

DEEPHULL: FAST CONVEX HULL APPROXIMATION IN HIGH DIMENSIONS

Randall Balestrierio^{1*}, Zichao Wang^{2*}, Richard G. Baraniuk²

¹ Facebook AI Research, USA

² Department of Electrical and Computer Engineering, Rice University

ABSTRACT

Computing or approximating the convex hull of a dataset plays a key role in a wide range of applications, including economics, statistics, and physics, to name just a few. However, convex hull computation and approximation is exponentially complex in terms of both memory and computation as the ambient space dimension increases. In this paper, we propose DeepHull, a new convex hull approximation algorithm based on convex deep networks (DNs) with continuous piecewise-affine nonlinearities and nonnegative weights. The key idea is that binary classification between true data samples and adversarially generated samples with such a DN naturally induces a polytope decision boundary that approximates the true data convex hull. A range of exploratory experiments demonstrates that DeepHull efficiently produces a meaningful convex hull approximation, even in a high-dimensional ambient space.

Index Terms— convex hull, approximation, input convex deep network, generative adversarial network

1. INTRODUCTION

Convex hulls are important geometrical objects that find applications in fields ranging from economics [1] to statistics [2–4] and optimization [5, 6]. Given a dataset X of N samples in a D -dimensional ambient space, the convex hull is the smallest polytope that contains all of the data samples; it can easily be shown that the vertices of the convex hull correspond to some of the samples [7, 8]. Two crucial challenges arise: (i) how to efficiently compute the convex hull, and (ii) how to efficiently store the convex hull. These tasks are particularly challenging when the data is not organized on a low-dimensional affine subspace of dimension $d \ll D$. In fact, as d increases, the number of faces and vertices describing the convex hull polytope grows exponentially. This exponential complexity holds whether one considers the \mathcal{H} -form of the polytope (in term of its supporting hyperplanes) or the \mathcal{V} -form of the polytope (in term of its vertices).

* denotes equal contribution.

ZW and RichB were supported by NSF grants CCF-1911094, IIS-1838177, and IIS-1730574; ONR grants N00014-18-12571, N00014-20-1-2534, and MURI N00014-20-1-2787; AFOSR grant FA9550-18-1-0478; and a Vannevar Bush Faculty Fellowship, ONR grant N00014-18-1-2047.

The computational complications that emerge with increasing dimension d have led the combinatorial geometry community to specialize the convex hull computation task to specific cases. For example, highly efficient algorithms for planar data have been developed about four decades ago [9–11]. Among those methods lies the popular Quickhull algorithm originally developed for $d = 2, 3$ [12, 13] with asymptotic complexity $\mathcal{O}(N \log(N))$, which was then extended to arbitrary dimensions [14]. More recently, specialized GPU implementations have been developed for $d = 3$ [15–17].

Beyond these specialized algorithms, exact convex hull computation remains an important open problem in high-dimensional spaces. Consequently, a parallel line of research has developed focusing on convex hull approximation. One illustrative approximation method takes the following form. Instead of creating the hull’s polytope faces based on data selection (via the vertices description of each face), one first starts with a set of hyperplanes (which will serve as the faces of the approximate convex hull) and then refines the locations of the hyperplanes such that the intersection of their half-spaces produces a good convex hull approximation [18–20]. An inspiration for our work noted that this hyperplane learning task can be cast as a two-layer deep network (DN) training task [21]. It is easy to show that K hyperplanes can be formed from a $K \times D$ weight matrix in the DN’s first layer. That layer’s positive outputs (thanks to the application of a ReLU thresholding activation function $u \mapsto \max(u, 0)$), project data samples applied to the input of the DN onto each half-space. This 2-layer DN formulation is thus an efficient half-space projection formulation that can be employed in any of the developed convex hull approximation methods

