ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-6654-0540-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICASSP43922.2022.9746031

DEEPHULL: FAST CONVEX HULL APPROXIMATION IN HIGH DIMENSIONS

Randall Balestriero [1*], *Zichao Wang* [2*], *Richard G. Baraniuk* [2]

[1] Meta AI Research, USA
[2] Department of Electrical and Computer Engineering, Rice University

ABSTRACT

Computing or approximating the convex hull of a dataset plays a 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 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, 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 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).

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 that was originally developed for $d = 2, 3$ [12, 13] with asymptotic complexity $\mathcal{O}(N \log(N))$, and 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 has 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 convex hull approximation method.

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 numbers of layers and units per layer) as a means to counter the exponential complexity of convex hull approximation in high-dimensional spaces. We dub our general approach *DeepHull*. Our first contribution is a proof that 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 layer, the level-set $\{f(x) = c \mid x \in \mathbb{R}^D\}$ defines the boundary of a polytope,

* 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.

Authorized licensed use limited to: Rasmussen College. Downloaded on June 13,2022 at 18:38:13 UTC from IEEE Xplore. Restrictions apply.

i.e., it produces a convex hull approximation. Our second contribution is to formalize an optimization problem that enables us to learn the parameters of DNs abiding by the above constraints such that $\{f(x) = c \mid x \in \mathbb{R}^D\}$ becomes the approximated convex hull for a given dataset X. Thanks to the approximation power of DNs, which grows exponentially with depth [22–25], our formulation produces efficient approximations even in high-dimensional spaces. Our third contribution is a relaxed form of the above optimization problem that is tractable regardless of the dataset size or space dimension. Our relaxed form *recasts convex hull approximation as 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.

Our results in this paper, including a number of 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. 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 in 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.

Notation. 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 written as a composition of L layer mappings $f = (f^{(L)} \circ \cdots \circ f^{(1)})$, 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 $D^{(\ell+1)}$. Certain $W^{(\ell)}$ will often have a specific structure (e.g., circulant) at different layers.

Deep Network based Convex Hull Formulation. Recall that DeepHull is designed to work with DNs that fulfill 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, 26]) *A DN is a convex mapping with respect to its input (which we will refer to as* input-convex*) if it obeys the following constraints:*

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

The above result holds for *strict convexity* by replacing the increasing activation with a strictly increasing activation function and the nonnegative weight matrices with strictly positive weight matrices. Input-convex DNs have been applied in control problems [26], 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 $\{f(x) < c \mid x \in \mathbb{R}^D\}$ and $\{f(x) > c \mid x \in \mathbb{R}^D\}$ 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

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

Theorem 1 *Given a DN f with sufficiently many layers/units, all the local minima of (1) are global minimia 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 to adapting the decision boundary such that it perfectly matches with the true data convex hull. Of course, this optimization problem is not practical, since $\mathrm{Vol}(\{f(x) < c : x \in \mathbb{R}^D\})$ 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.

Authorized licensed use limited to: Rasmussen College. Downloaded on June 13,2022 at 18:38:13 UTC from IEEE Xplore. Restrictions apply.

