Training Quantized Neural Networks to Global Optimality via Semidefinite Programming

Burak Bartan [1] **Mert Pilanci** [1]

Abstract

Neural networks (NNs) have been extremely successful across many tasks in machine learning. Quantization of NN weights has become an important topic due to its impact on their energy efficiency, inference time and deployment on hardware. Although post-training quantization is well-studied, training optimal quantized NNs involves combinatorial non-convex optimization problems which appear intractable. In this work, we introduce a convex optimization strategy to train quantized NNs with polynomial activations. Our method leverages hidden convexity in two-layer neural networks from the recent literature, semidefinite lifting, and Grothendieck's identity. Surprisingly, we show that certain quantized NN problems can be solved to global optimality provably in polynomial time in all relevant parameters via tight semidefinite relaxations. We present numerical examples to illustrate the effectiveness of our method.

1. Introduction

In this paper we focus on training quantized neural networks for efficient machine learning models. We consider the combinatorial and non-convex optimization of minimizing empirical error with respect to quantized weights. We focus on polynomial activation functions, where the training problem is still non-trivial to solve.

Recent work has shown that two-layer neural networks with ReLU (Pilanci & Ergen, 2020; Sahiner et al., 2021a) and leaky ReLU activations (Lacotte & Pilanci, 2020b) can be trained via convex optimization in polynomial time with respect to the number of samples and neurons. Moreover, degree-two polynomial activations can be trained to global optimality in polynomial time with respect to all problem dimensions using semidefinite programming (Bartan & Pilanci, 2021). In this work, we take a similar convex duality approach that involves semidefinite programming. However, our method and theoretical analysis are substantially different since we consider quantized weights, which involves a discrete non-convex optimization problem. The fact that the first layer weights are discrete renders it a combinatorial NP-hard problem and thus we cannot hope to obtain a similar result as in (Bartan & Pilanci, 2021) or (Pilanci & Ergen, 2020). In contrast, in (Bartan & Pilanci, 2021) it is shown that with the constraint $\|u_j\|_2 = 1$ and ℓ_1-norm regularization on the second layer weights, the global optimum can be found in fully polynomial time and that the problem becomes NP-hard in the case of quadratic regularization (i.e. weight decay).

The approach that we present in this paper for training quantized neural networks is significantly different from others in the quantization literature. In particular, our approach involves deriving a semidefinite program (SDP) and designing a sampling algorithm based on the solution of the SDP. Our techniques lead to a provable guarantee for the difference between the resulting loss and the optimal non-convex combinatorial loss. To the best of our knowledge, this is the first method that provides provably optimal neural networks with quantized parameters.

1.1. Prior work

Recently, there has been a lot of research effort in the realm of compression and quantization of neural networks for hardware implementations. In (Zhu et al., 2016), the authors proposed a method that reduces network weights into ternary values by performing training with ternary values. Experiments in (Zhu et al., 2016) show that their method does not suffer from performance degradation and achieve 16x compression compared to the original model. The authors in (Gong et al., 2014) focus on compressing dense layers using quantization to tackle model storage problems for large-scale models. The work presented in (Han et al., 2015) also aims to compress deep networks using a combination of pruning, quantization and Huffman coding. In (Lin et al., 2015), the authors present a quantization scheme where they use different bit-widths for different layers (i.e.,

[1]Department of Electrical Engineering, Stanford University, CA, USA. Correspondence to: Burak Bartan <bbartan@stanford.edu>, Mert Pilanci <pilanci@stanford.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

bit-width optimization). Other works that deal with fixed point training include (Lin & Talathi, 2016), (Gupta et al., 2015), (Hwang & Sung, 2014). Furthermore, (Anwar et al., 2015) proposes layer-wise quantization based on ℓ_2-norm error minimization followed by retraining of the quantized weights. However, these studies do not address optimal approximation. In comparison, our approach provides optimal quantized neural networks.

In (Allen-Zhu & Li, 2020), it was shown that the degree two polynomial activation functions perform comparably to ReLU activation in practical deep networks. Specifically, it was reported in (Allen-Zhu & Li, 2020) that for deep neural networks, ReLU activation achieves a classification accuracy of 0.96 and a degree two polynomial activation yields an accuracy of 0.95 on the Cifar-10 dataset. Similarly for the Cifar-100 dataset, it is possible to obtain an accuracy of 0.81 for ReLU activation and 0.76 for the degree two polynomial activation. These numerical results are obtained for the activation $\sigma(t) = t + 0.1t^2$. Furthermore, in encrypted computing, it is desirable to have low degree polynomials as activation functions. For instance, homomorphic encryption methods can only support additions and multiplications in a straightforward way. These constraints make low degree polynomial activations attractive for encrypted networks. In (Gilad-Bachrach et al., 2016), degree two polynomial approximations were shown to be effective for accurate neural network predictions with encryption. These results demonstrate that polynomial activation neural networks are a promising direction for further exploration.

Convexity of infinitely wide neural networks was first considered in (Bengio et al., 2006) and later in (Bach, 2017). A convex geometric characterization of finite width neural networks was developed in (Ergen & Pilanci, 2020a; Ergen & Pilanci, 2019; Bartan & Pilanci, 2019). Exact convex optimization representations of finite width two-layer ReLU neural network problems were developed first in (Pilanci & Ergen, 2020) and extended to leaky ReLU (Lacotte & Pilanci, 2020b) and polynomial activation functions (Bartan & Pilanci, 2021). These techniques were also extended to other network architectures including three-layer ReLU (Ergen & Pilanci, 2021), autoencoder (Sahiner et al., 2021b), autoregressive (Gupta et al., 2021), batch normalization (Ergen et al., 2021) and deeper networks (Ergen & Pilanci, 2020b).

1.2. Notation

We use $X \in \mathbb{R}^{n\times d}$ to denote the data matrix throughout the text, where its rows $x_i \in \mathbb{R}^d$ correspond to data samples and columns are the features. $y \in \mathbb{R}^n$ denotes the target vector. We use $\ell(\hat{y}, y)$ for convex loss functions where $\hat{y}$ is the vector of predictions and $\ell^*(v) = \sup_z(v^T z - \ell(z, y))$ denotes its Fenchel conjugate. tr refers to matrix trace. $\mathrm{sign}(\cdot)$ is the elementwise sign function. We use the notation $Z \succeq 0$ for positive semidefinite matrices (PSD). We use $\circ$ for the Hadamard product of vectors and matrices. The symbol $\otimes$ denotes the Kronecker product. We use $\lambda_{\max}(\cdot)$ to denote the largest eigenvalue of its matrix argument. If the input to $\mathrm{diag}(\cdot)$ is a vector, then the result is a diagonal matrix with its diagonal entries equal to the entries of the input vector. If the input to $\mathrm{diag}(\cdot)$ is a matrix, then the result is a vector with entries equal to the diagonal entries of the input matrix. $\bar{1}$ refers to the vector of 1's. $\mathbb{S}^{d\times d}$ represents the set of $(d \times d)$-dimensional symmetric matrices.

