LIPSCHITZ CONSTANT ESTIMATION FOR NEURAL NETWORKS VIA SPARSE POLYNOMIAL OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce LiPopt, a polynomial optimization framework for computing increasingly tighter upper bounds on the Lipschitz constant of neural networks. The underlying optimization problems boil down to either linear (LP) or semidefinite (SDP) programming. We show how to use structural properties of the network, such as sparsity, to significantly reduce the complexity of computation. This is specially useful for convolutional as well as pruned neural networks. We conduct experiments on networks with random weights as well as networks trained on MNIST, showing that in the particular case of the ℓ_∞-Lipschitz constant, our approach yields superior estimates as compared to other baselines available in the literature.

1 INTRODUCTION

We consider a neural network f_d defined by the recursion:

$$f_1(x) := W_1 x \qquad f_i(x) := W_i \sigma(f_{i-1}(x)), \quad i = 2, \dots, d \tag{1}$$

for an integer d larger than 1, matrices $\{W_i\}_{i=1}^d$ of appropriate dimensions and an *activation function* σ, understood to be applied element-wise. We refer to d as the *depth*, and we we focus on the case where f_d has a single real value as output.

In this work, we address the problem of estimating the *Lipschitz constant* of the network f_d. A function f is *Lipschitz continuous* with respect to a norm $\|\cdot\|$ if there exists a constant L such that for all x, y we have $|f(x) - f(y)| \leq L\|x - y\|$. The minimum over all such values satisfying this condition is called the *Lipschitz constant* of f and is denoted by $L(f)$.

The Lipschitz constant of a neural network is of major importance in many successful applications of *deep learning*. In the context of supervised learning, Bartlett et al. (2017) show how it directly correlates with the generalization ability of neural network classifiers, suggesting it as model complexity measure. It also provides a measure of robustness against adversarial perturbations (Szegedy et al., 2014) and can be used to improve such metric (Cisse et al., 2017). Moreover, an upper bound on $L(f_d)$ provides a certificate of robust classification around data points (Weng et al., 2018).

Another example is the discriminator network of the *Wasserstein GAN* (Arjovsky et al., 2017), whose Lipschitz constant is constrained to be at most 1. To handle this constraint, researchers have proposed different methods like heuristic penalties (Gulrajani et al., 2017), upper bounds (Miyato et al., 2018), choice of activation function (Anil et al., 2019), among many others. This line of work has shown that accurate estimation of such constant is key to generating high quality images.

Lower bounds or heuristic estimates of $L(f_d)$ can be used to provide a general sense of how robust a network is, but fail to provide true certificates of robustness to input perturbations. Such certificates require true upper bounds, and are paramount when deploying safety-critical deep reinforcement learning applications (Berkenkamp et al., 2017; Jin & Lavaei, 2018). The trivial upper bound given by the product of layer-wise Lipschitz constants is easy to compute but rather loose and overly pessimistic, providing poor insight into the true robustness of a network (Huster et al., 2018).

Indeed, there is a growing need for methods that provide tighter upper bounds on $L(f_d)$, even at the expense of increased complexity. For example Raghunathan et al. (2018a); Jin & Lavaei (2018); Fazlyab et al. (2019) derive upper bounds based on *semidefinite programming (SDP)*. While expensive to compute, these type of certificates are in practice surprisingly tight. Our work belongs in this vein of research, and aims to overcome some limitations in the current state-of-the-art.

Our Contributions.

- ▷ We present **LiPopt**, a general approach for upper bounding the Lipschitz constant of a neural network based on a relaxation to a *polynomial optimization problem (POP)* (Lasserre, 2015). This approach requires only that the unit ball be described with polynomial inequalities, which covers the common ℓ_2- and ℓ_∞-norms.
- ▷ Based on a theorem due to Weisser et al. (2018), we exploit the sparse connectivity of neural network architectures to derive a sequence of linear programs (LPs) of considerably smaller size than their vanilla counterparts. We provide an asymptotic analysis of the size of such programs, in terms of the number of neurons, depth and sparsity of the network.
- ▷ Focusing on the ℓ_∞-norm, we experiment on networks with random weights and networks trained on MNIST (Lecun et al., 1998). We evaluate different configurations of depth, width and sparsity and we show that the proposed sequence of LPs can provide tighter upper bounds on $L(f_d)$ compared to other baselines available in the literature.

Notation. We denote by n_i the number of columns of the matrix W_i in the definition (1) of the network. This corresponds to the size of the i-th layer, where we identify the input as the first layer. We let $n = n_1 + \ldots + n_d$ be the total number of neurons in the network. For a vector x, $\text{Diag}(x)$ denotes the square matrix with x in its diagonal and zeros everywhere else. For an array X, $\text{vec}(X)$ is the *flattened* array. The support of a sequence $\text{supp}(\alpha)$ is defined as the set of indices j such that α_j is nonzero. For $x = [x_1, \ldots, x_n]$ and a sequence of nonnegative integers $\gamma = [\gamma_1, \ldots, \gamma_n]$ we denote by x^γ the monomial $x_1^{\gamma_1} x_2^{\gamma_2} \ldots x_n^{\gamma_n}$. The set of nonnegative integers is denoted by $\mathbb{N}$.

Remark. The definition of network (1) covers typical architectures composed of dense and convolutional layers. In general, our proposed approach can be readily extended with minor modifications to any directed acyclic computation graph e.g., residual network architectures (He et al., 2016).

Structure of the paper. Section 2 describes an upper bound on $L(f_d)$ as a POP. Section 3 presents the general approach of **LiPopt**, and a particular instance providing a hierarchy of LPs solving the POP to arbitrary accuracy. Section 4 describes how to exploit the sparsity of the network to obtain smaller LPs, and contains the schematics of **LiPopt** (1). Section 5 describes a solution via a *quadratically constrained quadratic program (QCQP)* reformulation. In Section 6 we discuss related work. Section 7 presents numerical experiments comparing the proposed approach with state-of-the-art baselines. Finally, Section 8 presents some conclusions and possible future directions of research.