In this paper, we go one-step further by not only using DNs as a means to reformulate the convex hull approximation problem, but also leveraging more complicated DN architectures (DNs with varying number of layers and units per layer) as a means to counter the exponential complexity in high-dimensional spaces. Our first key contribution is a proof that states for any DN architecture $f : \mathbb{R}^D \mapsto \mathbb{R}$ using (i) continuous piecewise-affine (CPA) nonlinearities and (ii) nonnegative weights in all but their first layers, the set $\{f(x) = c | x \in \mathbb{R}^D\}$ defines the boundary of a polytope, i.e., it produces a convex hull approximation. Our second key

contribution is to formalize an optimization problem that enables us to learn the parameters of DNs following the above constraints such that $\{f(x) = c | x \in \mathbb{R}^D\}$ becomes the approximated convex hull for a given dataset X . Thanks to the approximation power of DNs which grows exponentially with depth [22–26], we will be able to leverage our formulation to produce efficient approximations even in high-dimensional spaces. Our third key contribution is a relaxed form of the above optimization problem that is tractable regardless of the dataset size or space dimension. In fact, our relaxed form falls back to a binary classification problem in which one discriminates between the true data samples and adversarial samples that lie within and outside the convex hull approximation, respectively. The goal of this paper is to demonstrate that *convex hull approximation, even in high-dimensional spaces, can be tackled from a carefully designed binary classification deep learning task.*

Our results in this paper, including various visualizations and quantitative approximation results, demonstrate the promise of using DN-based methods for efficient and effective convex hull approximation, especially in high-dimensional space. We leave it to future work to develop an implementation that can be applied universally. We dub our general approach *DeepHull*. In the remainder of the paper, we first develop our approach to convex hull approximation of a dataset X via convex DNs (Sec. 2). Then, we demonstrate how convex DNs can be trained on real data via a binary classification problem and an adversarial sampler (Sec. 3). We empirically validate our method on a range of datasets (Sec. 4). We conclude by discussing the limitations of DeepHull and future research directions (Sec. 5).

2. DEEPHULL: FROM DEEP NETWORKS TO CONVEX HULLS

In this section, we develop *DeepHull*, a new convex hull approximation method that relies on two ingredients: (i) a convex DN f using continuous piecewise-affine (CPA) nonlinearities and nonnegative weights for all but its first layer; and (ii) a learned adversarial data sampler that generates positive samples to train f to discriminate against the true data samples (negative samples). We first introduce some notation to ease our development.

Notations. We denote the DN input-output mapping as $f : \mathbb{R}^D \mapsto \mathbb{R}$. In this paper, we will consider only DNs with univariate output for a reason that will become clear in the next section. This DN can be further written as a composition of L layer mappings $f = (f^{(L)} \circ \dots \circ f^{(1)})$ where $f^{(\ell)} : \mathbb{R}^{D^{(\ell)}} \mapsto \mathbb{R}^{D^{(\ell+1)}}$. At each layer ℓ , the input-output mapping takes the form $f^{(\ell)}(v) = \sigma^{(\ell)}(W^{(\ell)}v + b^{(\ell)})$ where σ is a pointwise activation function, $W^{(\ell)}$ is a weight matrix of dimensions $D^{(\ell+1)} \times D^{(\ell)}$, and $b^{(\ell)}$ is a bias vector of length $D^{(\ell+1)}$. $W^{(\ell)}$ could additionally take specific constraints, such as a circulant structure that depends on the type

of layer.

Deep Network based Convex Hull Formulation. Recall that DeepHull is designed to work with DNs that fulfill some specific constraints. These constraints result in DNs with a special property called *input convex*, which we formulate in Prop. 1.

Proposition 1 (Input Convex DNs [23,24,27]) *A DN is input convex if it obeys the following constraints*

1. *the activation functions $\sigma^{(\ell)}, \forall \ell$ are CPA functions; the inner (i.e. all but the first) activation functions $\sigma^{(\ell)}, \ell = 2, \dots, L$ are increasing functions (e.g., leaky-ReLU);*
2. *the inner layer weight matrices $W^{(\ell)}, \ell = 2, \dots, L$ are nonnegative, the first slope matrix $W^{(1)}$ is arbitrary*

that is, the DN is a convex mapping with respect to its input.

The above result holds for *strict* convexity by replacing the increasing activation with a *strictly* increasing activation and the nonnegative slope matrices with *strictly positive* slope matrices. Input convex DNs have been applied for control problems [27], where the convexity property enabled the simplification of the gradient based optimization of the DN input. DeepHull relies heavily on input convex DNs that employ CPA nonlinearities for a reason that is made clear in the following formal result.

