OPT2019: 11th Annual Workshop on Optimization for Machine Learning

Estimating the Lipschitz constant of Neural Networks with Polynomial Optimization methods

Fabian Latorre FABIAN.LATORRE@EPFL.CH
Paul Rolland PAUL.ROLLAND@EPFL.CH
Volkan Cevher VOLKAN.CEVHER@EPFL.CH
École polytechnique fédérale de Lausanne (EPFL), Switzerland

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. The sparsity of the network can 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 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. [2] 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 [20] and can be used to improve such metric [3]. Moreover, an upper bound on $L(f_d)$ provides a certificate of robust classification around data points [24].

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 Fazlyab et al. [4], Jin and Lavaei [10], Raghunathan et al. [17] 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.

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)* [15]. This approach requires only that the unit ball be described with polynomial inequalities, which covers the common ℓ_2- and ℓ_∞-norms.

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, $\mathrm{Diag}(x)$ denotes the square matrix with x in its diagonal and zeros everywhere else. For an array X, $\mathrm{vec}(X)$ is the *flattened* array. The support of a sequence $\mathrm{supp}(\alpha)$ 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 $\mathbb{N}$.

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*

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

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.

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

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

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

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

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 $i = 1, \ldots, n_1$. 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).

3. Hierarchical solution based on a Polynomial Positivity certificate

For ease of exposition, we rewrite (4) as a POP constrained in $[0,1]^n$ 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 [15] start from the 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 a well-known classical result in algebraic geometry, the so-called *Krivine's positivity certificate*[1], but in theory we can use any positivity certificate like sum-of-squares (SOS). The following is a straightforward adaptation of Krivine's certificate to our setting:

Theorem 2 *(Adapted from [8, 12, 19]) If the polynomial $\lambda - p$ is strictly positive on $[0,1]^n$, then there exist finitely many positive weights $c_{\alpha\beta}$ such that*

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

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 [15, 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 $k = d$.

An advantage of using Krivine's positivity certificate over SOS is that one obtains an LP hierarchy (rather than SDP), for which commercial solvers can reliably handle a large instances. Other positivity certificates offering a similar advantage are the DSOS and SDSOS hierarchies [1], 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 $|\mathbb{N}_k^{2n}| = \mathcal{O}(n^k)$. To make this approach more scalable, in appendix C we describe how to exploit the sparsity of the polynomial p to find LPs of drastically smaller size than (6), but with similar approximation properties.

4. Experiments

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

1. also known as *Krivine's Positivstellensatz*

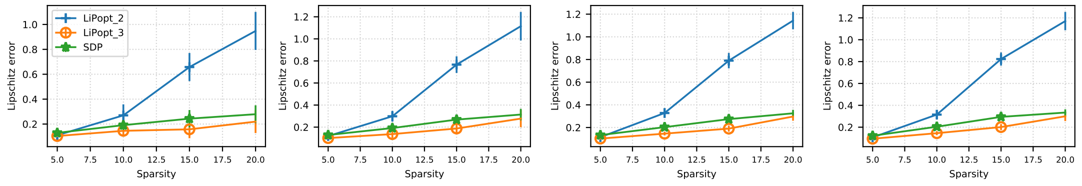
Name	Description
SDP	Upper bound arising from the solution of the SDP relaxation described in appendix D
LipOpt-k	Upper bound arising from the k-th degree of the LP hierarchy (6) based on the sparse Krivine Positivstellenstatz.
Lip-SDP	Upper bound from Fazlyab et al. [4] multiplied $\sqrt{d}$ where d 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 around zero, and evaluating the dual norm of the gradient

4.1. Experiments on random networks

We compare the bounds obtained by the algorithms described above on networks with random weights and either one or two hidden layers. 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 the experimental setup details, see appendix E.

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. When we increment the degree by 1, **LiPopt-k** becomes uniformly better than **SDP** over all tested configurations. We remark that the upper bounds given by **UBP** are too large to be shown in the plots. Similarly, for the 1-hidden layer networks, the bounds from **LipSDP** are too large to be plotted.

We measured the computation time of the different methods on each tested network (Figures 2 and 4). We observe that the computation time for **LiPopt-k** heavily depends on the network sparsity, which reflects the fact that such structure is exploited in the algorithm.



