition* to (unquantized) full-precision neural network training. Due to lack of space, we cannot list all SOTA methods in this table, but we emphasize that ours is the only method that introduces no trainable or non-trainable variables with only one hyperparameter; please see our response to Q2 of Reviewer Kqkn to observe that our method is *not* sensitive to small changes in this parameter.

Method	Trainable variables	Non-trainable variables	Tunable hyper-parameters
SQ-BWN (Dong et al., 2017)	Yes	Yes	0
BWN (Rastegari et al., 2016)	No	Yes	0
HWGQ (Cai et al., 2017)	No	Yes	0
PCCN (Gu et al., 2019)	Yes	Yes	0
BWHN (Hu et al., 2018)	No	Yes	0
ADMM (Leng et al., 2018)	Yes	No	1
IR-Net (Qin et al., 2020)	No	Yes	0
LCR-BNN (Shang et al., 2022)	No	Yes	2
DAQ (Kim et al., 2021)	No	Yes	1
ProxyBNN (He et al., 2020)	Yes	Yes	1
TWN (Li et al., 2016)	No	Yes	1
QNet (Yang et al., 2019)	No	Yes	1
QIL (Jung et al., 2019)	Yes	Yes	0
DoReFa-Net (Zhou et al., 2016)	No	Yes	0
LQ (Zhang et al., 2018)	Yes	Yes	0
DSQ (Gong et al., 2019)	Yes	Yes	0
Ours (ℓ_{bin} and ℓ_{ter})	No	No	1

Table 4: Top-1 accuracy of binarized ResNet-18 on ImageNet for regularized training with varying values of λ.

λ	1	10	100	1000	1e4
Top-1 %	62.4	**62.8**	62.7	59.8	57.4

Table 5: Top-1 accuracy of ternarized ResNet-18 on ImageNet for regularized training with varying values of λ.

λ	1e3	1e4	1e5	1e6	1e7	1e8
Top-1 %	62.8	64.1	**65.3**	65.3	65.1	64.0

Table 6: Top-1 accuracy of binarized ResNet-20 on CIFAR10 for regularized training with varying values of λ.

λ	1	10	100	1000
Top-1 %	89.8	**90.7**	90.1	89.9

Table 7: Top-1 accuracy of ternarized ResNet-20 on CIFAR10 for regularized training with varying values of λ.

λ	1000	1e4	1e5	1e6	1e7
Top-1 %	90.8	90.9	**91.0**	91.0	90.9

Method	Top-1 (%)
ResNet-18 (FP)	69.8
SQ-BWN (Dong et al., 2017)	58.4
BWN (Rastegari et al., 2016)	60.8
HWGQ (Cai et al., 2017)	61.3
PCCN (Gu et al., 2019)	63.5
BWHN (Hu et al., 2018)	64.3
ADMM (Leng et al., 2018)	64.8
IR-Net (Qin et al., 2020)	66.5
LCR-BNN (Shang et al., 2022)	66.9
DAQ (Kim et al., 2021)	67.2
ProxyBNN (He et al., 2020)	67.7
Ours (ℓ_{bin})	62.8±0.09

Table 8: Top-1 accuracy of ResNet-18 with binary-valued weights on ImageNet. FP stands for the full-precision model accuracy.

Method	Top-1 (%)
ResNet-18 (FP)	69.8
TWN (Li et al., 2016)	61.8
SQ-TWN (Dong et al., 2017)	63.8
QNet (Yang et al., 2019)	66.5
ADMM (Leng et al., 2018)	67.0
LQ (Zhang et al., 2018)	68.0
QIL (Jung et al., 2019)	68.1
Ours (ℓ_{ter})	65.3±0.08

Table 9: Top-1 accuracy of ResNet-18 with ternary-valued weights on ImageNet. FP stands for the full-precision model accuracy.

Method	Top-1 (%)
ResNet-20 (FP)	91.7
DoReFa-Net (Zhou et al., 2016)	90.0
LQ (Zhang et al., 2018)	90.1
DSQ (Gong et al., 2019)	90.2
IR-Net (Qin et al., 2020)	90.8
DAQ (Kim et al., 2021)	91.2
LCR-BNN (Shang et al., 2022)	91.2
Ours (ℓ_{bin})	90.3±0.17
Ours (ℓ_{ter})†	91.0±0.11

Table 10: Top-1 accuracy of ResNet-20 on CIFAR-10 (c) with binary- and ternary-valued weights. FP stands for the full-precision model accuracy.

For our neural network weight quantization experiments in Section 3.4, we use PyTorch (Paszke et al., 2019). We used a machine with eight NVIDIA GeForce RTX 4090 GPUs with 24 GB memory. Training ResNet-18 on ImageNet took approximately 800 seconds per epoch for a batch size of 1024. Training ResNet-20 on CIFAR-10 took approximately five seconds for a batch size of 128.