2 Polynomial optimization formulation

In this section we derive an upper bound on $L(f_d)$ given by the value of a POP, i.e. the minimum value of a polynomial subject to polynomial inequalities. Our starting point is the following theorem, which casts $L(f)$ as an optimization problem:

Theorem 1. *Let f be a differentiable and Lipschitz continuous function on an open, convex subset $\mathcal{X}$ of an euclidean space. Let $\|\cdot\|_*$ be the dual norm. The Lipschitz constant of f is given by*

$$L(f) = \sup_{x \in \mathcal{X}} \|\boldsymbol{\nabla} f(x)\|_* \tag{2}$$

For completeness we provide a proof in appendix A. In our setting, we assume that the activation function σ is Lipschitz continuous and differentiable. In this case, the assumptions of Theorem 1 are fulfilled because f_d is a composition of activations and linear transformations. The differentiability assumption rules out the common ReLU activation $\sigma(x) = \max\{0, x\}$ but allows many others such as the exponential linear unit (ELU) (Clevert et al., 2015) or the softplus.

Using the chain rule, the compositional structure of f_d yields the following formula for its gradient:

$$\boldsymbol{\nabla} f_d(x) = W_1^T \prod_{i=1}^{d-1} \text{Diag}(\sigma'(f_i(x))) W_{i+1}^T \tag{3}$$

For every $i = 1, \dots, d-1$ introduce a new variable $s_i = \sigma'(f_i(x))$ corresponding to the derivative of σ at the i-th hidden layer of the network. For common activation functions like ELU or softplus, their derivative is globally bounded between 0 and 1, which implies the inequalities $0 \leq s_i \leq 1$. This bound together with the definition of the dual norm $\|x\|_* := \sup_{\|t\| \leq 1} t^T x$ implies that we can upper bound $L(f_d)$ as follows:

$$L(f_d) \leq \max \left\{ t^T W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(s_i) W_{i+1}^T : 0 \leq s_i \leq 1, \|t\| \leq 1 \right\} \tag{4}$$

We will refer to the polynomial objective of this problem as the ***norm-gradient polynomial*** of the network f_d, a central object of study in this work.

For some frequently used ℓ_p-norms the constraint $\|t\|_p \leq 1$ can be written with polynomial inequalities. In the rest of this work, **we use exclusively the ℓ_∞-norm** for which $\|t\|_\infty \leq 1$ is equivalent to the polynomial inequalities $-1 \leq t_i \leq 1$, for $i = 1, \dots, n_1$. However, note that when $p \geq 2$ is a positive even integer, $\|t\|_p \leq 1$ is equivalent to a single polynomial inequality $\|t\|_p^p \leq 1$, and our proposed approach can be adapted with minimal modifications.

In such cases, the optimization problem in the right-hand side of (4) is a POP. Optimization of polynomials is a NP-hard problem and we do not expect to have efficient algorithms for solving (4) in this general form. In the next sections we describe **LiPopt**: a systematic way of obtaining an upper bound on $L(f_d)$ via tractable approximation methods of the POP (4).

Local estimation. In many practical escenarios we have additional bounds on the input of the network. For example in the case of image classification tasks, valid input is constrained in a hypercube. In the robustness certification task, we are interested in all possible input in a ϵ-ball around some data point. In those cases, it is interesting to compute a *local Lipschitz constant*, that is, the Lipschitz constant of a function restricted to a subset.

We can achieve this by deriving tighter bounds $0 \leq l_i \leq s_i \leq u_i \leq 1$, as a consequence of the restricted input (see for example, Algorithm 1 in Wong & Kolter (2018)). By incorporating this knowledge in the optimization problem (4) we obtain bounds on local Lipschitz constants of f_d.

Choice of norm. We highlight the importance of computing good upper bounds on $L(f_d)$ with respect to the ℓ_∞-norm. It is one of the most commonly used norms to assess robustness in the adversarial examples literature. Moreover, it has been shown that in practice ℓ_∞-norm robust networks are also robust in other more plausible measures of perceptibility like the Wasserstein distance (Wong et al., 2019). This motivates our focus on this choice.

3 Hierarchical solution based on a Polynomial Positivity certificate

For ease of exposition, we rewrite (4) as a POP constrained in the hypercube $[0,1]^n$ using the substitution $s_0 := (t+1)/2$. Denote by p the norm-gradient polynomial, and let $x = [s_0, \dots, s_{d-1}]$ be the concatenation of all variables. Polynomial optimization methods (Lasserre, 2015) start from the simple observation that a value λ is an upper bound for p over a set K if and only if the polynomial $\lambda - p$ is positive over K.

In **LiPopt** we will employ the so-called *Krivine's positivity certificate*[1] (Krivine, 1964; Stengle, 1974; Handelman, 1988), but in theory we could use any polynomial positivity certificate like sum-of-squares (SOS). The following is a straightforward adaptation of Krivine's certificate to our particular setting:

Theorem 2. *If the polynomial $\lambda - p$ is strictly positive on $[0,1]^n$, then there exist finitely many positive weights $c_{\alpha\beta}$ such that*

$$\lambda - p = \sum_{(\alpha,\beta) \in \mathbb{N}^{2n}} c_{\alpha\beta} h_{\alpha\beta}, \qquad h_{\alpha\beta}(x) := \prod_{j=1}^{n} x_j^{\alpha_j} (1 - x_j)^{\beta_j} \tag{5}$$

[1] also known as *Krivine's Positivstellensatz*

By truncating the degree of Krivine's positivity certificate (Theorem 2) and minimizing over all possible upper bounds λ we obtain a *hierarchy* of LP problems (Lasserre, 2015, Section 9):

$$\theta_k := \min_{c \geq 0, \lambda} \left\{ \lambda : \lambda - p = \sum_{(\alpha,\beta) \in \mathbb{N}_k^{2n}} c_{\alpha\beta} h_{\alpha\beta} \right\} \tag{6}$$

where $\mathbb{N}_k^{2n}$ is the set of nonnegative integer sequences of length $2n$ adding up to at most k. This is indeed a sequence of LPs as the polynomial equality constraint can be implemented by equating coefficients in the canonical monomial basis. For this polynomial equality to be feasible, the degree of the certificate has to be at least that of the norm-gradient polynomial p, which is equal to the depth d. This implies that the first nontrivial bound ($\theta_k < \infty$) corresponds to $k = d$.