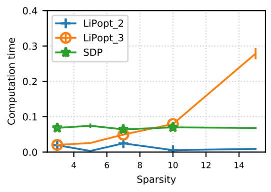
(a) 40×40 (b) 80×80 (c) 160×160 (d) 320×320

Figure 1: Lipschitz approximated relative error for 1-hidden layer networks

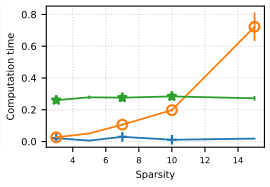
4.2. Experiments on trained networks

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.

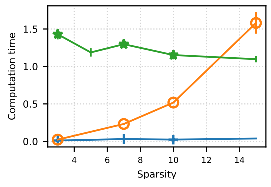
In order to improve the scalability of our method, we train the network using the pruning strategy described in [7]. Doing so, we were able to remove 95% of the weights, while preserving the same



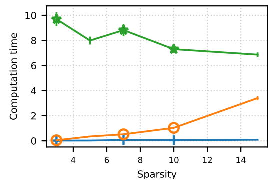
(a) 40 × 40



(b) 80 × 80

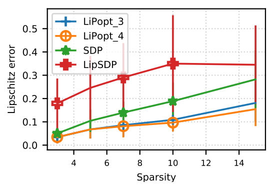


(c) 160 × 160

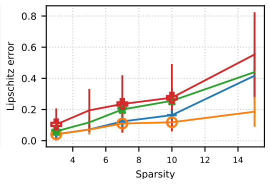


(d) 320 × 320

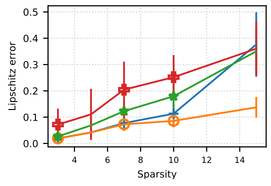
Figure 2: Computation times for 1-hidden layer networks (seconds)



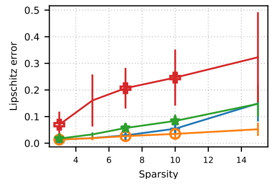
(a) 5 × 5 × 10



(b) 10 × 10 × 10

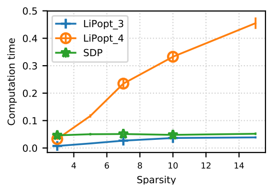


(c) 20 × 20 × 10

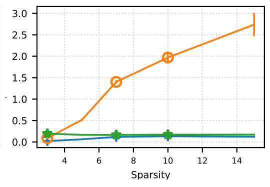


(d) 40 × 40 × 10

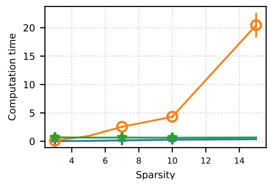
Figure 3: Lipschitz approximated relative error for 2-hidden layer networks



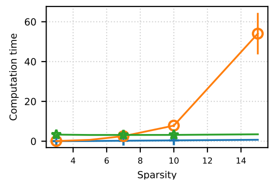
(a) 5 × 5 × 10



(b) 10 × 10 × 10



(c) 20 × 20 × 10



(d) 40 × 40 × 10

Figure 4: Computation times for 2-hidden layer networks (seconds)

test accuracy. We recorded the Lipschitz bounds for various methods in Table 4.2. 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-4**	**LiPopt-3**	**SDP**	**UBP**
Lipschitz bound	84.2	88.3	94.6	98.8	691.5

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement 725594 - time-data). PR is supported by the Swiss National Science Foundation (SNSF) under grant number 407540_167319. FL is supported through a PhD fellowship of the Swiss Data Science Center, a joint venture between EPFL and ETH Zurich. VC acknowledges the 2019 Google Faculty Research Award.

References

[1] 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.

[2] 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*, pages 6240–6249. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7204-spectrally-normalized-margin-bounds-for-neural-networks.pdf.

[3] 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*, pages 854–863, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/cisse17a.html.

[4] 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.

[5] 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.

[6] Bissan Ghaddar, Jakub Marecek, and Martin Mevissen. Optimal power flow as a polynomial optimization problem. *IEEE Transactions on Power Systems*, 31(1):539–546, 2015.

[7] 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.

[8] 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.