2. Lifting Neural Network Parameters

We focus on two-layer neural networks with degree two polynomial activations $\sigma(t) := at^2+bt+c$. Let $f : \mathbb{R}^d \to \mathbb{R}$ denote the neural network defined as

$$f(x) = \sum_{j=1}^{m} \sigma(x^T u_j)\alpha_j \tag{1}$$

where $x \in \mathbb{R}^d$ is the input sample, $u_j \in \mathbb{R}^d$ and $\alpha_j \in \mathbb{R}$ are the first and second layer weights, respectively. This is a fully connected neural network with m neurons in the hidden layer. We focus on the setting where the first dm weights (i.e., $u_j \in \mathbb{R}^d$, $j = 1, \ldots, m$) in the hidden layer are constrained to be integers.

The results are extended to neural networks with vector outputs, i.e., $f : \mathbb{R}^d \to \mathbb{R}^C$, in Section B of the Appendix.

2.1. Bilinear activation networks

Now we introduce a simpler architecture with bilinear activation $\mathcal{X} \to u^T \mathcal{X} v$ and binary quantization given by

$$f'(\mathcal{X}) = \sum_{j=1}^{m'} u_j^T \mathcal{X} v_j \alpha_j$$
$$\text{with } u_j, v_j \in \{-1, +1\}^d,\ \alpha_j \in \mathbb{R},\ \forall j \tag{2}$$

where $\mathcal{X} \in \mathbb{R}^{d\times d}$ is the lifted version of the input $x \in \mathbb{R}^d$ as will be defined in the sequel. We show that this architecture is sufficient to represent multi-level integer quantization and degree two polynomial activations without any loss of generality. In addition, these networks can be mapped to the standard network in (1) in a straightforward way as we formalize in this section. Hence, some of our results leverage the above architecture for training and transform a bilinear activation network into a polynomial activation network.

Theorem 1 (Reduction to binary quantization and bilinear activation)**.** *The following multi-level (i.e. $M + 1$ levels)*

quantized neural network

$$f(x) = \sum_{j=1}^{m} \sigma(x^T u_j)\alpha_j \quad \textit{where}$$

$$u_j \in \{-M, -M+2, \ldots, 0, \ldots, M-2, M\}^d, \alpha_j \in \mathbb{R}, \forall j$$

can be represented as a binary quantized bilinear activation network

$$f'(\mathcal{X}) = \sum_{j=1}^{m'} u_j^T \mathcal{X} v_j \alpha_j \textit{ where } u_j, v_j \in \{-1, +1\}^{dM+1},$$

$\mathcal{X} := \begin{bmatrix} a\tilde{x}\tilde{x}^T & \frac{b}{2}\tilde{x} \\ \frac{b}{2}\tilde{x}^T & c \end{bmatrix}$ *and* $\tilde{x} := x \otimes 1_M$. *Conversely, any binary quantized bilinear activation network* $f'(\mathcal{X})$ *of this form can be represented as a multi-level quantized neural network* $f(x)$.

In the remainder of this section, we provide a constructive proof of the above theorem by showing the reduction in three steps: Reducing to binary quantization, lifting and reducing to bilinear activation.

2.2. Reducing multi-level quantization to binary

In this section, we show that the two level binary quantization $\{-1, 1\}$ model is sufficient to model other quantization schemes with integer levels. Hence, we can focus on binary quantized neural network models without loss of generality. Suppose that q represents a hidden neuron quantized to $M+1$ levels given by

$$q \in \mathcal{Q}_M^d := \{-M, -M+2, \ldots, 0, \ldots, M-2, M\}^d . \tag{3}$$

Then we equivalently have

$$q^T x = \sum_{i=1}^{d} q_i x_i = \sum_{i=1}^{d} \sum_{k=1}^{M} u_{k+(i-1)M} x_i = u^T \tilde{x} , \tag{4}$$

where $\tilde{x} = x \otimes 1_M = [x_1, x_1, \ldots, x_2, x_2, \ldots,]^T \in \mathbb{R}^{dM}$ since $\sum_{k=1}^{M} u_{k+(i-1)M} \in \mathcal{Q}_M \, \forall i$. Therefore, stacking the input data x by replication as $\tilde{x} \in \mathbb{R}^{dM}$ enables $M+1$ level quantization to be represented as binary quantization.

2.3. Lifting dimensions

We first show that binary quantized networks with degree two polynomial activations are equivalent to binary quantized networks with quadratic activations. Note that the network output can be expressed as

$$\begin{aligned} f(x) &= \sum_{j=1}^{m} \big(a(x^T u_j)^2 + b(x^T u_j) + c\big)\alpha_j \\ &= \sum_{j=1}^{m} \tilde{u}_j^T \begin{bmatrix} axx^T & \frac{b}{2}x \\ \frac{b}{2}x^T & c \end{bmatrix} \tilde{u}_j \alpha_j \end{aligned} \tag{5}$$

where we defined the augmented weight vectors $\tilde{u}_j := [u_j^T, 1]^T$. Consequently, we can safely represent this via the relaxation $\tilde{u}_j \in \{-1, +1\}^{d+1}$ since $\tilde{u}_j^T \begin{bmatrix} axx^T & \frac{b}{2}x \\ \frac{b}{2}x^T & c \end{bmatrix} \tilde{u}_j = (-\tilde{u}_j)^T \begin{bmatrix} axx^T & \frac{b}{2}x \\ \frac{b}{2}x^T & c \end{bmatrix} (-\tilde{u}_j)$ and we can assume $(\tilde{u}_j)_{d+1} = 1$ without loss of generality.

2.4. Reduction to bilinear activation

Now we show that we can consider the network model

$$f(x) = \sum_{j=1}^{m} u_j^T \underbrace{\begin{bmatrix} axx^T & \frac{b}{2}x \\ \frac{b}{2}x^T & c \end{bmatrix}}_{\mathcal{X}} v_j \alpha_j = \sum_{j=1}^{m} u_j^T \mathcal{X} v_j \alpha_j \tag{6}$$

where $\{u_j, v_j\}_{j=1}^m$ are the model parameters to represent networks with quadratic activation without loss of generality. Using the symmetrization identity

$$2u^T A v = (u+v)^T A (u+v) - u^T A u - v^T A v , \tag{7}$$

we can express the neural network output as

$$\begin{aligned} &2f(x) = \\ &= \sum_{j=1}^{m} \Bigg((u_j + v_j)^T \begin{bmatrix} axx^T & \frac{b}{2}x \\ \frac{b}{2}x^T & c \end{bmatrix} (u_j + v_j)\alpha_j \\ &\quad - u_j^T \begin{bmatrix} axx^T & \frac{b}{2}x \\ \frac{b}{2}x^T & c \end{bmatrix} u_j \alpha_j - v_j^T \begin{bmatrix} axx^T & \frac{b}{2}x \\ \frac{b}{2}x^T & c \end{bmatrix} v_j \alpha_j \Bigg) . \end{aligned}$$

Note that $\frac{1}{2}(u_j + v_j) \in \{-1, 0, 1\}^d$ and the above can be written as a quantized network with quadratic activation and $3m$ hidden neurons.