The sequence $\{\theta_k\}_{k=1}^{\infty}$ is non-incresing and converges to the maximum of the upper bound (4). Note however that for any level of the hierarchy, the solution of the LP (6) still provides a valid upper bound on $L(f_d)$.

An advantage of using the Krivine's positivity certificate over SOS is that one obtains an LP hierarchy (rather than SDP), for which state-of-the-art commercial solvers can reliably handle a large number of variables and constraints. Other positivity certificates that offer a similar advantage are the DSOS and SDSOS hierarchies (Ahmadi & Majumdar, 2019), which boil down to LP or *second order cone programming* (SOCP), respectively.

Drawback. The size of the LPs given by Krivine's positivity certificate can become quite large. The dimension of the variable c is $|\mathbb{N}_k^{2n}| = \mathcal{O}(n^k)$. For reference, if we consider the MNIST dataset and a one-hidden-layer network with 100 neurons we have $\left|\mathbb{N}_2^{2n}\right| \approx 1.5 \times 10^6$ while $\left|\mathbb{N}_3^{2n}\right| \approx 9.3 \times 10^8$. In order to make this approach more scalable, in the next section we exploit the structure and sparsity of the polynomial p to find LPs of drastically smaller size than (6), but with similar approximation properties.

Remark. In order to compute upper bounds for local Lipschitz constants, first obtain tighter bounds $0 \leq l_i \leq s_i \leq u_i$ and then perform the change of variables $\widetilde{s}_i = (s_i - l_i)/(u_i - l_i)$ to rewrite the problem (4) as a POP constrained on $[0,1]^n$.

4 Reducing the number of variables

Many neural network architectures, like those composed of convolutional layers, have a highly sparse connectivity between neurons. Moreover, it has been empirically observed that up to 90% of network weights can be *pruned* (set to zero) without harming accuracy (Frankle & Carbin, 2019). In such cases their norm-gradient polynomial has a special structure that allows polynomial positivity certificates of smaller size than the one given by Krivine's positivity certificate (Theorem 2).

In this section we describe an implementation of **LiPopt** (Algorithm 1). It exploits the sparsity of the network to decrease the complexity of the LPs (6) given by the Krivine's positivity certificate. In this way we obtain upper bounds on $L(f_d)$ that require less computation and memory. Let us start with the definition of a *valid sparsity pattern*:

Definition 1. *Let $I = \{1, \ldots, n\}$ and p be a polynomial with variable $x \in \mathbb{R}^n$. A valid sparsity pattern of p is a sequence $\{I_i\}_{i=1}^m$ of subsets of I, called cliques, such that*

▷ $\bigcup_{i=1}^m I_i = I$

▷ $p = \sum_{i=1}^m p_i$ *where p_i is a polynomial that depends only on the variables $\{x_j : j \in I_i\}$*

▷ *for all $i = 1, \ldots, m-1$ there is an $l \leq i$ such that $(I_{i+1} \cap \bigcup_{r=1}^i I_r) \subseteq I_l$*

When the polynomial objective p in a POP has a valid sparsity pattern, there is an extension of Theorem 2 providing a smaller positivity certificate for $\lambda - p$ over $[0,1]^n$. We refer to it as the *sparse Krivine's certificate*.

Theorem 3 (Adapted from Weisser et al. (2018)). *Let a polynomial p have a valid sparsity pattern $\{I_i\}_{i=1}^m$. Define N_i as the set of sequences $(\alpha, \beta) \in \mathbb{N}^{2n}$ where the support of both α and β is*

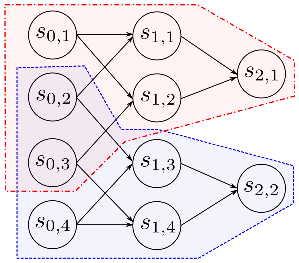


Figure 1: Sparsity pattern of Proposition 1 for a network of depth three.

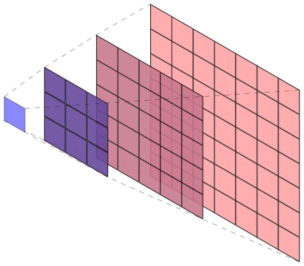


Figure 2: Structure of one set in the sparsity pattern from Proposition 1 for a network with 2D convolutional layers with 3 × 3 filters.

contained in I_i. If $\lambda - p$ is strictly positive over $K = [0,1]^n$, there exist finitely many positive weights $c_{\alpha\beta}$ such that

$$\lambda - p = \sum_{i=1}^{m} h_i, \qquad h_i = \sum_{(\alpha,\beta)\in N_i} c_{\alpha\beta} h_{\alpha\beta} \tag{7}$$

where the polynomials $h_{\alpha\beta}$ are defined as in (5).

The sparse Krivine's certificate can be used like the general version (Theorem 2) to derive a sequence of LPs approximating the upper bound on $L(f_d)$ stated in (4). However, the number of different polynomials $h_{\alpha\beta}$ of degree at most k appearing in the sparse certificate can be drastically smaller, the amount of which determines how *good* the sparsity pattern is.

We introduce a graph that depends on the network f_d, from which we will extract a sparsity pattern for the norm-gradient polynomial of a network.

