Gradient Networks

Shreyas Chaudhari, *Student Member, IEEE*, Srinivasa Pranav, *Student Member, IEEE*, and José M.F. Moura, *Life Fellow, IEEE*

***Abstract*—Directly parameterizing and learning gradients of functions has widespread significance, with specific applications in inverse problems, generative modeling, and optimal transport. This paper introduces gradient networks (`GradNets`): novel neural network architectures that parameterize gradients of various function classes. `GradNets` exhibit specialized architectural constraints that ensure correspondence to gradient functions. We provide a comprehensive `GradNet` design framework that includes methods for transforming `GradNets` into monotone gradient networks (`mGradNets`), which are guaranteed to represent gradients of convex functions. Our results establish that our proposed `GradNet` (and `mGradNet`) universally approximate the gradients of (convex) functions. Furthermore, these networks can be customized to correspond to specific spaces of potential functions, including transformed sums of (convex) ridge functions. Our analysis leads to two distinct `GradNet` architectures, `GradNet-C` and `GradNet-M`, and we describe the corresponding monotone versions, `mGradNet-C` and `mGradNet-M`. Our empirical results demonstrate that these architectures provide efficient parameterizations and outperform existing methods by up to 15 dB in gradient field tasks and by up to 11 dB in Hamiltonian dynamics learning tasks.**

***Index Terms*—Neural networks, learning gradients, convex, monotone, universal gradient function approximation, subgradients, gradient fields, Hamiltonian mechanics.**

I. Introduction

DEEP neural networks are prized for their ability to parameterize and easily learn complicated, high-dimensional functions. Researchers have devoted substantial effort to developing deep neural networks that achieve state-of-the-art performance on numerous tasks spanning computer vision [1], natural language processing [2], and reinforcement learning [3]. These neural networks are commonly unconstrained and effectively parameterize the space of all functions. Many applications instead require learned functions that exhibit specific properties, which necessitates the design of neural networks corresponding to specific function classes – a problem that has seldom been studied. Constraining neural networks to belong to a particular function class not only enhances interpretability, but also leads to theoretical performance guarantees essential for deploying trained models in safety-critical applications.

In particular, neural networks corresponding to *gradients* of functions hold significant importance across several science and engineering disciplines. For example, physicists often wish to use a finite set of measurements to characterize the gradient field of a potential function, such as the temperature gradient over a surface. Score-based generative models are another application of learning gradient functions, where neural networks are trained to learn $\nabla_{\boldsymbol{x}} \log p(x)$, the score function of an unknown probability distribution [4], [5]. These methods have been applied to image [6], [7] and 3D shape [8] generation.

Learning gradients of *convex* functions holds particular significance in optimal transport theory. Brenier's theorem states that the unique solution to the Monge problem with Euclidean cost is given by the gradient of a convex function [9], [10]. The theorem has inspired methods for learning Monge maps [11], [12], [13], [14] using gradients of parameterized convex functions. Applications of learned gradients of convex functions also extend to gradient-based optimization. Optimization routines can incorporate a learned gradient function to define a set of iterative updates that map an input to a desired output [15]. In particular, gradients of convex functions can be used to define gradients of regularization terms in optimization objectives, as in the regularization by denoising (RED) [16], [17] and plug-and-play (PnP) [18], [19] frameworks for solving inverse problems. In order to enhance interpretability and obtain convergence guarantees, recent works have designed denoisers for these frameworks that have symmetric Jacobians [20], [21], [22], thereby ensuring that the denoisers are gradient functions.

Previously in [14], we considered learning gradients of convex functions and proposed monotone gradient networks. In this paper, we generalize the problem setting in [14] to consider significantly broader function classes. We introduce new `GradNet` architectures for learning gradients of these function classes, provide extensive theoretical analysis concerning `GradNet` universal approximation capabilities, and validate our methods with extensive experiments.