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

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

[11] Masakazu Kojima, Sunyoung Kim, and Hayato Waki. Sparsity in sums of squares of polynomials. *Mathematical Programming*, 103(1):45–62, May 2005. ISSN 1436-4646. doi: 10.1007/s10107-004-0554-3. URL https://doi.org/10.1007/s10107-004-0554-3.

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

[13] 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, pages 5041–5046 vol.5, Dec 2000. doi: 10.1109/CDC.2001.914738.

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

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

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

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

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

[19] 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`.

[20] 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`.

[21] 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.

[22] Zizhuo Wang, Song Zheng, Stephen Boyd, and Yinyu Ye. Further relaxations of the sdp approach to sensor network localization. *Tech. Rep.*, 2006.

[23] 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.

[24] 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*, pages 5276–5285, Stockholmsmässan,

Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/weng18a.html.

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

[26] Zhengfeng Yang, Chao Huang, Xin Chen, Wang Lin, and Zhiming Liu. A linear programming relaxation based approach for generating barrier certificates of hybrid systems. In John Fitzgerald, Constance Heitmeyer, Stefania Gnesi, and Anna Philippou, editors, *FM 2016: Formal Methods*, pages 721–738, Cham, 2016. Springer International Publishing. ISBN 978-3-319-48989-6.

[27] 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.

[28] Y. Zhang, Z. Yang, W. Lin, H. Zhu, X. Chen, and X. Li. Safety verification of nonlinear hybrid systems based on bilinear programming. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(11):2768–2778, Nov 2018. doi: 10.1109/TCAD.2018.2858383.

Appendix 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*

$$L(f) = \sup_{x\in\mathcal{X}} \|\nabla f(x)\|_* \tag{7}$$

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

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

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

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

■

Appendix B. Proof of Proposition 6

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

$$I_i := \{s_{(d-1,i)}\} \cup \{s_{(j,k)} : \textit{there exists a directed path from } s_{(j,k)} \textit{ to } s_{(d-1,i)} \textit{ in } G_d\} \tag{8}$$

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 [14]).

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 $(a_1, \dots, a_r, a_1)$ 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.

■

Appendix C. 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 [5]. 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 how to exploit the sparsity of the network to decrease the complexity of the LPs (6) given by the Krivine's positivity certificate. In this way, **LiPopt** can 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 3 *Let $I = \{1, \dots, n\}$ 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 $\bigcup_{i=1}^m I_i = I$ and:*

- ▷ $p = \sum_{i=1}^m p_i$ *where p_i is a polynomial that depends only on the variables $\{x_j : j \in I_i\}$*
- ▷ *for all $i = 1, \dots, m-1$ 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 due to Weisser et al. [23], providing a smaller positivity certificate for $\lambda - p$ over $[0,1]^n$. We refer to it as the *sparse Krivine's certificate* and we include it here for completeness:

Theorem 4 (Adapted from Weisser et al. [23]) *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 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*

$$\lambda - p = \sum_{i=1}^m h_i, \qquad h_i = \sum_{(\alpha,\beta)\in N_i} c_{\alpha\beta} h_{\alpha\beta} \tag{9}$$

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

$$\begin{aligned} V &= \{s_{i,j} : 0 \leq i \leq d-1,\ 1 \leq j \leq n_i\} \\ E &= \{(s_{i,j}, s_{i+1,k}) : 0 \leq i \leq d-2,\ [W_i]_{k,j} \neq 0\} \end{aligned} \tag{10}$$

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 delegate the proof to appendix B.

Proposition 6 *Let f_d be a dense network (all weights are nonzero). The following sets, indexed by $i = 1, \dots, n_d$, form a valid sparsity pattern for the norm-gradient polynomial of the network f_d:*

$$I_i := \left\{s_{(d-1,i)}\right\} \cup \left\{s_{(j,k)} : \text{ there exists a directed path from } s_{(j,k)} \text{ to } s_{(d-1,i)} \text{ in } G_d\right\} \tag{11}$$

We refer to this as the *sparsity pattern induced by G_d*. An example is depicted in in Figure 5.