3. Convex Duality of Quantized Neural Networks and SDP Relaxations

We consider the following non-convex training problem for the two-layer polynomial activation network

$$\begin{aligned} p^* = \min_{\text{s.t. } u_j \in \{-1,1\}^d, \alpha_j \in \mathbb{R}\; j \in [m]} \; & \ell\left(\sum_{j=1}^{m} \sigma(X u_j)\alpha_j, \, y \right) + \\ & + \beta d \sum_{j=1}^{m} |\alpha_j| . \end{aligned} \tag{8}$$

Here, $\ell(\cdot, y)$ is a convex and Lipschitz loss function, $\sigma(t) := at^2 + bt + c$ is a degree-two polynomial activation function, m is the number of neurons, β is the regularization parameter.

It is straightforward to show that this optimization problem is NP-hard even for the case when $m = 1$, $\sigma(t) = t$ is

the linear activation and $\ell(u, y) = (u - y)^2$ is the squared loss via a reduction to the MaxCut problem (Goemans & Williamson, 1995).

Note that we scale the regularization term by d to account for the fact that the hidden neurons have Euclidean norm $\sqrt{d}$, which simplifies the notation in the sequel. Taking the convex dual with respect to the second layer weights $\{\alpha_j\}_{j=1}^m$, the optimal value of the primal is lower bounded by

$$\begin{aligned} p^* \geq d^* &= \max_{|v^T \sigma(Xu)| \leq \beta d, \forall u \in \{-1,1\}^d} -\ell^*(-v) \\ &= \max_{\max_{u:u\in\{-1,1\}^d} |v^T \sigma(Xu)| \leq \beta d} -\ell^*(-v). \end{aligned} \tag{9}$$

Remarkably, the above dual problem is a convex program since the constraint set is an intersection of linear constraints. However, the number of linear constraints is exponential due to the binary quantization constraint.

We now describe an SDP relaxation which provides a lower-bounding and tractable dual convex program. Our formulation is inspired by the SDP relaxation of MaxCut (Goemans & Williamson, 1995), which is analogous to the constraint subproblem in (9). Let us assume that the activation is quadratic $\sigma(u) = u^2$, since we can reduce degree two polynomial activations to quadratics without loss of generality as shown in the previous section. Then, we have $|v^T \sigma(Xu)| = |u^T (\sum_{i=1}^n v_i x_i x_i^T) u|$.

The constraint $\max_{u:u_i^2=1,\forall i} |v^T (Xu)^2| \leq \beta d$ can be equivalently stated as the following two inequalities

$$\begin{aligned} q_1^* &= \max_{u:u_i^2=1,\forall i} u^T \left(\sum_{i=1}^n v_i x_i x_i^T \right) u \leq \beta d \,, \\ q_2^* &= \max_{u:u_i^2=1,\forall i} u^T \left(-\sum_{i=1}^n v_i x_i x_i^T \right) u \leq \beta d \,. \end{aligned} \tag{10}$$

The SDP relaxation for the maximization $\max_{u:u_i^2=1,\forall i} u^T \left(\sum_{i=1}^n v_i x_i x_i^T \right) u$ is given by

$$\hat{q}_1 = \max_{U \succeq 0,\, U_{ii}=1, \forall i} \operatorname{tr} \left(\sum_{i=1}^n v_i x_i x_i^T U \right) , \tag{11}$$

where $U \in \mathbb{S}^{d \times d}$. This is a relaxation since we removed the constraint $\operatorname{rank}(U) = 1$. Hence, the optimal value of the relaxation is an upper bound on the optimal solution, i.e., $\hat{q}_1 \geq q_1^*$. Consequently, the relaxation leads to the following lower bound:

$$d^* \geq \max_{q_1^* \leq \beta d,\, q_2^* \leq \beta d} -\ell^*(-v) \geq \max_{\hat{q}_1 \leq \beta d,\, \hat{q}_2 \leq \beta d} -\ell^*(-v) \,. \tag{12}$$

More precisely, we arrive at $d^* \geq d_{\text{SDP}}$ where

$$\begin{aligned} d_{\text{SDP}} := \max_{v} \; & -\ell^*(-v) \\ \text{s.t.} \; & \max_{U \succeq 0,\, U_{ii}=1, \forall i} \operatorname{tr} \left(\sum_{i=1}^n v_i x_i x_i^T U \right) \leq \beta d \\ & \max_{U \succeq 0,\, U_{ii}=1, \forall i} \operatorname{tr} \left(-\sum_{i=1}^n v_i x_i x_i^T U \right) \leq \beta d \,. \end{aligned} \tag{13}$$

The dual of the SDP in the constraint (11) is given by the dual of the MaxCut SDP relaxation, which can be stated as

$$\begin{aligned} \min_{z \in \mathbb{R}^d} \quad & d \cdot \lambda_{\max} \left(\sum_{i=1}^n v_i x_i x_i^T + \operatorname{diag}(z) \right) \\ \text{s.t.} \quad & \bar{1}^T z = 0 \,. \end{aligned} \tag{14}$$

Since the primal problem is strictly feasible, it follows from Slater's condition that the strong duality holds between the primal SDP and the dual SDP. This allows us to reformulate the problem in (13) as

$$\begin{aligned} \max_{v, z_1, z_2} \quad & -\ell^*(-v) \\ \text{s.t.} \quad & \lambda_{\max} \left(\sum_{i=1}^n v_i x_i x_i^T + \operatorname{diag}(z_1) \right) \leq \beta \\ & \lambda_{\max} \left(-\sum_{i=1}^n v_i x_i x_i^T + \operatorname{diag}(z_2) \right) \leq \beta \\ & \bar{1}^T z_1 = 0, \; \bar{1}^T z_2 = 0 \,, \end{aligned} \tag{15}$$

where the variables have dimensions $v \in \mathbb{R}^n$, $z_1, z_2 \in \mathbb{R}^d$ and $\lambda_{\max}$ denotes the largest eigenvalue. Expressing the largest eigenvalue constraints as linear matrix inequalities yields

$$\begin{aligned} d_{\text{SDP}} := \max_{v, z_1, z_2} \quad & -\ell^*(-v) \\ \text{s.t.} \quad & \sum_{i=1}^n v_i x_i x_i^T + \operatorname{diag}(z_1) - \beta I_d \preceq 0 \\ & -\sum_{i=1}^n v_i x_i x_i^T + \operatorname{diag}(z_2) - \beta I_d \preceq 0 \\ & \bar{1}^T z_1 = 0, \; \bar{1}^T z_2 = 0 \,. \end{aligned} \tag{16}$$