Definition 2. *Let f_d be a network with weights $\{W_i\}_{i=1}^d$. Define a directed graph $G_d = (V, E)$ as:*

$$\begin{aligned} V &= \{s_{i,j} : 0 \le i \le d-1,\, 1 \le j \le n_i\} \\ E &= \{(s_{i,j}, s_{i+1,k}) : 0 \le i \le d-2, [W_i]_{k,j} \neq 0\} \end{aligned} \tag{8}$$

which we call the computational graph of the network f_d.

In the graph G_d the vertex $s_{(i,j)}$ represents the j-th neuron in the i-th layer. There is a directed edge between two neurons in consecutive layers if they are joined by a nonzero weight in the network. The following result shows that for fully connected networks we can extract a valid sparsity pattern from this graph. We relegate the proof to appendix B.

Proposition 1. *Let f_d be a dense network (all weights are nonzero). The following sets, indexed by $i = 1, \ldots, n_d$, form a valid sparsity pattern for the norm-gradient polynomial of the network f_d:*

$$I_i := \left\{s_{(d-1,i)}\right\} \cup \left\{s_{(j,k)} : \textit{ there exists a directed path from } s_{(j,k)} \textit{ to } s_{(d-1,i)} \textit{ in } G_d\right\} \tag{9}$$

We refer to this as the *sparsity pattern induced by G_d*. An example is depicted in in Figure 1.

Remark. When the network is not dense, the the third condition (Definition 1) for the sparsity pattern (9) to be valid might not hold. In that case we lose the guarantee that the values of the corresponding LPs converge to the maximum of the POP (4). Nevertheless, it still provides a valid positivity certificate that we can use to upper bound $L(f_d)$. In Section 7 we show that in practice it provides upper bounds of good enough quality. If needed, a valid sparsity pattern can be obtained via a chordal completion of the *correlative sparsity graph* of the norm-gradient polynomial (Waki et al., 2006).

We now quantify how good this sparsity pattern is. Let s be the size of the largest clique in a sparsity pattern, and let $N_{i,k}$ be the subset of N_i (defined in Theorem 3) composed of sequences summing up to k. The number of different polynomials for the k-th LP in the hierarchy given by the sparse Krivine's certificate can be bounded as follows:

$$\left|\bigcup_{i=1}^{m} N_{i,k}\right| \le \sum_{i=1}^{m} \binom{2|I_i| + k}{k} = \mathcal{O}\left(ms^k\right) \tag{10}$$

We immediately see that the dependence on the number of cliques m is really mild (linear) but the size of the cliques as well as the degree of the hierarchy can really impact the size of the optimization problem. Nevertheless, this upper bound can be quite loose; polynomials $h_{\alpha\beta}$ that depend only on variables in the intersection of two or more cliques are counted more than once.

The number of cliques given in the sparsity pattern induced by G_d is equal to the size of the last layer $m = n_d$ and the size of each clique depends on the particular implementation of the network. We now study different architectures that could arise in practice, and determine the amount of polynomials in their sparse Krivine's certificate.

Fully connected networks. Even in the case of a network with all nonzero connections, the sparsity pattern induced by G_d decreases the size of the LPs when compared to Krivine's certificate. In this case the cliques have size $n_1 + \ldots + n_{d-1} + 1$ but they all have the same common intersection equal to all neurons up to the second-to-last hidden layer. A straightforward counting argument shows that the total number of polynomials is $\mathcal{O}(n(n_1 + \ldots + n_{d-1} + 1)^{k-1})$, improving the upper bound (10).

Unstructured sparsity. In the case of networks obtained by pruning (Hanson & Pratt, 1989) or generated randomly from a distribution over graphs (Xie et al., 2019), the sparsity pattern can be highly irregular. In this case the size of the resulting LPs varies at runtime. Under the layer-wise assumption that any neuron is connected to at most r neurons in the previous layer, the size of the cliques in (9) is bounded as $s = \mathcal{O}(r^d)$. This estimate has an exponential dependency on the depth but completely ignores that many neurons might share connections to the same inputs in the previous layer, thus being potentially loose. Using the bound (10) we get that the number of different polynomials is $\mathcal{O}(n_d r^{dk})$.

2D Convolutional networks. The sparsity in the weight matrices of convolutional layers has a certain *local structure*; neurons are connected to contiguous inputs in the previous layer. Adjacent neurons also have many input pixels in common (see Figure 2). Assuming a constant number of channels per layer, the size of the cliques in (9) is $\mathcal{O}(d^3)$. Intuitively, such number is proportional to the volume of the pyramid depicted in Figure 2 where each dimension depends linearly on d. Using (10) we get that there are $\mathcal{O}(n_d d^{3k})$ different polynomials in the sparse Krivine's certificate. This is a drastic decrease in complexity when compared to the unstructured sparsity case.

Algorithm 1 LiPopt for ELU activations and sparsity pattern

```
Input: matrices $\{W_i\}_{i=1}^d$, sparsity pattern $\{I_i\}_{i=1}^m$, hierarchy degree $k$.
$p \leftarrow (2s_0 - 1)^T W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(s_i) W_{i+1}^T$    ▷ compute norm-gradient polynomial
$x \leftarrow [s_0, \ldots, s_{d-1}]$
$b \leftarrow [b_\gamma : \gamma \in \mathbb{N}_k^n]$ where $p(x) = \sum_{\gamma \in \mathbb{N}_k^n} b_\gamma x^\gamma$    ▷ compute coefficients of $p$ in basis
for $i = 1, \ldots, m$ do
    $N_{i,k} \leftarrow \{(\alpha, \beta) \in \mathbb{N}_k^{2n} : \mathrm{supp}(\alpha) \cap \mathrm{supp}(\beta) \subseteq I_i\}$
$\widetilde{N}_k \leftarrow \cup_{i=1}^m N_{i,k}$
$h \leftarrow \sum_{(\alpha,\beta) \in \widetilde{N}} c_{\alpha\beta} h_{\alpha\beta}$    ▷ compute positivity certificate
$c \leftarrow [c_{\alpha\beta} : (\alpha, \beta) \in \widetilde{N}_k]$;  $y \leftarrow [\lambda, c]$    ▷ linear program variables
$Z \leftarrow [z_\gamma]_{\gamma \in \mathbb{N}_k^n}$ where $\lambda - h(x) = \sum_{\gamma \in \mathbb{N}_k^n} (z_\gamma^T y) x^\gamma$    ▷ compute coefficients of $\lambda - h$ in basis
return $\min\{\lambda : b = Zy,\ y = [\lambda, c],\ c \geq 0\}$    ▷ solve LP
```