Proposition 2 *For any architecture and parameters of an input convex DN f , the set $\{f(x) = c : x \in \mathbb{R}^D\}$ defines the boundary of a polytope. The sets $\{f(x) < c | x \in \mathbb{R}^D\}$ and $\{f(x) > c | x \in \mathbb{R}^D\}$ are the interior and exterior of the polytope, respectively.*

Input convex DNs with CPA nonlinearities thus have the capability to approximate the convex hull of a dataset X . This is done by finding the parameters θ of the DN such that the DN-induced polytope contains all the data samples while minimizing its volume as in

$$\min_{\theta, c} \overbrace{\text{Vol}(\{f(x) < c : x \in \mathbb{R}^D\})}^{\text{volume minimization}} \quad \text{s.t. } \underbrace{X \subset \{f(x) \leq c : x \in \mathbb{R}^D\}}_{\text{dataset inclusion}}. \quad (1)$$

Theorem 1 *Given a DN f with enough layers/units, all the local minima of (1) are global minima and result in $\{f(x) = c : x \in \mathbb{R}^D\}$ being the exact convex hull of X .*

The proof of the Theorem 1 follows easily by considering f to be able to represent decision boundaries with as many piecewise linear regions as needed for the convex hull of X . In that setting, minimizing (1) simply amounts in adapting the decision boundary such that it perfectly matches with the true data convex hull. Of course, this optimization problem is not practical, since $\text{Vol}(\{f(x) < c : x \in \mathbb{R}^D\})$ would require a tremendous amount of computation to obtain. We thus propose a training method to obtain f and its parameters θ, c on a relaxed optimization problem in the next section.

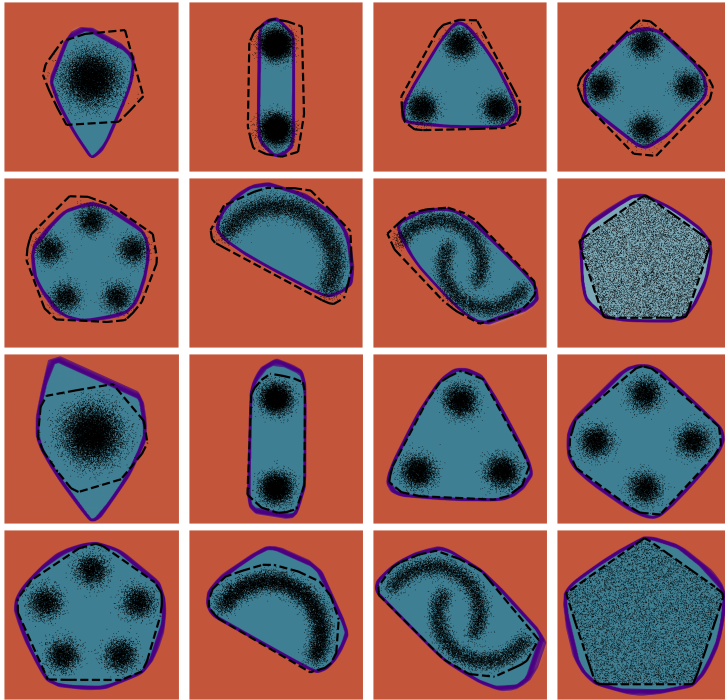


Fig. 1. DeepHull examples in 2-dimensions of datasets obtained by sampling from 1 to 5 Gaussians, 1 to 2 moons and uniform over a pentagon. The **top two rows** demonstrate the exact convex hull in black dashed line and our approximation in purple when using the loss from (5) with an hyper-parameter value of $\lambda = 1$. The **bottom two rows** demonstrate the exact same setting but now with a larger hyperparameter $\lambda = 1.5$. In both cases the approximation captures the geometry of the data; the value of λ controls the tightness of the approximation, potentially at the cost of disregarding a few of the dataset outliers when a tighter approximation for the majority of the remaining samples is possible. We do not consider automatic selection of λ in this study, however it is clear that one could increase/decrease its value during training to obtain the greatest possible value of λ that does not cause data samples to exit the convex hull approximation.

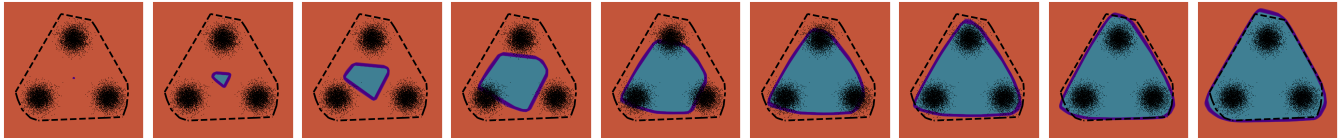


Fig. 2. Visualization of how the decision boundary changes during training at the 0, 25, 50, 75, 100, 125, 300, 1500, and 5000 epochs. Our model converges quickly gives a good approximation even at early stages of the training.