Next, we find the dual optimization problem. First we ex-

press the Lagrangian:

$$L(v, z_1, z_2, Z_1, Z_2, \rho_1, \rho_2) = \\ = -\ell^*(-v) - \sum_{i=1}^{n} v_i x_i^T (Z_1 - Z_2) x_i + \beta \operatorname{tr}(Z_1 + Z_2) \\ - \sum_{j=1}^{d} (Z_{1,jj} z_{1,j} + Z_{2,jj} z_{2,j}) + \sum_{j=1}^{d} (\rho_1 z_{1,j} + \rho_2 z_{2,j}) \tag{17}$$

where $Z_1, Z_2 \in \mathbb{S}^{d\times d}$ and $\rho_1, \rho_2 \in \mathbb{R}$ are the Lagrange multipliers. Maximizing the Lagrangian with respect to v, z_1, z_2 leads to the following convex program

$$\begin{aligned} \min_{Z_1, Z_2, \rho_1, \rho_2} \quad & \ell\left(\begin{bmatrix} x_1^T (Z_1 - Z_2) x_1 \\ \vdots \\ x_n^T (Z_1 - Z_2) x_n \end{bmatrix}, y\right) + \beta \operatorname{tr}(Z_1 + Z_2) \\ \text{s.t.} \quad & Z_{1,jj} = \rho_1,\ Z_{2,jj} = \rho_2,\ j = 1, \dots, d \\ & Z_1 \succeq 0,\ Z_2 \succeq 0\,. \end{aligned} \tag{18}$$

Finally, we obtain $p^* \geq d_{\text{SDP}}$ where

$$\begin{aligned} d_{\text{SDP}} := \min_{Z_1, Z_2, \rho_1, \rho_2} \quad & \ell(\hat{y}, y) + \beta d(\rho_1 + \rho_2) \\ \text{s.t.} \quad & \hat{y}_i = x_i^T (Z_1 - Z_2) x_i,\ i = 1, \dots, n \\ & Z_{1,jj} = \rho_1,\ Z_{2,jj} = \rho_2,\ j = 1, \dots, d \\ & Z_1 \succeq 0,\ Z_2 \succeq 0\,. \end{aligned} \tag{19}$$

Remarkably, the above SDP is a polynomial time tractable lower bound for the combinatorial non-convex problem p^*.

3.1. SDP relaxation for bilinear activation networks

Now we focus on the bilinear architecture $f(x) = \sum_{j=1}^{m} (x^T u_j)(x^T v_j)\alpha_j$ and provide an SDP relaxation for the corresponding non-convex training problem. It will be shown that the resulting SDP relaxation is provably tight, where a matching upper bound can be obtained via randomization. Moreover, the resulting feasible solutions can be transformed into a quantized neural network with polynomial activations as we have shown in Section 2. Consider the non-convex training problem for the two-layer network with the bilinear activation given by

$$p_b^* = \min_{\text{s.t. } u_j, v_j \in \{-1,1\}^d, \alpha_j \in \mathbb{R}\ \forall j \in [m]} g\left(\{u_j, v_j, \alpha_j\}_{j=1}^m\right) \tag{20}$$

where

$$g\left(\{u_j, v_j, \alpha_j\}_{j=1}^m\right) = \\ = \ell\left(\sum_{j=1}^{m} ((Xu_j) \circ (Xv_j))\alpha_j,\ y\right) + \beta d \sum_{j=1}^{m} |\alpha_j|\,. \tag{21}$$

Here $\circ$ denotes the Hadamard, i.e., direct product of two vectors. Repeating an analogous duality analysis (see Section A.3 for the details), we obtain a tractable lower-bounding problem given by

$$\begin{aligned} p_b^* \geq d_{\text{bSDP}} := \min_{Q, \rho} \quad & \ell(\hat{y},\ y) + \beta d \rho \\ \text{s.t.} \quad & \hat{y}_i = 2 x_i^T Z x_i,\ i = 1, \dots, n \\ & Q_{jj} = \rho,\ j = 1, \dots, 2d \\ & Q = \begin{bmatrix} V & Z \\ Z^T & W \end{bmatrix} \succeq 0\,. \end{aligned} \tag{22}$$

The above optimization problem is a convex SDP, which can be solved efficiently in polynomial time.

4. Main result: SDP Relaxation is Tight

We now introduce an existence result on covariance matrices which will be used in our quantized neural network construction.

Theorem 2 (Trigonometric covariance shaping)**.** *Suppose that $Z^* \in \mathbb{R}^{d\times d}$ is an arbitrary matrix such that $\exists V, W$: $\begin{bmatrix} V & Z^* \\ Z^{*T} & W \end{bmatrix} \succeq 0$ and $V_{jj} = W_{jj} = 1\ \forall j$. Then, there exists a PSD matrix $Q \in \mathbb{R}^{2d\times 2d} \succeq 0$ satisfying $Q_{jj} = 1\ \forall j$ and*

$$\arcsin(Q_{(12)}) = \gamma Z^* \tag{23}$$

where $Q = \begin{bmatrix} Q_{(11)} & Q_{(12)} \\ Q_{(21)} & Q_{(22)} \end{bmatrix}$, $\gamma = \ln(1 + \sqrt{2})$, and $\arcsin$ is the elementwise inverse sine function.

Our construction is based on randomly generating quantized neurons whose empirical covariance matrix is matched to the optimal solution of the convex SDP. The above theorem is an existence result which will be crucial in our sampling algorithm. The important observation is that, if we let $\begin{bmatrix} u \\ v \end{bmatrix} \sim \operatorname{sign}(\mathcal{N}(0, Q))$ with some $Q \succeq 0$, $Q_{jj} = 1\ \forall j$, then $\mathbb{E}\begin{bmatrix} u \\ v \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}^T = \frac{2}{\pi}\arcsin(Q)$, which is referred to as Grothendieck's Identity (Alon & Naor, 2004). Therefore, $\mathbb{E}[uv^T] = \arcsin Q_{(12)} = \gamma Z^*$, which is proportional to the target covariance matrix. This algorithm is inspired by Krivine's analysis of the Grothendieck's constant and its applications in approximating the cut norm using semidefinite programming (Alon & Naor, 2004).