5 QCQP REFORMULATION AND SHOR'S SDP RELAXATION

Another way of upper bounding $L(f_d)$ comes from a further relaxation of (4) to an SDP. We consider the following equivalent formulation where the variables s_i are normalized to lie in the interval $[-1, 1]$, and we rename $t = s_0$:

$$L(f_d) \leq \max \left\{ \frac{1}{2^{d-1}} s_0^T W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(s_i + 1) W_{i+1}^T : -1 \leq s_i \leq 1 \right\} \tag{11}$$

Any polynomial optimization problem like (11) can be cast as a (possibly non-convex) *quadratically constrained quadratic program* (QCQP) by introducing new variables and quadratic constraints.

This is a well-known procedure described in Park & Boyd (2017, Section 2.1). Note that when $d = 2$ problem (11) is already a QCQP (for the ℓ_∞ and ℓ_2-norm cases) and no modification is necessary.

QCQP reformulation. We illustrate the case $d = 3$ where we have the variables s_1, s_2 corresponding to the first and second hidden layer and a variable s_0 corresponding to the input. The norm-gradient polynomial in this case is cubic, and it can be rewritten as a quadratic polynomial by introducing new variables corresponding to the product of the first and second hidden layer variables.

More precisely the introduction of a variable $s_{1,2}$ with quadratic constraint $s_{1,2} = \text{vec}(s_1 s_2^T)$ allows us to write the objective (11) as a quadratic polynomial. The problem then becomes a QCQP with variable $y = [1, s_0, s_1, s_2, s_{1,2}]$ of dimension $1 + n + n_1 n_2$.

SDP relaxation. Any quadratic objective and constraints can then be relaxed to linear constraints on the positive semidefinite variable $yy^T = X \succcurlyeq 0$ yielding the so-called *Shor's relaxation* of (11) (Park & Boyd, 2017, Section 3.3). When $d = 2$ the resulting SDP corresponds precisely to the one studied in Raghunathan et al. (2018a). This resolves a common misconception (Raghunathan et al., 2018b) that this approach is only limited to networks with one hidden layer.

Note that in our setting we are only interested in the optimal value rather than the optimizers, so there is no need to extract a solution for (11) from that of the SDP relaxation.

Drawback. This approach includes a further relaxation step from (11), thus being fundamentally limited in how tightly it can upper bound the value of $L(f_d)$. Moreover when compared to LP solvers, off-the-shelf semidefinite programming solvers are, in general, much more limited in the number of variables they can efficiently handle.

Nevertheless, we remark that in the case $d = 2$ this relaxation provides a constant factor approximation to the original QCQP (Ye, 1999). The approximation qualities of such hierarchical optimization approaches to NP-hard problems are a big topic of research in theoretical computer science and are out of the scope of this work.

Relation to sum-of-squares. On the surface, this approach might appear fundamentaly different to the hierarchical optimization approaches to POPs, like the one described in Section 3. However, it is known that Shor's SDP relaxation corresponds exactly to the first degree of the SOS hierarchical SDP solution to the QCQP relaxation (Lasserre, 2000). Thus, the solution described in section 3 and the one in this section are, in essence, the same; they only differ in the choice of polynomial positivity certificate.

6 Related work

Estimation of $L(f_d)$ with ℓ_2-norm is studied by Virmaux & Scaman (2018); Combettes & Pesquet (2019); Fazlyab et al. (2019); Jin & Lavaei (2018). The method **SeqLip** proposed in Virmaux & Scaman (2018) has the drawback of not providing true upper bounds. It is in fact a heuristic method for solving (4) but which provides no guarantees and thus can not be used for robustness certification. In contrast the **LipSDP** method of Jin & Lavaei (2018); Fazlyab et al. (2019) provides true upper bounds on $L(f_d)$ and in practice shows superior performance over both **SeqLip** and **CPLip** (Combettes & Pesquet, 2019).

Despite the accurate estimation of **LipSDP**, its formulation is limited to the ℓ_2-norm. The only estimate available for other ℓ_p-norms comes from the equivalence of norms in euclidean spaces. For instance, we can obtain an upper bound for the ℓ_∞-norm after multiplying the ℓ_2 Lipschitz constant upper bound by the square root of the input dimension. The resulting bound can be rather loose and our experiments in section 7 confirm the issue. In contrast, our proposed approach **LiPopt** can acommodate any norm whose unit ball can be described via polynomial inequalities.

Let us point to one key advantage of **LiPopt**, compared to **LipSDP** (Jin & Lavaei, 2018; Fazlyab et al., 2019). In the context of robustness certification we are given a sample $x^\natural$ and a ball of radius ϵ around it. Computing an upper bound on the local Lipschitz constant in this subset, rather than a global one, can provide a larger region of certified robustness. Taking into account the restricted domain we can refine the bounds in our POP (see remark in section 1). This potentially yields a tighter estimate of the local Lipschitz constant. On the other hand, it is not clear how to include such additional information in **LipSDP**, which only computes one global bound on the Lipschitz constant for the unconstrained network.