Table 1. Precision and recall of DeepHull’s approximation.

D=3		D=4		D=5		D=6		D=7	
P	R	P	R	P	R	P	R	P	R
92.1	92.0	85.8	85.5	79.2	78.4	79.2	79.2	77.8	77.8

3. DEEPHULL FITTING: BINARY CLASSIFICATION

We demonstrated in the last section how by applying simple constraints on any given DN architecture one could obtain an input convex DN form which level sets define polytope boundaries. Given that approximator, we train it such that the polytope boundaries match as closely as possible to true data convex hull. We now construct a binary classification problem to solve this task efficiently.

Relaxed Dataset Inclusion Loss. From Sec. 2, it is clear that the convex hull approximator, f , must fulfill $f(x) < c, \forall x \in X$ where we recall that X is the training set, i.e., the set of samples for which we try to approximate the convex hull. For the remaining of this paper, we consider the last

layer to have linear activation function ($\sigma^{(L)}(u) = u$), and we incorporate the constant c as part of the last layer bias as in $b^{(L)} \leftarrow b^{(L)} - c$. Given that parametrization, one differentiable loss that can be used to enforce data inclusion is given by

$$\mathcal{L}_{\text{pos}}(x) = -\log(1 - \text{sigmoid}(f(x))). \quad (2)$$

As a result, as $\frac{1}{N} \sum_{n=1}^N \mathcal{L}_{\text{pos}}(x) \rightarrow 0$ as the approximated convex hull ($\{f(x) < 0 : x \in \mathbb{R}^D\}$) contains all the training data. Minimizing (2) is however not enough since it does not enforce tightness of the approximation, i.e., the convex hull approximation can cover more and more space and still minimize (2).

Relaxed Volume Minimization Loss. We also introduce the following relaxed version of the volume minimization term from (1) as

$$\mathcal{L}_{\text{neg}}(z) = -\log(\text{sigmoid}(f(z))), \quad (3)$$

and obtain the following total loss

$$\mathcal{L} = \frac{1}{N} \sum_{n=1}^N \mathcal{L}_{\text{pos}}(x) + \lambda \mathbb{E}_{z \sim U(\mathbb{R}^D)} [\mathcal{L}_{\text{neg}}(z)], \quad (4)$$

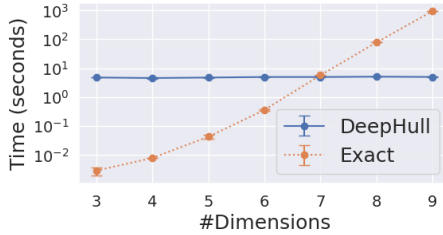


Fig. 3. Computation time for the proposed DeepHull approximation algorithm and for exact convex hull computation

where λ is a hyperparameter controlling the tightness. As samples z are sampled from the ambient space, as the DN decision boundary will be refined to contain the samples X , but as few as possible of anything else in order to minimize the second term. If λ is too large, however, then the DN will start disregarding some of the samples in X while on the other hand if λ is too small, then the optimal tightness might not be achieved. We will perform an ablation study of this parameter in Sec. 4.

We propose one last alteration to (4) to further improve its efficiency in high-dimensional settings. Note that we do not need to sample uniformly in \mathbb{R}^D to ensure tightness. Instead, we only need to be able to sample around the boundary of the current convex hull approximation. This is also true in term of gradient dynamics as samples z position far away from the current convex hull boundary approximation will have vanishing gradient from (2) and thus will not impact the update of the DN weights. Consequently, we propose our final loss function

$$\mathcal{L} = \underbrace{\frac{1}{N} \sum_{n=1}^N \mathcal{L}_{\text{pos}}(x)}_{\text{dataset inclusion}} + \underbrace{\lambda \mathbb{E}_{z \sim G} [\mathcal{L}_{\text{neg}}(z)]}_{\text{tightness loss around approximated boundary}} + \underbrace{\mathbb{E}_{z \sim U(\mathbb{R}^d)} [\text{sigmoid}(f(G(z)))]}_{\text{adversarial training of distribution } G}, \quad (5)$$

4. EXPERIMENTAL VALIDATION

We now report on a series of carefully controlled experiments that validate and illustrate the behavior of DeepHull.