Contributions: We propose gradient networks (`GradNets`), neural networks for directly parameterizing gradients of arbitrary functions. We adapt `GradNets` to monotone gradient networks (`mGradNets`) that correspond to gradients of convex functions. Our proposed neural network architectures directly model the gradient function ∇F without first learning the underlying potential function F. We also

Received 16 March 2024; revised 27 August 2024 and 18 October 2024; accepted 26 October 2024. Date of publication 13 November 2024; date of current version 8 January 2025. This work was supported in part by the NSF Graduate Research Fellowships under Grant DGE-1745016 and Grant DGE-2140739, in part by the NSF Grant CCF-2327905, and in part by the ARCS Fellowship. The associate editor coordinating the review of this article and approving it for publication was Thomas Oberlin. *(Shreyas Chaudhari and Srinivasa Pranav contributed equally to this work.) (Corresponding author: Shreyas Chaudhari.)*

The authors are with the Electrical and Computer Engineering Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: shreyasc@andrew.cmu.edu).

Digital Object Identifier 10.1109/TSP.2024.3496692

1053-587X © 2024 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies.
Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

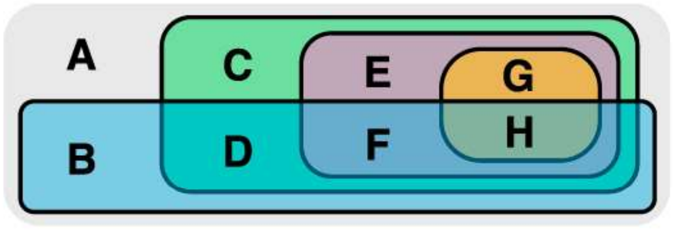


Fig. 1. Relationships between relevant function classes considered in corresponding theory sections of this paper: (a) all functions; (b) monotone functions; Section V-A: (c) gradients, (d) monotone gradients; Section V-C: (e) gradients of transformed sums of ridges, (f) gradients of transformed sums of *convex* ridges; Section V-B: (g) gradients of sums of ridges, (h) gradients of sums of *convex* ridges; Section V-D: respective subgradients.

analyze the expressivity of our proposed `GradNet` and `mGradNet` networks and prove that they are universal approximators of gradients of general and convex functions, respectively. We further describe methods for modifying these networks to represent popular subsets of these function classes, including gradients of sums of (convex) ridge functions and their (convexity-preserving) transformations. Our theoretical results translate into improved experimental results over existing works. In our gradient field approximation experiments, we find that our architectures achieve up to 15 dB lower mean squared error (MSE) than existing methods. Additionally, we predict the dynamics of the two-body problem and demonstrate an 11 dB improvement over existing methods.

Paper Organization: We first review existing methods for parameterizing and learning gradients of functions in Sec. II. We then present *gradient networks* (`GradNets`) for learning gradients of arbitrary functions in Sec. III. In Sec. IV, we describe *monotone gradient networks* (`mGradNets`): modified `GradNets` for representing gradients of convex functions. We analyze the expressivity of our proposed `GradNets` and `mGradNets` in Sec. V and demonstrate methods for composing `GradNets` and `mGradNets` to learn (sub)gradients of the function classes described in Fig. 1. We present architectural modifications in Sec. VI that empirically yield efficient parameterizations. Finally, in Sec. VII, we evaluate our proposed models on gradient field and Hamiltonian dynamics learning tasks.

Notation: w and W respectively denote vectors and matrices. $\nabla F(x)$ is the gradient of a potential function F at a point x and is a vector-valued function. $\boldsymbol{H}_F(x)$ is the Hessian of F at x, the matrix of second-order partial derivatives. When a function f is vector-valued, $\boldsymbol{J}_f(x)$ denotes the Jacobian of f at x, the matrix of partial derivatives. $\boldsymbol{A} \succeq 0$ indicates that $\boldsymbol{A}$ is symmetric positive semidefinite (PSD). $\operatorname{diag}(\cdot)$ is the vector-to-diagonal-matrix operator. $C^0(\mathcal{D})$ and $C^k(\mathcal{D})$ respectively denote the space of continuous and k-times continuously differentiable functions that map from domain $\mathcal{D}$ to the real numbers $\mathbb{R}$. $C^k_\times(\mathcal{D})$ are the *convex* functions in $C^k(\mathcal{D})$.