Proof of Theorem 2. Note that the condition $\exists V, W$: $\begin{bmatrix} V & Z^* \\ Z^{*T} & W \end{bmatrix} \succeq 0$ and $V_{jj} = W_{jj} = 1\ \forall j$ implies that there exists unit norm vectors $x_1, \dots, x_d, y_1, \dots, y_d$ such that $Z_{ij}^* = x_i^T y_j$. Consequently, applying Lemma 4.2 of (Alon & Naor, 2004) and Grothendieck's Identity completes the proof. □

Algorithm 1 Sampling algorithm for quantized neural networks

1. Solve the SDP in (22). Define the scaled matrix $Z_s^* \leftarrow Z^*/\rho^*$.
2. Solve the problem
$$Q^* := \arg \min_{Q \succeq 0, Q_{jj}=1 \forall j} \|Q_{(12)} - \sin(\gamma Z_s^*)\|_F^2 . \tag{24}$$
3. Sample the first layer weights $u_1, \ldots, u_m, v_1, \ldots, v_m$ from multivariate normal distribution as $\begin{bmatrix} u \\ v \end{bmatrix} \sim \text{sign}(\mathcal{N}(0, Q^*))$ and set the second layer weights as $\alpha_j = \rho^* \frac{\pi}{\gamma m}$, $\forall j$.
4. (optional) Transform the quantized bilinear activation network to a quantized polynomial activation network.

4.1. Sampling algorithm for approaching the global optimum

Now we present our sampling algorithm which generates quantized neural networks parameters based on the solution of the lower-bounding convex SDP. The algorithm is listed in Algorithm 1. We explain each step of the algorithm below.

1. Solve the SDP in (22) to minimize the training loss. Denote the optimal solution as Z^* and ρ^* and define the scaled matrix $Z_s^* \leftarrow Z^*/\rho^*$.
2. Find the $2d \times 2d$ covariance matrix Q^* by solving (24) with $Q = \begin{bmatrix} Q_{(11)} & Q_{(12)} \\ Q_{(21)} & Q_{(22)} \end{bmatrix}$ where the notation $Q_{(ij)}$ denotes a $d \times d$ block matrix. $\gamma = \ln(1+\sqrt{2})$, and $\sin(\cdot)$ is the element-wise sine function. The objective value is guaranteed to be zero due to Theorem 2. Therefore we have $\arcsin(Q_{(12)}^*) = \gamma Z_s^*$ and $Q^* \succeq 0, Q_{jj}^* = 1 \, \forall j$.
3. Sample $u_1, \ldots, u_m, v_1, \ldots, v_m$ via $\begin{bmatrix} u \\ v \end{bmatrix} \sim \text{sign}(\mathcal{N}(0, Q^*))$. Since $\mathbb{E}[uv^T] = \frac{2}{\pi} \arcsin Q_{(12)}^* = \frac{2\gamma}{\pi} Z_s^*$ as a corollary of Theorem 2, we have $\mathbb{E}[\frac{1}{m} \sum_{j=1}^m u_j v_j^T] = \frac{2\gamma}{\pi} Z_s^*$. We set $\alpha_j = \rho^* \frac{\pi}{\gamma m}$, $\forall j$ to obtain $\mathbb{E}[\sum_{j=1}^m u_j v_j^T \alpha_j] = 2 Z_s^* \rho^* = 2 Z^*$. This is as desired since the SDP computes the predictions via $\hat{y}_i = 2 x_i^T Z x_i$.
4. This optional step can be performed as described in Section 2.

The extension of the sampling algorithm to the vector output networks is given in Section B.

4.2. Concentration around the mean

We establish a probabilistic bound on the convergence of the empirical sum $\frac{1}{m} \sum_{j=1}^m u_j v_j^T$ in the step 3 of the sampling algorithm to its expectation. Our technique involves applying Matrix Bernstein concentration bound for sums of i.i.d. rectangular matrices (Tropp, 2015) to obtain:

$$\mathbb{P}\left[\left\| \frac{1}{m} \sum_{j=1}^m u_j v_j^T - \mathbb{E}[u_1 v_1^T] \right\|_2 \geq \epsilon \right] \leq \exp\left(- \frac{m\epsilon^2}{(2\gamma/\pi)^2 \|Z_s^*\|_2^2 + d(c' + 2\epsilon/3)} + \log(2d) \right) \tag{25}$$

for all $\epsilon > 0$.

We summarize this analysis in the following theorem, which is our main result.

Theorem 3 (Main result)**.** *Suppose that the number of neurons satisfies $m \geq c_1 \frac{L_c^2 R_m^4 d \log(d)}{\epsilon^2}$. Let L_c denote the Lipschitz constant of the vectorized loss function under the ℓ-infinity norm, i.e. $|\ell(z) - \ell(z')| \leq L_c \|z - z'\|_\infty$, and define $R_m := \max_{i \in [n]} \|x_i\|_2$. Then, Algorithm 1 generates a quantized neural network with weights $\hat{u}_j, \hat{v}_j \in \{-1, +1\}^d$ and $\hat{\alpha}_j = \frac{\rho^* \pi}{m \log(1+\sqrt{2})}$, $j = 1, \ldots, m$ that achieve near optimal loss, i.e.,*

$$\Big| \ell\Big(\sum_{j=1}^m ((X\hat{u}_j) \circ (X\hat{v}_j)) \hat{\alpha}_j, \, y \Big) - \ell\Big(\sum_{j=1}^m ((X u_j^*) \circ (X v_j^*)) \alpha_j^*, \, y \Big) \Big| \leq \epsilon \tag{26}$$

with probability at least $1 - c_2 e^{-c_3 \epsilon^2 m/d}$ for certain constants c_1, c_2, c_3 when the regularization coefficient satisfies $\beta \leq \frac{\epsilon}{d} \min\left(\frac{1}{\sum_j |\hat{\alpha}_j|}, \frac{1}{\sum_j |\alpha_j^*|} \right)$. The weights $u_j^*, v_j^* \in \{-1, +1\}^d, \alpha_j^* \in \mathbb{R}$, $j = 1, \ldots, m$ are the optimal network parameters for the non-convex combinatorial problem in (21).

Remark 1. *For loss functions of the form $\ell(z) = \frac{1}{n} \sum_{i=1}^n \phi(z_i)$, where $\phi(\cdot)$ is a scalar L_c-Lipschitz loss satisfying $|\phi(s) - \phi(s')| \leq L_c |s - s'|$, the vectorized loss function $\ell(z)$ is L_c-Lipschitz under the infinity norm. This fact follows from $\left| \frac{1}{n} \sum_{i=1}^n \phi(z_i) - \frac{1}{n} \sum_{i=1}^n \phi(z_i') \right| \leq \frac{1}{n} \sum_{i=1}^n L_c |z_i - z_i'| \leq L_c \|z - z'\|_\infty$. Examples of 1-Lipschitz loss functions include hinge loss, logistic loss and ℓ_1 loss, which satisfy our assumption with $L_c = 1$.*

Remark 2. *Our main result also holds when $\beta \to 0$. In this regime, the constraint $\beta \leq \frac{\epsilon}{d} \min\left(\frac{1}{\sum_j |\hat{\alpha}_j|}, \frac{1}{\sum_j |\alpha_j^*|} \right)$ is always satisfied.*

The proof of Theorem 3 is provided in Section A.2. To the best of our knowledge, this is the first result on polynomial-time optimal trainability of quantized neural networks. We remark that one can transform the near optimal quantized bilinear activation network to a near optimal quantized polynomial activation network with the mapping shown in Section 2. Consequently, this result also applies to approximating the solution of (8).