Role of λ and Training Dynamics. We first propose in Fig. 1 a collection of 2-dimensional datasets where we study the impact of the hyper-parameter λ (recall (5)). The role of this parameter is to ensure that the convex hull approximation is not degenerate, as in, the approximation includes the entire space. If this parameter is too large, however, the approximation will start disregarding some of the samples if it can allow to have a tighter approximation for the majority of the remaining samples. While out of the scope of this paper, this could open the door to further application in convex hull approximation in the presence of outliers. We depict in Fig. 2 how the approximation is progressively built through the training updates of the DN. We can see how in the first stages the approximation is degenerate around 0. This is because we chose to keep the DN initialization as done in usual

classification tasks, with $b^{(L)} = 0$. Once the approximation expands to include the training samples, we see that the tightness loss term (recall (3)) takes effect and prevents the convex hull to keep expanding beyond that point. We believe that one important question is to design DN initialization of the parameters $W^{(\ell)}, b^{(\ell)}, \forall \ell$ that will provide more adapted initial guess on the convex hull approximation.

DeepHull in Higher Dimensions. We also validate our approach for higher dimensions and demonstrate that our approach has a stable computational time while achieving reasonable convex hull approximation. We randomly sample 100,000 points from a isotropic multivariate Gaussian $\mathcal{N}(\mathbf{1}, \Sigma)$ where $\mathbf{1} \in \mathbb{R}^D$ and $\Sigma \in \mathbb{R}^{D \times D}$ is a diagonal matrix with 0.01 on the diagonal. We vary $D \in \{3, 4, 5, 6, 7, 8, 9\}$ and measure the goodness of the DeepHull approximation and compares DeepHull’s computation time with the exact convex hull method. Going beyond $D = 9$ makes the exact convex hull computation highly prohibitive. Because it is impossible to visualize a convex hull in high dimensions, we compute both **precision** and **recall** to measure the tightness and coverage of the approximation, respectively, by uniformly sampling 500,000 points in the D -dimensional space, computing the number of points in the ground-truth convex hull [14], and the number of points in the approximated convex hull. Table 1 shows DeepHull’s convex hull approximation performance for $D \in \{3, 4, 5, 6, 7\}$. We omit $D \in \{8, 9\}$ because computing whether a point belongs to a convex hull becomes computationally infeasible for $D > 7$ on our hardware. We can see that DeepHull maintains reasonable performance, indicating that the approximation is tight and covers most of the ground-truth hull. In Fig. 3 we compare the DeepHull’s computation time with the exact method. We observe that as D increases, the exact method requires nearly exponentially more compute time, while DeepHull’s compute time remains nearly constant. This observation is particularly appealing in practice, because DeepHull gives an efficient and decent approximation to computing convex hull, which becomes computationally infeasible for classical, exact methods.

5. CONCLUSIONS

We have opened the door to the potential use of DNs for the important but expensive task of convex hull approximation. Our preliminary results with DeepHull demonstrate the validity of the approach while opening many future avenues to improve those results and obtain an approximation solution with theoretical guarantees. Future research directions include better-suited DN parameter initialization, adaptive hyper-parameter (λ) tuning, application to convex hull approximation in the presence of outliers, and approximation error guarantees between the produced convex hull and the ground truth one.