Raghunathan et al. (2018a) find an upper bound for $L(f_d)$ with ℓ_∞ metric starting from problem (4) but only in the context of one-hidden-layer networks ($d = 2$). To compute such bound they use its corresponding Shor's relaxation and obtain as a byproduct a differentiable regularizer for training networks. They claim such approach is limited to the setting $d = 2$ but, as we remark in section 5, it is just a particular instance of the SDP relaxation method for QCQPs arising from a polynomial optimization problem. We find that this method fits into the **LiPopt** framework, using SOS certificates instead of Krivine's. We expect that the SDP-based bounds described in 5 can also be used as regularizers promoting robustness.

Weng et al. (2018) provide an upper bound on the local Lipschitz constant for networks based on a sequence of ad-hoc bounding arguments, which are particular to the choice of ReLU activation function. In contrast, our approach applies in general to activations whose derivative is bounded.

7 Experiments

We consider the following estimators of $L(f_d)$ with respect to the ℓ_∞ norm:

Name	**Description**
SDP	Upper bound arising from the solution of the SDP relaxation described in 5
LipOpt-k	Upper bound arising from the k-th degree of the LP hierarchy (6) based on the sparse Krivine Positivstellenstatz.
Lip-SDP	Upper bound from Jin & Lavaei (2018); Fazlyab et al. (2019) multiplied $\sqrt{d}$ where d is the input dimension of the network.
UBP	Upper bound determined by the product of the layer-wise Lipschitz constants with ℓ_∞ metric
LBS	Lower bound obtained by sampling 50000 random points and evaluating the dual norm of the gradient

7.1 Experiments on random networks

We first compare the various bounds obtained by the algorithms described above on networks with random weights. We use feedforward networks with either one or two hidden layers. In addition, we define the sparsity level of a network as the maximum number of neurons any neuron in one layer is connected to in the next layer. For example, the network represented on Figure 1 has sparsity 2. The non-zero weights of network's i-th layer are generated using the default pytorch initializations, i.e., independently sampled from Unif$(-\frac{1}{\sqrt{n_i}}, \frac{1}{\sqrt{n_i}})$ where n_i is the number of neurons in layer i.

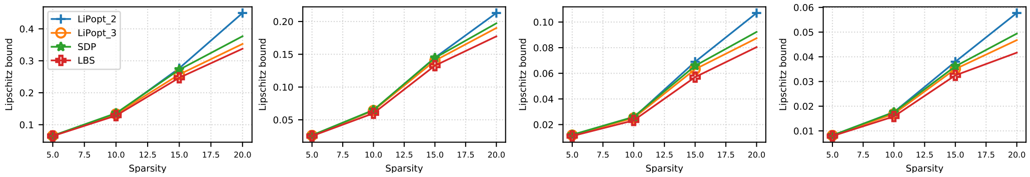
We computed the Lipschitz bounds obtained by these different methods for various network sizes and sparsities. For each configuration, we generated 10 different random networks on which to test the various methods, and average the obtain Lipschitz bounds. Figures 3 and 4 show the results in the case of 1 and 2 hidden layers respectively.

When the chosen degree for **LiPopt-k** is the smallest as possible, i.e., equal to the depth of the network, we observe that the method is already competitive with the **SDP** method, especially in the case of 2 hidden layers. Moreover, when incrementing the degree by 1, **LiPopt-k** becomes uniformly better than **SDP** over all tested configurations.

Remark: The upper bounds given by **UBP** are much too large to be shown on the plots. Similarly, for the 1-hidden layer networks, the bounds from **LipSDP** are also too large to be plotted.

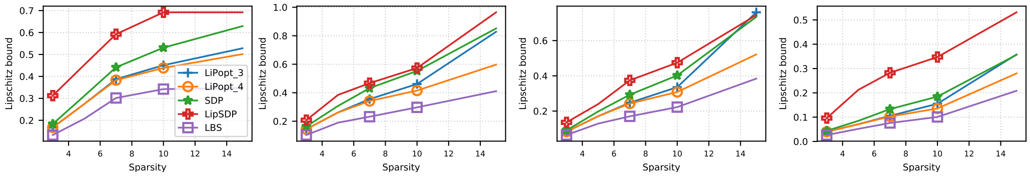
7.2 Experiments on trained networks

Similarly, we compare these methods on networks trained on MNIST. The architecture we use is a fully connected network with two hidden layers with 300 and 100 neurons respectively, and with one-hot output of size 10. Since the output is multi-dimensional, we restrict the network to a single output, and estimate the Lipschitz constant with respect to label 8.



(a) 40 × 40 (b) 80 × 80 (c) 160 × 160 (d) 320 × 320

Figure 3: Lipschitz bound comparison for 1-hidden layer networks



(a) 5 × 5 × 10 (b) 10 × 10 × 10 (c) 20 × 20 × 10 (d) 40 × 40 × 10

Figure 4: Lipschitz bound comparison for 2-hidden layer networks

Moreover, in order to improve the scalability of our method, we train the network using the pruning strategy described in Han et al. (2015). After training the full network using a standard technique, the weights of smallest magnitude are set to zero. Then, the network is trained for additional iterations, only updating the nonzero parameters. Doing so, we were able to remove 95% of the weights, while preserving the same test accuracy. The code that we used for training is the one provided in the paper Han et al. (2015) and can be found here: https://github.com/mightydeveloper/Deep-Compression-PyTorch.

We recorded the Lipschitz bounds for various methods in Table 7.2. We observe clear improvement of the Lipschitz bound obtained from **LiPopt-k** compared to **SDP** method, even when using $k = 3$. Also note that the input dimension is too large for the method **Lip-SDP** to provide competitive bound, so we do not provide the obtained bound for this method.

Algorithm	**LBS**	**LiPopt-3**	**SDP**	**UBP**
Lipschitz bound	93.2	94.6	98.8	691.5

8 Conclusion and future work

In this work, we have introduced a general approach for computing an upper bound on the Lipschitz constant of neural networks. This approach is based on polynomial positivity certificates and generalizes some existing methods available in the literature. We have empirically demonstrated that it can tightly upper bound such constant. The resulting optimization problems are computationally expensive but the sparsity of the network can reduce this burden.