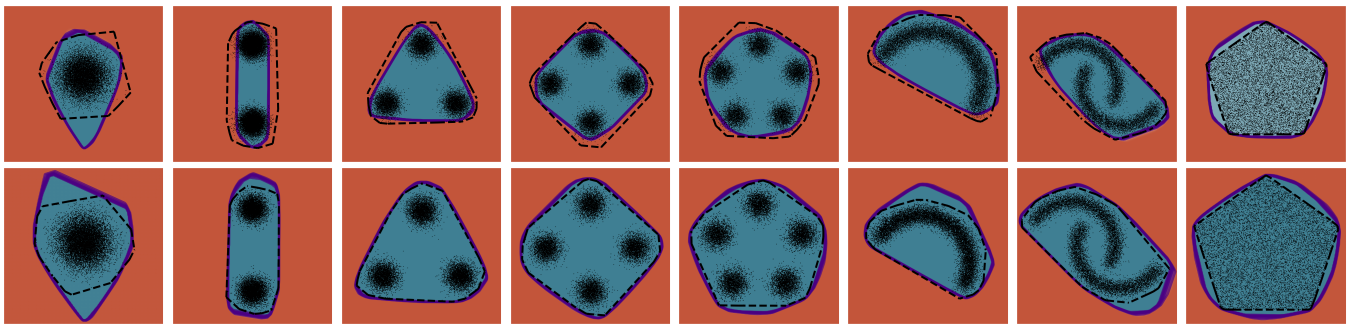

Fig. 1. DeepHull examples for various 2-dimensional datasets uniformly sampled from eight distributions: a Gaussian, mixtures of 2 to 5 Gaussians, 1 to 2 moons, and a pentagon. The black dashed line represents the exact convex hull, while the purple line represents DeepHull approximated convex hull. The plots in the **top row** use $\lambda = 1$ in the loss (5). The plots in the **bottom row** use $\lambda = 1.5$ in the loss (5). In both cases, the DeepHull approximation captures the geometry of the data. λ controls the tightness of the approximation, potentially at the cost of disregarding a few of the data samples that lie outside the approximated convex hull when a tighter approximation fits the majority of the remaining samples. One can easily tune λ to obtain an approximation that includes all data samples in the resulting approximated convex hull. Methods for automatically tuning λ are left for future work.

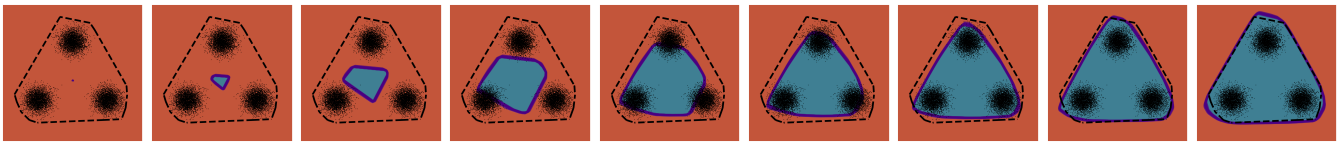

Fig. 2. Visualization of the decision boundary/convex hull approximation during training at the 0, 25, 50, 75, 100, 125, 300, 1500, and 5000-th epochs. The DeepHull model converges quickly and provides a good approximation even at early stages of training.

3. EFFICIENT DEEPHULL FITTING VIA BINARY CLASSIFICATION

We demonstrated in the previous section how imposing simple constraints on any given DN architecture leads to an input-convex DN whose level sets define polytope boundaries. Given an input-convex DN, we train it such that the polytope boundaries match as closely as possible to the 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 X is the training set, i.e., the set of samples for which we try to approximate the convex hull. For the remainder 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 the above parametrization, we use the following differentiable loss function to enforce data inclusion

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

As a result, as $\frac{1}{N}\sum_{n=1}^{N}\mathcal{L}_{\text{pos}}(x) \to 0$, the approximated convex hull ($\{f(x) < 0 : x \in \mathbb{R}^D\}$) contains all of 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)

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

to obtain the total loss

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

where λ is an hyper-parameter that controls the tightness of the approximated convex hull. As samples z are sampled from the ambient space, the DN decision boundary will be trained to contain the samples X to minimize the first term in (4) and exclude anything else to minimize the second term. The hyper-parameter λ needs to be chosen carefully: if λ is too large, then the DN will start to exclude some of the samples in X; if λ is too small, then the approximated convex hull might not be tight enough. We will visualize the effect of λ in Sec. 4.

We propose one last modification 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 because samples z positioned far away from the boundary of the current convex hull approximation will have vanishing gradient from (2) and thus will not impact the update of the DN weights. Consequently, our final

Authorized licensed use limited to: Rasmussen College. Downloaded on June 13,2022 at 18:38:13 UTC from IEEE Xplore. Restrictions apply.