6. REFERENCES

- [1] Bowen Hua and Ross Baldick, “A convex primal formulation for convex hull pricing,” *IEEE Trans. Power Syst.*, vol. 32, no. 5, pp. 3814–3823, 2016.
- [2] William F Eddy, “Convex hull peeling,” in *COMPSTAT Symp.*, 1982, pp. 42–47.
- [3] Tom F Wilderjans, Eva Ceulemans, and Kristof Meers, “Chull: A generic convex-hull-based model selection method,” *Behav. Res. Methods*, vol. 45, no. 1, pp. 1–15, 2013.
- [4] Alexander K Hartmann, Satya N Majumdar, Hendrik Schawe, and Grégory Schehr, “The convex hull of the run-and-tumble particle in a plane,” *J. Statistical Mechanics Theory and Experiment*, vol. 2020, no. 5, pp. 053401, 2020.
- [5] Thomas Lachand-Robert and Édouard Oudet, “Minimizing within convex bodies using a convex hull method,” *SIAM J. on Optim.*, vol. 16, no. 2, pp. 368–379, 2005.
- [6] Samy Missoum, Palaniappan Ramu, and Raphael T Haftka, “A convex hull approach for the reliability-based design optimization of nonlinear transient dynamic problems,” *Comput. Methods Appl. Mechanics Eng.*, vol. 196, no. 29–30, pp. 2895–2906, 2007.
- [7] Herbert Edelsbrunner, *Algorithms in Combinatorial Geometry*, vol. 10, Springer Science & Business Media, 1987.
- [8] János Pach and Pankaj K Agarwal, *Combinatorial geometry*, vol. 37, John Wiley & Sons, 2011.
- [9] William F Eddy, “A new convex hull algorithm for planar sets,” *ACM Trans. on Math. Softw.*, vol. 3, no. 4, pp. 398–403, 1977.
- [10] David G Kirkpatrick and Raimund Seidel, “The ultimate planar convex hull algorithm?,” *SIAM J. Comput.*, vol. 15, no. 1, pp. 287–299, 1986.
- [11] Gerth Stølting Brodal and Riko Jacob, “Dynamic planar convex hull,” in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, 2002, pp. 617–626.
- [12] Jonathan Scott Greenfield, “A proof for a quickhull algorithm,” 1990.
- [13] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa, “The quickhull algorithm for convex hull,” Tech. Rep., Tech. Report GCG53, Geometry Center, Univ. of Minnesota, 1993.
- [14] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa, “The quickhull algorithm for convex hulls,” *ACM Trans. on Math. Softw.*, vol. 22, no. 4, pp. 469–483, 1996.
- [15] Ayal Stein, Eran Geva, and Jihad El-Sana, “Cudahull: Fast parallel 3D convex hull on the GPU,” *Comput. & Graph.*, vol. 36, no. 4, pp. 265–271, 2012.
- [16] Stanley Tzeng and John D Owens, “Finding convex hulls using quickhull on the GPU,” *arXiv preprint arXiv:1201.2936*, 2012.
- [17] Mingcen Gao, Thanh-Tung Cao, Ashwin Nanjappa, Tiow-Seng Tan, and Zhiyong Huang, “gHull: A GPU algorithm for 3D convex hull,” *ACM Trans. on Math. Softw.*, vol. 40, no. 1, pp. 1–19, 2013.
- [18] Jon Louis Bentley, Franco P Preparata, and Mark G Faust, “Approximation algorithms for convex hulls,” *Commun. ACM*, vol. 25, no. 1, pp. 64–68, 1982.
- [19] Ladislav Kavan, Ivana Kolingerova, and Jiri Zara, “Fast approximation of convex hull,” *Advances Comput. Sci. Technol.*, vol. 6, pp. 101–104, 2006.
- [20] Hossein Sartipizadeh and Tyrone L Vincent, “Computing the approximate convex hull in high dimensions,” *arXiv preprint arXiv:1603.04422*, 2016.
- [21] Yee Leung, Jiang-She Zhang, and Zong-Ben Xu, “Neural networks for convex hull computation,” *IEEE Trans. Neural Networks*, vol. 8, no. 3, pp. 601–611, 1997.
- [22] Guido Montúfar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio, “On the number of linear regions of deep neural networks,” *arXiv preprint arXiv:1402.1869*, 2014.
- [23] Randall Balestriero et al., “A spline theory of deep learning,” in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 374–383.
- [24] Randall Balestriero and Richard G. Baraniuk, “Mad max: Affine spline insights into deep learning,” *Proc. IEEE*, vol. 109, no. 5, pp. 704–727, 2021.
- [25] Michael Unser, “A representer theorem for deep neural networks,” *J. Mach. Learn. Res.*, vol. 20, no. 110, pp. 1–30, 2019.
- [26] Ingrid Daubechies, Ronald DeVore, Simon Foucart, Boris Hanin, and Guergana Petrova, “Nonlinear approximation and (deep) ReLU networks,” *Constructive Approximation*, pp. 1–46, 2021.
- [27] Brandon Amos, Lei Xu, and J Zico Kolter, “Input convex neural networks,” in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 146–155.