II. Related Work

Several existing works use standard neural networks to directly model the gradient of a potential function [7], [17], [23], [24]. While these methods demonstrate satisfactory empirical performance, they lack theoretical justification and are often considered heuristic approaches. It is not guaranteed, and in fact highly unlikely, that an arbitrary neural network with matching input and output dimension corresponds to the gradient of a scalar-valued function. More formally, let $\mathcal{F}$ be a set such that each function $F \in \mathcal{F}$ is the gradient of a scalar-valued function. Given an $F \in \mathcal{F}$, standard neural network architectures, e.g., multilayer perceptrons (MLPs) and convolutional neural networks (CNNs), with nonpolynomial activations, can be used to approximate F. However, these architectures offer no guarantee that the learned function $\widehat{F}$ is itself in $\mathcal{F}$. In contrast, the methods and network architectures discussed in this paper ensure that the approximator $\widehat{F}$ is always a member of $\mathcal{F}$, thereby enhancing interpretability and enabling robust theoretical guarantees in practice.

Rather than directly approximating the gradient of a function using a standard neural network, an alternative approach may be to first parameterize and learn the underlying scalar potential function using a standard neural network. Subsequently differentiating the network with respect to its input yields the desired gradient. However, closely approximating a target function F with an arbitrary approximator G does not guarantee that ∇G closely approximates ∇F [25]. Empirical evidence in [26] confirms this fact when the approximating functions are neural networks. Other works directly train the gradient map of a neural network that parameterizes a potential function. For instance, [21] considers the potential $F(x) = \frac{1}{2}\|x - f_{\boldsymbol{\theta}}(x)\|_2^2$ with neural network $f_{\boldsymbol{\theta}} : \mathbb{R}^n \to \mathbb{R}^n$ and uses automatic differentiation to obtain a gradient network architecture $\nabla F(x)$. Nevertheless, these approaches often exhibit ill-behaved product structures in practice [27] and, due to the Runge phenomena [28], optimizing their parameters becomes cumbersome as the input dimension grows [29].

Feedforward neural networks have also been proposed to directly parameterize gradients of specific classes of potential functions, namely potentials $F(x)$ expressible as the sum of ridge functions: $F(x) = \sum_{i=1}^n \psi_i(w_i^\top x)$. Such potentials were considered in the *fields of experts* framework proposed in [30]. The fields of experts framework has inspired several methods for learning image priors that effectively generalize the "transform domain thresholding" approach discussed in [17], [31]. These include works like [32], [33] that correspond to nonconvex potentials and respectively achieve commendable performance for image restoration and magnetic resonance imaging reconstruction. More recently, [34], [35] propose neural networks with elementwise, learnable spline activations to learn gradients of potentials expressible as sums of ridge functions (class G in Fig. 1). The networks can be modified to correspond to gradients of sums of *convex* ridge functions (class H in Fig. 1) by restricting the splines to be nondecreasing. In this paper, we consider parameterizing the activations as learnable neural networks and discuss how these networks can be adapted to represent (sub)gradients of convex potentials. Furthermore, the class of functions expressible as a sum of convex ridges as in [34], [35] does not include several

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

common convex functions including

$$\text{Infinity Norm}: \max\{|x_1|, |x_2|, \ldots, |x_n|\}$$
$$\text{Exponential Product}: \exp\left(\sum_{i=1}^{n} x_i\right)$$

Thus, we also use group activations to yield universal approximation of gradients of (convex) function classes that extend beyond those expressible as sums of ridges.

The literature on parameterizing and learning the gradients of convex functions (monotone gradient functions) includes various approaches. Input Convex Neural Networks (ICNN) [36] parameterize only convex functions by requiring positive weight matrices and convex, monotonically increasing elementwise activation functions. To extract the gradient, [13], [37], [38] use a two-step approach of first evaluating a convex potential parameterized by an ICNN and then differentiating the network with respect to its input (via backpropagation). While [38] provides theoretical justification for this approach, the ICNN's architectural restrictions result in well-documented training difficulties in practice [11], [29], [39], [40]. In this paper, we avoid these challenges and bypass the two-step process by directly characterizing and learning the gradient.