Additionally, note that the second layer weights are all identical, which allows us to represent the sampled neural network using $2md$ bits and only one scalar floating point variable. One can employ the reduction in Section 2.2 to train optimal multi-level quantized neural networks using the above result in polynomial time.

Furthermore, it is interesting to note that, overparameterization is a key component in enabling optimization over the combinatorial search space of quantized neural networks in polynomial time. In contrast, the problems in (21) and (8) are NP-hard when $m = 1$.

5. Numerical Results

In this section, we present numerical results that verify our theoretical findings. Additional numerical results can be found in the Appendix.

We compare the performance of the proposed SDP based method against a backpropagation based method that we describe in the next subsection. We have used CVXPY (Diamond & Boyd, 2016; Agrawal et al., 2018) for solving the convex SDP. In particular, we have used the open source solver SCS (splitting conic solver) (O'Donoghue et al., 2016; 2019) in CVXPY, which is a scalable first order solver for convex cone problems. Furthermore, in solving the non-convex neural network training problems that we include for comparison, we have used the stochastic gradient descent (SGD) algorithm with momentum in PyTorch (Paszke et al., 2019).

The experiments have been carried out on a MacBook with 2.2 GHz 6-Core Intel Core i7 processor and 16 GB of RAM.

5.1. Planted dataset experiment

Figure 1 shows the cost as a function of the number of neurons m. The neural network architecture is a two-layer fully connected network with bilinear activation, i.e., $f(x) = \sum_{j=1}^{m}(x^T u_j)(x^T v_j)\alpha_j$. This experiment has been done using a planted dataset. The plot compares the method described in Section 4 against a backpropagation based quantization method.

The algorithm in Section 4 solves the relaxed SDP and then samples binary weights as described previously. This procedure results in $2md$ binary weights for the first layer.

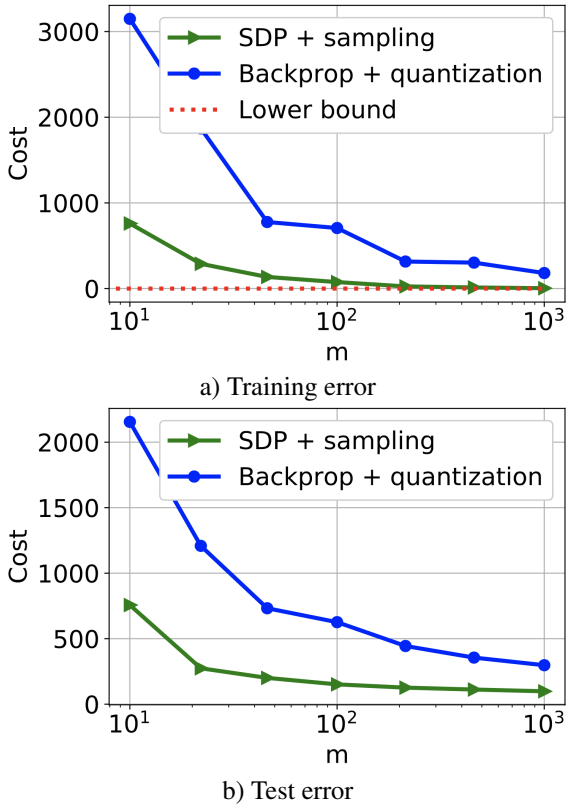


Figure 1. Objective against the number of neurons m. Dataset X has been synthetically generated via sampling from standard Gaussian distribution and has dimensions $n = 100$, $d = 20$. The target vector y has been computed via a planted model with 10 planted neurons. In the planted model, the first layer weights are binary and the second layer weights are real and non-negative. The regularization coefficient is $\beta = 10^{-4}$. The lower bound is obtained by solving the SDP in Section 3. Plots a and b show the cost on the training and test sets, respectively. The test set has been generated synthetically by sampling from the same distribution as the training set.

The second layer weights are all equal to $\rho^*\pi/(\gamma m)$. This network requires storage of $2md$ bits and a single real number. Furthermore, post-training quantization using backpropagation works as follows. First, we train a two-layer neural network with bilinear activation in PyTorch (Paszke et al., 2019) with m neurons using stochastic gradient descent (SGD). We fix the second layer weights to $1/m$ during training. After training, we form the matrices $\hat{Z} = \sum_{j=1}^{m} \text{sign}(u_j)\,\text{sign}(v_j^T)$ and $Z^* = \sum_{j=1}^{m} u_j v_j^T \frac{1}{m}$. Then, the solution of the problem $\min_{c\in\mathbb{R}} \|c\hat{Z} - Z^*\|_F^2$ is used to determine the second layer weights as c. The optimal solution is given by $c = \frac{\langle \hat{Z}, Z^* \rangle}{\langle Z^*, Z^* \rangle}$. This procedure results in $2md$ bits for the first layer and a single real number for the second layer and hence requires the same amount of storage as the SDP based method.

In addition to low storage requirements, this particular network is very efficient in terms of computation. This is very critical for many machine learning applications as this translates to shorter inference times. For the two-layer neural network with bilinear activation, the hidden layer computations are $2md$ additions since the weights are $\{+1, -1\}$ and the bilinear activation layer performs m multiplications (i.e. $(x^T u_j)(x^T v_j)$ $j = 1, \ldots, m$). The second layer requires only m additions and one multiplication since the second layer weights are the same.

Figure 1 shows that the SDP based method outperforms the backpropagation approach. Also, we observe that the cost of the SDP based method approaches the lower bound rapidly as the number of neurons m is increased. Furthermore, plot b shows that the test set performance for the SDP based method is also superior to the backpropagation based method.

We note that another advantage of the SDP based sampling method over backpropagation is that we do not need to solve the SDP for a fixed number of neurons m. That is, the SDP does not require the number of neurons m as an input. The number of neurons is used only during the sampling process. This enables one to experiment with multiple values for the number of neurons without re-solving the SDP.