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

loss function takes the form

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

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 λ is to ensure that the convex hull approximation neither degenerates, i.e., it covers the entire space, nor underfits, i.e., it excludes many samples that should belong to the convex hull. With a carefully chosen λ, we see that we achieve a good approximation, while λ too small leads to underfitting. While it is out of the scope of this paper, enabling underfitting could open the door for new convex hull approximation applications, e.g., robust to outliers.

We depict in Fig. 2 how the convex hull approximation is progressively built through the training updates of the DN. In the first stages the approximation, the approximation is degenerate around 0, because we chose to keep initialize the DN with $b^{(L)} = 0$ (standard practice). 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 from expanding beyond that point. One interesting open question is the design of a parameter initialization for $W^{(\ell)}, b^{(\ell)}$ that provides a more adapted initial guess for the convex hull approximation.

DeepHull in Higher Dimensions. We validate our approach in higher dimensions and demonstrate that it has a stable computation time while achieving reasonable convex hull approximation. We randomly sample 100,000 points from an 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 its computation time with the exact convex hull method. Going beyond $D = 9$ makes the exact convex hull computation highly prohibitive, precluding comparison. Since it is impossible to visualize a convex hull in higher dimensions, we compute the **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 summarizes DeepHull's convex hull approximation performance for $D \in \{3, 4, 5, 6, 7\}$; we do not report performance for $D \in \{8, 9\}$, because computing whether a point belongs to a convex hull becomes computationally infeasible for $D > 7$ on our hardware. We see that DeepHull maintains reasonable performance for all D, 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 convex hull computation method. We observe that as d (and D) increases, the exact method requires nearly exponentially more computation time, while DeepHull's computation time remains nearly constant. Additionally, our method also benefits from linear computational complexity with respect to the number of samples N. This can even be made constant via mini-batching, in both cases we shall highlight that current approximation methods for high dimension are quadratic in N and constant in d [20]. In summary, DeepHull can efficiently compute a decent approximation to convex hulls that are computationally infeasible with classical, exact methods.

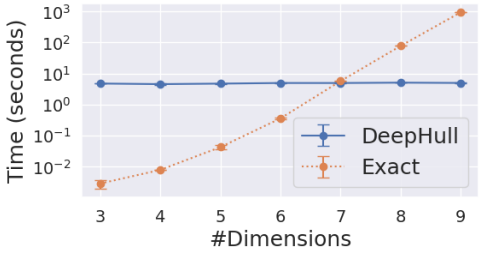

Fig. 3. Computation time of DeepHull vs. the exact convex hull method (QHull implementation of Quickhull [27]).

5. CONCLUSIONS

This paper has opened the door to the potential use of DNs for the important but computationally expensive task of convex hull approximation. Our preliminary results with DeepHull demonstrate the validity of our approach and demonstrated that linear in N and constant in d solutions exist. While thorough comparisons are needed to pinpoint the exact beneficial use-cases of our method against current approximation methods (quadratic in N and constant in d), we believe that many future research directions should first be explored. For example, include adaptive DN parameter initialization, adaptive hyper-parameter (λ) tuning, convex hull approximation in the presence of outliers, and approximation error guarantees between the approximated and true convex hull.

Authorized licensed use limited to: Rasmussen College. Downloaded on June 13,2022 at 18:38:13 UTC from IEEE Xplore. Restrictions apply.

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] Brandon Amos, Lei Xu, and J Zico Kolter, "Input convex neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 146–155.

[27] C Bradford Barber, David P Dobkin, and Hannu Huhdanpaa, "Qhull: Quickhull algorithm for computing the convex hull," *Astrophysics Source Code Library*, pp. ascl–1304, 2013.

Authorized licensed use limited to: Rasmussen College. Downloaded on June 13,2022 at 18:38:13 UTC from IEEE Xplore. Restrictions apply.