Remark. When the network is not dense, the the second condition (Definition 3) for the sparsity pattern (11) 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

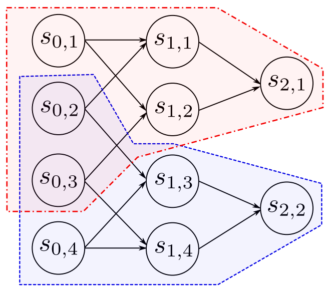


Figure 5: Sparsity pattern of Proposition 6 for a network of depth three.

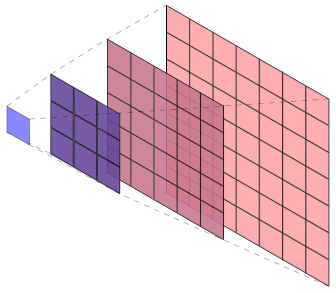


Figure 6: Structure of one set in the sparsity pattern from Proposition 6 for a network with 2D convolutional layers with 3×3 filters.

positivity certificate that we use to upper bound $L(f_d)$. In Section 4 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 POP [21].

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 4) 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:

$$\left| \bigcup_{i=1}^{m} N_{i,k} \right| \leq \sum_{i=1}^{m} \binom{2|I_i| + k}{k} = \mathcal{O}\left(ms^k\right) \tag{12}$$

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 (12).

Unstructured sparsity. In the case of networks obtained by pruning [9] or generated randomly from a distribution over graphs [25], the sparsity pattern can be arbitrary. 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 (11) is bounded as $s = \mathcal{O}(r^d)$. This estimate has an exponential dependency on the depth but ignores that many neurons might share connections to the same inputs in the previous layer, thus being potentially loose. The bound (12) implies 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 6). Assuming a constant number of channels per layer, the size of the cliques in (11) is $\mathcal{O}(d^3)$. Intuitively, such number is proportional to the volume of the pyramid depicted in Figure 6 where each dimension depends linearly on d. Using (12) 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.

The use of sparsity in polynomial optimization preceeds Theorem 4 [23]. First studied in the context of sum-of-squares by Kojima et al. [11] and further refined in Lasserre [14], Waki et al. [21] (and references therein), it has found applications in safety verification [26, 28], sensor localization Wang et al. [22], optimal power flow [6] and many others. Our work fits precisely into this set of important applications.

Appendix D. 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 $t = s_0$:

$$L(f_d) \leq \max \left\{ \frac{1}{2^{d-1}} s_0^T W_1^T \prod_{i=1}^{d-1} \mathrm{Diag}(s_i + 1) W_{i+1}^T : -1 \leq s_i \leq 1 \right\} \tag{13}$$

Any polynomial optimization problem like (13) 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 and Boyd [16, Section 2.1]. When $d = 2$ problem (13) 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 $s_{1,2} = \mathrm{vec}(s_1 s_2^T)$ allows us to write the objective (13) 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 (13) [16, Section 3.3]. When $d = 2$ the resulting SDP corresponds precisely to the one studied in Raghunathan et al. [17]. This resolves a common misconception [18] 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 (13) from that of the SDP relaxation.

Drawback. This approach includes a further relaxation step from (13), 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.

In the case $d = 2$ this relaxation provides a constant factor approximation to the original QCQP [27]. 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. The QCQP 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 [13]. Thus, the approach in section 3 and the one in this section are, in essence, the same; they only differ in the choice of polynomial positivity certificate.

Appendix E. Experimental setup details

The non-zero weights of network's i-th layer are sampled uniformly in $[-\frac{1}{\sqrt{n_i}}, \frac{1}{\sqrt{n_i}}]$ where n_i is the number of neurons in layer i.

For different configurations of width and sparsity, we generate 10 random networks and average the obtained Lipschitz bounds. For better comparison, we plot the relative error. Since we do not know the true Lipschitz constant, we cannot compute the true relative error. Instead, we take as reference the lower bound given by **LBS**. Figures 1 and 3 show the relative error, i.e., $(\hat{L} - L_{LBS})/L_{LBS}$ where L_{LBS} is the lower bound computed by **LBS** and $\hat{L}$ is the estimated upper bound.

LiPopt uses the Gurobi LP solver, while **SDP** uses Mosek. All methods run on a single machine with Core i7 2.8Ghz quad-core processor and 16Gb of RAM.