5.2. Real dataset experiment

Figure 2 compares the backpropagation approach and the SDP based method on a real dataset from UCI machine learning repository (Dua & Graff, 2017). The dataset is the binary classification "breast-cancer" dataset and has $n = 228$ training samples and 58 test samples and the samples are $d = 9$ dimensional. Figure 2 shows the classification accuracy against time for various methods which we describe below. The regularization coefficient β is picked for each method separately by searching the value that yields the highest accuracy and the resulting β values are provided in the captions of the figures.

Figure 2 shows the training and test accuracy curves for backpropagation without quantization by the blue solid curve. After the convergence of the backpropagation, we quantize the weights as described in the previous subsection, and the timing and accuracy for the quantized model are indicated by the cyan cross marker. The timing and accuracy of the SDP based method are shown using the red cross marker. Figure 2 demonstrates that the SDP based method requires less time to return its output. We observe that quantization reduces the accuracy of backpropagation to a lower accuracy than the SDP based method's accuracy.

It is important to note that in neural network training, since the optimization problems are non-convex, it takes considerable effort and computation time to determine the hyperparameters that will achieve convergence and good performance. For instance, among the hyperparameters that require tuning is the learning rate (i.e. step size). We have performed the learning rate tuning for the backpropagation algorithm offline and hence it is not reflected in Figure 2. Remarkably, the proposed convex SDP based method does not require this step as it is readily handled by the convex SDP solver.

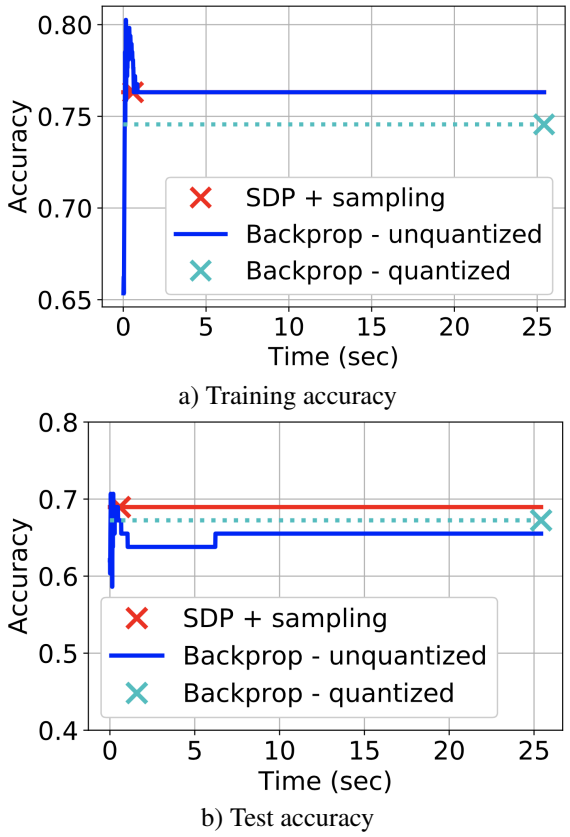
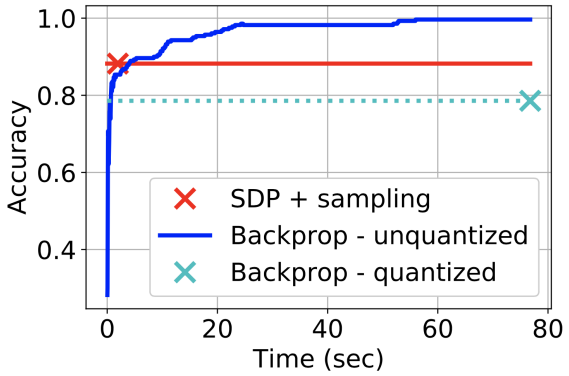
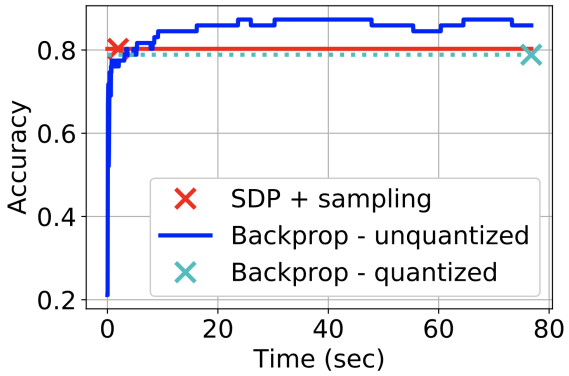


Figure 2. Classification accuracy against wall-clock time. Breast cancer dataset with $n = 228, d = 9$. The number of neurons is $m = 250$ and the regularization coefficient is $\beta = 0.1$ for the SDP based method and $\beta = 0.1$ for the backpropagation.

Figure 3 shows results for the UCI repository dataset "ionosphere". This is a binary classification dataset with $n = 280$ training samples and 71 test samples. The samples are $d = 33$ dimensional. The experiment setting is similar to Figure 2 with the main difference that the number of neurons is 10 times higher (i.e., $m = 2500$). We observe that the SDP based method outperforms the quantized network trained with backpropagation on both training and test sets.



a) Training accuracy



b) Test accuracy

Figure 3. Classification accuracy against wall-clock time. Ionosphere dataset with $n = 280, d = 33$. The number of neurons is $m = 2500$ and the regularization coefficient is $\beta = 10$ for the SDP based method, $\beta = 10^{-6}$ for backpropagation.

6. Conclusion

We introduced a convex duality based approach for training optimal quantized neural networks with degree two polynomial activations. We first presented a lower-bounding semidefinite program which is tractable in polynomial time. We also introduced a bilinear activation architecture, and the corresponding SDP lower-bound. We showed that bilinear architectures with binary quantization are sufficient to train optimal multi-level quantized networks with polynomial activations. We presented a sampling algorithm to generate quantized neural networks using the SDP by leveraging Grothendieck's identity and the connection to approximating the cut norm. Remarkably, we showed that mild overparameterization is sufficient to obtain a near-optimal quantized neural network via the SDP based sampling approach. Numerical experiments show that our method can generate significantly more accurate quantized neural networks compared to the standard post-training quantization approach. Moreover, the convex optimization solvers are faster than backpropagation in small to medium scale problems.

An immediate open question is to extend our results to deeper networks and different architectures, such as ReLU networks. For instance, our algorithm can be applied with polynomial approximations of ReLU. Moreover, one can apply our algorithm layerwise to optimally quantize a pretrained neural network by knowledge distillation.

We acknowledge that our current numerical results are limited to small and medium datasets due to the memory constraints of standard SDP solvers. However, one can design custom optimization methods to obtain approximate solutions of the SDP for larger dimensional instances. The SDPs can also be defined and solved in deep learning frameworks with appropriate parameterizations. Random projection and sketching based optimizers for high-dimensional convex programs (Yurtsever et al., 2021; 2017; Lacotte & Pilanci, 2020a) and randomized preconditioning (Lacotte et al., 2020; Lacotte & Pilanci, 2020d;c; Ozaslan et al., 2019; Lacotte & Pilanci, 2021) can address these computational challenges. We leave this as an important open problem.