Input Convex Gradient Networks (ICGNs) [29] extend the approach in [41] and characterize the gradient of twice-differentiable convex functions by exploiting the symmetric positive semidefinite structure of their Hessians. ICGNs specifically parameterize a Gram factor of the Hessian. The gradient map is then obtained by numerically integrating the learned Hessian. However, as noted in [29], the only known architecture suitable for parameterizing the Gram factor is $\sigma(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{b})$, for which numerical integration is not required and serves only as a proof of concept. These architectures, for which numerical integration is not required, parameterize gradients of sums of convex ridges (class H in Fig. 1). Meanwhile, in this work we consider parameterizing functions for several other larger classes of functions, including transformations of sums of (convex) ridges, differentiable convex functions, and L-smooth nonconvex functions. Additionally, the existence of deeper networks suitable for parameterizing the Gram factor of the ICGN is conjectured in [29], but have not been identified to date. Numerical integration for deeper networks would also introduce computational challenges in high dimensions (e.g., error accumulation, convergence) [42].

III. Gradient Networks (GradNet)

In this section, we introduce *gradient networks* (`GradNets`) for parameterizing and learning gradients of continuously differentiable functions. We first state sufficient conditions for a neural network to be a `GradNet` and then discuss various approaches for constructing `GradNets`.

Definition 1 (`GradNet`): A gradient network (`GradNet`) is a neural network f satisfying $f = \nabla F$ for some $F \in C^1\left(\mathbb{R}^d\right)$.

By Def. 1, differentiating a neural network that parameterizes a function $F \in C^1(\mathbb{R}^d)$ yields a `GradNet`. However, due to the discussion in Sec. II, we focus on directly modeling and learning ∇F (without first learning F). We now consider methods for parameterizing `GradNets`. In the case where a `GradNet` parameterizes the gradient of a *twice* continuously differentiable function, we use the following known result.

Lemma 1 (Antiderivatives and Symmetric Jacobians): A differentiable function $f: \mathbb{R}^d \to \mathbb{R}^d$ has a scalar-valued antiderivative if and only if its Jacobian is symmetric everywhere, i.e., $\forall \boldsymbol{x} \in \mathbb{R}^d\ \boldsymbol{J}_f(\boldsymbol{x}) = \boldsymbol{J}_f(\boldsymbol{x})^\top$.

The sufficient condition in Lemma 1 follows from the theorem of symmetric second derivatives credited to Clairaut, Schwarz, Young, and others [43]. The necessary condition follows from a slight modification of Prop. 1 in [29] to consider all functions with symmetric Jacobians. Thus, a neural network is a `GradNet` if its Jacobian with respect to its input is everywhere symmetric. We use Lemma 1 to provide a neural `GradNet` parameterization in Prop. 1 below.

Proposition 1: The neural network

$$\boldsymbol{W}^\top \sigma(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{a})+\boldsymbol{b} \tag{1}$$

is a `GradNet` if there exists $\psi \in C^1(\mathbb{R}^m)$ such that $\sigma = \nabla\psi$. In particular, such a ψ exists if activation $\sigma: \mathbb{R}^m \to \mathbb{R}^m$ is differentiable and its Jacobian $\boldsymbol{J}_\sigma$ is everywhere symmetric.

Proof: (1) is the gradient of $\psi(\boldsymbol{W}\boldsymbol{x}+\boldsymbol{a})+\boldsymbol{b}^\top\boldsymbol{x}$. If σ is additionally differentiable with $\boldsymbol{J}_\sigma$ everywhere symmetric then, by Lemma 1, there exists $\psi \in C^2(\mathbb{R}^d)$ such that $\sigma = \nabla\psi$. □

Prop. 1 states that (1) is a `GradNet` if the activation function σ has an antiderivative. It hence permits the use of *group* activations such as softmax. Furthermore, to guarantee existence of an antiderivative, it is sufficient, but not necessary, that σ is differentiable and its Jacobian is symmetric everywhere. For example, the elementwise activation $\text{ReLU}(\boldsymbol{x}) = \max(0, \boldsymbol{x})$ is continuous and nondifferentiable, but it is the gradient of $\sum_i \max(0, \frac{1}{2}x_i^2)$. Architectures of the form in (1), with a continuous elementwise activation function $\sigma(\boldsymbol{x}) = \begin{bmatrix}\sigma_1(x_1) & \ldots & \sigma_m(x_m)\end{bmatrix}^\top$, include "transform domain thresholding" methods [17], [31] where the activation is a continuous thresholding operator. Similar architectures appear in [35], which uses linear spline activations, and in [33], which parameterizes each σ_i as a linear combination of Gaussian radial basis functions. An alternative approach is to use neural networks to parameterize the elementwise activation functions.