In order to further scale such methods to larger and deeper networks, we are interested in several possible directions: (i) divide-and-conquer approaches splitting the computation on sub-networks in the same spirit of Fazlyab et al. (2019), (ii) exploiting parallel optimization algorithms leveraging the structure of the polynomials, (iii) custom optimization algorithms with low-memory costs such as Frank-wolfe-type methods for SDP (Yurtsever et al., 2019) as well as stochastic handling of constraints (Fercoq et al., 2019) and (iv), exploting the symmetries in the polynomial that arise from weight sharing in typical network architectures to further reduce the size of the problems.

References

A. Ahmadi and A. Majumdar. Dsos and sdsos optimization: More tractable alternatives to sum of squares and semidefinite optimization. *SIAM Journal on Applied Algebra and Geometry*, 3(2):193–230, 2019. doi: 10.1137/18M118935X. URL `https://doi.org/10.1137/18M118935X`.

Cem Anil, James Lucas, and Roger Grosse. Sorting out Lipschitz function approximation. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 291–301, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/anil19a.html`.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/arjovsky17a.html`.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems 30*, pp. 6240–6249. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7204-spectrally-normalized-margin-bounds-for-neural-networks.pdf`.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in Neural Information Processing Systems 30*, pp. 908–918. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/6692-safe-model-based-reinforcement-learning-with-stability-guarantees.pdf`.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 854–863, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL `http://proceedings.mlr.press/v70/cisse17a.html`.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv e-prints*, art. arXiv:1511.07289, Nov 2015.

Patrick L. Combettes and Jean-Christophe Pesquet. Lipschitz Certificates for Neural Network Structures Driven by Averaged Activation Operators. *arXiv e-prints*, art. arXiv:1903.01014, Mar 2019.

Mahyar Fazlyab, Alexander Robey, Hamed Hassani, Manfred Morari, and George J. Pappas. Efficient and Accurate Estimation of Lipschitz Constants for Deep Neural Networks. *arXiv e-prints*, art. arXiv:1906.04893, Jun 2019.

Olivier Fercoq, Ahmet Alacaoglu, Ion Necoara, and Volkan Cevher. Almost surely constrained convex optimization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1910–1919, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/fercoq19a.html`.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=rJl-b3RcF7`.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems 30*, pp. 5767–5777. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7159-improved-training-of-wasserstein-gans.pdf`.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

David Handelman. Representing polynomials by positive linear functions on compact convex polyhedra. *Pacific J. Math.*, 132(1):35–62, 1988. URL `https://projecteuclid.org:443/euclid.pjm/1102689794`.

Stephen Jose Hanson and Lorien Y. Pratt. Comparing biases for minimal network construction with back-propagation. In *Advances in Neural Information Processing Systems 1*, pp. 177–185. Morgan-Kaufmann, 1989. URL http://papers.nips.cc/paper/156-comparing-biases-for-minimal-network-construction-with-back-propagation.pdf.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 06 2016. doi: 10.1109/CVPR.2016.90.

Todd Huster, Cho-Yu Jason Chiang, and Ritu Chadha. Limitations of the Lipschitz constant as a defense against adversarial examples. *arXiv e-prints*, art. arXiv:1807.09705, Jul 2018.

Ming Jin and Javad Lavaei. Stability-certified reinforcement learning: A control-theoretic perspective. *arXiv e-prints*, art. arXiv:1810.11505, Oct 2018.

Jean-Louis Krivine. Anneaux préordonnés. *Journal d'analyse mathématique*, 12:p. 307–326, 1964. URL https://hal.archives-ouvertes.fr/hal-00165658.

J. B. Lasserre. Convergent lmi relaxations for nonconvex quadratic programs. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No.00CH37187)*, volume 5, pp. 5041–5046 vol.5, Dec 2000. doi: 10.1109/CDC.2001.914738.

Jean B Lasserre. Convergent sdp-relaxations in polynomial optimization with sparsity. *SIAM Journal on Optimization*, 17(3):822–843, 2006.

Jean Bernard Lasserre. *An Introduction to Polynomial and Semi-Algebraic Optimization*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2015. doi: 10.1017/CBO9781107447226.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. doi: 10.1109/5.726791.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1QRgziT-.

Jaehyun Park and Stephen Boyd. General heuristics for nonconvex quadratically constrained quadratic programming. *arXiv preprint arXiv:1703.07870*, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=Bys4ob-Rb.

Aditi Raghunathan, Jacob Steinhardt, and Percy S Liang. Semidefinite relaxations for certifying robustness to adversarial examples. In *Advances in Neural Information Processing Systems*, pp. 10877–10887, 2018b.

Gilbert Stengle. A nullstellensatz and a positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207(2):87–97, Jun 1974. ISSN 1432-1807. doi: 10.1007/BF01362149. URL https://doi.org/10.1007/BF01362149.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014. URL http://arxiv.org/abs/1312.6199.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *Advances in Neural Information Processing Systems 31*, pp. 3835–3844. Curran Associates, Inc., 2018.

H. Waki, S. Kim, M. Kojima, and M. Muramatsu. Sums of squares and semidefinite program relaxations for polynomial optimization problems with structured sparsity. *SIAM Journal on Optimization*, 17(1):218–242, 2006. doi: 10.1137/050623802.

Tillmann Weisser, Jean B Lasserre, and Kim-Chuan Toh. Sparse-bsos: a bounded degree sos hierarchy for large scale polynomial optimization with sparsity. *Mathematical Programming Computation*, 10(1):1–32, 2018.

Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for ReLU networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5276–5285, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/weng18a.html`.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 5286–5295, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Eric Wong, Frank Schmidt, and Zico Kolter. Wasserstein adversarial examples via projected Sinkhorn iterations. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6808–6817, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/wong19a.html`.