From a complexity theoretical perspective, it is remarkable that overparameterization breaks computational barriers in combinatorial and non-convex optimization. Specifically, it is straightforward to show that training a quantized neural network when $m = 1$, i.e., a single neuron is NP-hard due to the connection to the MaxCut problem. However, allowing $m = \mathcal{O}(d \log d)$ enables optimization over a combinatorial search space in polynomial time. Exploring the other instances and limits of this phenomenon is another interesting research direction.

Acknowledgments

This work was partially supported by the National Science Foundation under grants IIS-1838179 and ECCS-2037304, Facebook Research, Adobe Research and Stanford SystemX Alliance.

References

Agrawal, A., Verschueren, R., Diamond, S., and Boyd, S. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

Allen-Zhu, Z. and Li, Y. Backward feature correction: How deep learning performs deep learning. *arXiv preprint arXiv:2001.04413*, 2020.

Alon, N. and Naor, A. Approximating the cut-norm via grothendieck's inequality. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pp. 72–80, 2004.

Anwar, S., Hwang, K., and Sung, W. Fixed point opti-

mization of deep convolutional neural networks for object recognition. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1131–1135, April 2015. doi: 10.1109/ICASSP.2015.7178146.

Bach, F. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

Bartan, B. and Pilanci, M. Convex relaxations of convolutional neural nets. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4928–4932. IEEE, 2019.

Bartan, B. and Pilanci, M. Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time. *arXiv preprint arXiv:2101.02429*, 2021.

Bengio, Y., Le Roux, N., Vincent, P., Delalleau, O., and Marcotte, P. Convex neural networks. *Advances in neural information processing systems*, 18:123, 2006.

Diamond, S. and Boyd, S. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Ergen, T. and Pilanci, M. Convex optimization for shallow neural networks. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 79–83, 2019.

Ergen, T. and Pilanci, M. Convex geometry of two-layer relu networks: Implicit autoencoding and interpretable models. In *International Conference on Artificial Intelligence and Statistics*, pp. 4024–4033. PMLR, 2020a.

Ergen, T. and Pilanci, M. Revealing the structure of deep neural networks via convex duality. *arXiv preprint arXiv:2002.09773*, 2020b.

Ergen, T. and Pilanci, M. Implicit convex regularizers of cnn archi-tectures: Convex optimization of two-and three-layer networks in polynomial time. *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:2006.14798*, 2021.

Ergen, T., Sahiner, A., Ozturkler, B., Pauly, J., Mardani, M., and Pilanci, M. Demystifying batch normalization in relu networks: Equivalent convex optimization models and implicit regularization. *arXiv preprint arXiv:2103.01499*, 2021.

Gilad-Bachrach, R., Dowlin, N., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *International Conference on Machine Learning*, pp. 201–210, 2016.

Goemans, M. X. and Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42:1115–1145, 1995.

Gong, Y., Liu, L., Yang, M., and Bourdev, L. D. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014. URL http://arxiv.org/abs/1412.6115.

Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. Deep learning with limited numerical precision. *CoRR*, abs/1502.02551, 2015. URL http://arxiv.org/abs/1502.02551.

Gupta, V., Bartan, B., Ergen, T., and Pilanci, M. Exact and relaxed convex formulations for shallow neural autoregressive models. *International Conference on Acoustics, Speech, and Signal Processing*, 2021.

Han, S., Mao, H., and Dally, W. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. URL http://arxiv.org/abs/1510.00149.

Hwang, K. and Sung, W. Fixed-point feedforward deep neural network design using weights +1, 0, and -1. In *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, pp. 1–6, Oct 2014. doi: 10.1109/SiPS.2014.6986082.

Lacotte, J. and Pilanci, M. Adaptive and oblivious randomized subspace methods for high-dimensional optimization: Sharp analysis and lower bounds. *arXiv preprint arXiv:2012.07054*, 2020a.

Lacotte, J. and Pilanci, M. All local minima are global for two-layer relu neural networks: The hidden convex optimization landscape. *arXiv preprint arXiv:2006.05900*, 2020b.

Lacotte, J. and Pilanci, M. Effective dimension adaptive sketching methods for faster regularized least-squares optimization. *arXiv preprint arXiv:2006.05874*, 2020c.

Lacotte, J. and Pilanci, M. Optimal randomized first-order methods for least-squares problems. In *International Conference on Machine Learning*, pp. 5587–5597. PMLR, 2020d.

Lacotte, J. and Pilanci, M. Fast convex quadratic optimization solvers with adaptive sketching-based preconditioners. *arXiv preprint arXiv:2104.14101*, 2021.

Lacotte, J., Liu, S., Dobriban, E., and Pilanci, M. Limiting spectrum of randomized hadamard transform and optimal iterative sketching methods. *arXiv preprint arXiv:2002.00864*, 2020.

Lin, D. D. and Talathi, S. S. Overcoming challenges in fixed point training of deep convolutional networks. *CoRR*, abs/1607.02241, 2016. URL http://arxiv.org/abs/1607.02241.

Lin, D. D., Talathi, S. S., and Annapureddy, V. S. Fixed point quantization of deep convolutional networks. *CoRR*, abs/1511.06393, 2015. URL http://arxiv.org/abs/1511.06393.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, June 2016. URL http://stanford.edu/~boyd/papers/scs.html.

O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S. SCS: Splitting conic solver, version 2.1.2. https://github.com/cvxgrp/scs, November 2019.

Ozaslan, I. K., Pilanci, M., and Arikan, O. Iterative hessian sketch with momentum. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7470–7474. IEEE, 2019.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, 2019.

Pilanci, M. and Ergen, T. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*, pp. 7695–7705. PMLR, 2020.

Sahiner, A., Ergen, T., Pauly, J., and Pilanci, M. Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms. *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:2012.13329*, 2021a.

Sahiner, A., Mardani, M., Ozturkler, B., Pilanci, M., and Pauly, J. Convex regularization behind neural reconstruction. *International Conference on Learning Representations (ICLR), arXiv preprint arXiv:2012.05169*, 2021b.

Tropp, J. A. An introduction to matrix concentration inequalities. *arXiv preprint arXiv:1501.01571*, 2015.

Yurtsever, A., Udell, M., Tropp, J., and Cevher, V. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *Artificial intelligence and statistics*, pp. 1188–1196. PMLR, 2017.

Yurtsever, A., Tropp, J. A., Fercoq, O., Udell, M., and Cevher, V. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

Zhu, C., Han, S., Mao, H., and Dally, W. J. Trained ternary quantization. *CoRR*, abs/1612.01064, 2016. URL http://arxiv.org/abs/1612.01064.