Saining Xie, Alexander Kirillov, Ross Girshick, and Kaiming He. Exploring Randomly Wired Neural Networks for Image Recognition. *International Conference on Computer Vision*, Apr 2019.

Yinyu Ye. Approximating quadratic programming with bound and quadratic constraints. *Mathematical Programming*, 84(2):219–226, Feb 1999. ISSN 1436-4646. doi: 10.1007/s10107980012a. URL `https://doi.org/10.1007/s10107980012a`.

Alp Yurtsever, Olivier Fercoq, and Volkan Cevher. A conditional-gradient-based augmented lagrangian framework. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7272–7281, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/yurtsever19a.html`.

A Proof of Theorem 1

Theorem. *Let f be a differentiable and Lipschitz continuous function on an open, convex subset $\mathcal{X}$ of a euclidean space. Let $\|\cdot\|$ be the dual norm. The Lipschitz constant of f is given by*

$$L(f) = \sup_{x \in \mathcal{X}} \|\boldsymbol{\nabla} f(x)\|_* \tag{12}$$

Proof. First we show that $L(f) \leq \sup_{x\in\mathcal{X}} \|\boldsymbol{\nabla} f(x)\|_*$.

$$\begin{aligned} |f(y) - f(x)| &= \left| \int_0^1 \nabla f((1-t)x + ty)^T (y-x)\, dt \right| \\ &\leq \int_0^1 \left| \nabla f((1-t)x + ty)^T (y-x) \right| dt \\ &\leq \int_0^1 \|\nabla f((1-t)x + ty)\|_* \, dt \, \|y - x\| \\ &\leq \sup_{x\in\mathcal{X}} \|\nabla f(x)\|_* \|y - x\| \end{aligned}$$

were we have used the convexity of $\mathcal{X}$.

Now we show the reverse inequality $L(f) \geq \sup_{x\in\mathcal{X}} \|\nabla f(x)\|_*$. To this end, we show that for any positive ϵ, we have that $L(f) \geq \sup_{x\in\mathcal{X}} \|\nabla f(x)\|_* - \epsilon$.

Let $z \in \mathcal{X}$ be such that $\|\nabla f(z)\|_* \geq \sup_{x\in\mathcal{X}} \|\nabla f(x)\|_* - \epsilon$. Because $\mathcal{X}$ is open, there exists a sequence $\{h_k\}_{k=1}^{\infty}$ with the following properties:

1. $\langle h_k, \nabla f(z)\rangle = \|h_k\|\|\nabla f(z)\|_*$
2. $z + h_k \in \mathcal{X}$
3. $\lim_{k\to\infty} h_k = 0$.

By definition of the gradient, there exists a function δ such that $\lim_{h\to 0} \delta(h) = 0$ and the following holds:

$$f(z+h) = f(z) + \langle h, \nabla f(z)\rangle + \delta(h)\|h\|$$

For our previously defined iterates h_k we then have

$$\Rightarrow |f(z+h_k) - f(z)| = |\|h_k\|\|\nabla f(z)\|_* + \delta(h_k)\|h_k\||$$

Dividing both sides by $\|h_k\|$ and using the definition of $L(f)$ we finally get

$$\Rightarrow L(f) \geq \left|\frac{f(z+h_k) - f(z)}{\|h_k\|}\right| = |\|f(z)\|_* + \delta(h_k)|$$
$$\Rightarrow L(f) \geq \lim_{k\to\infty} |\|f(z)\|_* + \delta(h_k)| = \|f(z)\|_*$$
$$\Rightarrow L(f) \geq \sup_{x\in\mathcal{X}} \|\nabla f(x)\|_* - \epsilon$$

□

B Proof of Proposition 1

Proposition. *Let f_d be a dense network (all weights are nonzero). The following sets, indexed by $i = 1, \ldots, n_d$, form a valid sparsity pattern for the norm-gradient polynomial of the network f_d:*

$$I_i := \left\{s_{(d-1,i)}\right\} \cup \left\{s_{(j,k)} : \textit{ there exists a directed path from } s_{(j,k)} \textit{ to } s_{(d-1,i)} \textit{ in } G_d\right\} \tag{13}$$

Proof. First we show that $\cup_{i=1}^{m} I_i = I$. This comes from the fact that any neuron in the network is connected to at least one neuron in the last layer. Otherwise such neuron could be removed from the network altogether.

Now we show the second property of a valid sparsity pattern. Note that the norm-gradient polynomial is composed of monomials corresponding to the product of variables in a path from input to a final neuron. This imples that if we let p_i be the sum of all the terms that involve the neuron $s_{(d-1,i)}$ we have that $p = \sum_i p_i$, and p_i only depends on the variables in I_i.

We now show the last property of the valid sparsity pattern. This is the only part where we use that the network is dense. For any network architecture the first two conditions hold. We will use the fact that the maximal cliques of a chordal graph form a valid sparsity pattern (see for example Lasserre (2006)).

Because the network is dense, we see that the clique I_i is composed of the neuron in the last layer $s_{(d-1,i)}$ and all neurons in the previous layers. Now consider the extension of the computational graph $\hat{G}_d = (V, \hat{E})$ where

$$\hat{E} = E \cup \{(s_{j,k}, s_{l,m}) : j, l \leq d-2)\}$$

which consists of adding all the edges between the neurons that are not in the last layer. We show that this graph is chordal. Let $(a_1, \ldots, a_r, a_1)$ be a cycle of length at least 4 ($r \geq 4$). notice that because neurons in the last layer are not connected between them in $\hat{G}$, no two consecutive neurons in this cycle belong to the last layer. This implies that in the subsequence $(a_1, a_2, a_3, a_4, a_5)$ at most three belong to the last layer. A simple analysis of all cases implies that it contains at least two nonconsecutive neurons not in the last layer. Neurons not in the last layer are always connected in $\hat{G}$. This constitutes a chord. This shows that $\hat{G}_d$ is a chordal graph. Its maximal cliques correspond exactly to the sets in proposition.

□