Remark 1: (1) is a `GradNet` if $\sigma(\boldsymbol{x})$ is an elementwise activation where each σ_i is a neural network in $C^1(\mathbb{R})$.

We later prove in Sec. V-B that the network in Rem. 1 can approximate the gradient of any differentiable function that is the sum of ridge functions. In addition, if we restrict the domain of the `GradNet` to be compact, then it is sufficient for (1) to have continuous elementwise activation σ, since all continuous σ_i are integrable on compact subsets of $\mathbb{R}$. Lastly, by linearity of the gradient operator, the networks in (1), and other `GradNet` parameterizations, can be linearly combined to yield another `GradNet`.

Remark 2: Linear combinations of `GradNets` are also `GradNets`.

This section established methods for designing `GradNets`: neural networks guaranteed to correspond to gradients of continuously differentiable functions. It introduced a specific `GradNet` architecture, which we analyze in Sec. V, and serves

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

as a foundation for designing other GradNet architectures like *monotone* gradient networks in the following section.

IV. Monotone Gradient Networks (mGradNet)

Monotone gradient networks (mGradNets) are neural networks guaranteed to represent gradients of convex functions and are a subclass of gradient networks (GradNet) [14]. This section introduces mGradNets in a manner similar to the presentation of GradNets in Sec. III. We first define mGradNets and then discuss examples and their properties.

Definition 2 (mGradNet): A monotone gradient network (mGradNet) is a neural network f satisfying $f = \nabla F$ for some convex $F \in C^1_\times(\mathbb{R}^d)$.

In Sec. V-D, we extend Def. 2 to accommodate subgradients of non-differentiable convex functions. Next, we recall that a differentiable function is convex if and only if its gradient is monotone [44]. Therefore, mGradNets are guaranteed to be monotone functions.

Definition 3 (Monotonicity): $f : \mathbb{R}^d \to \mathbb{R}^d$ is monotone if:

$$\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d,\ (f(\boldsymbol{x}) - f(\boldsymbol{y}))^\top (\boldsymbol{x} - \boldsymbol{y}) \geq 0 \tag{2}$$

It is generally challenging to design an mGradNet satisfying (2) for all possible pairs of inputs. Instead, it is more tractable to rely on single-input characterizations of monotonicity. First, we provide an approach to parameterize gradients of convex functions using GradNets (from Sec. III) and known Lipschitz regularity techniques.

Remark 3: Let $L > 0$ and f be a GradNet that is L-Lipschitz, i.e., $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, $\|f(\boldsymbol{x}) - f(\boldsymbol{y})\| \leq L\|\boldsymbol{x} - \boldsymbol{y}\|$. Then $g(\boldsymbol{x}) = L\boldsymbol{x} - f(\boldsymbol{x})$ is an mGradNet.

Rem. 3 follows from Def. 1 and Remark 2.2 in [35], which implies monotonicity of g. It demonstrates how to construct an mGradNet using a GradNet with known Lipschitz constant. Rem. 3 relates to several Regularization by Denoising (RED) works that design learnable functions of the form $\boldsymbol{x} - f(\boldsymbol{x})$ where $f(\boldsymbol{x})$ is constrained to be nonexpansive [45], [46]. These methods differ from our approach as they do not guarantee that the Jacobian of $\boldsymbol{x} - f(\boldsymbol{x})$ is symmetric.

Next, we propose an alternative method that directly parameterizes monotone gradients and avoids Lipschitz assumptions. We use the fact that $F \in C^2(\mathbb{R}^d)$ is convex if and only if its Hessian $\boldsymbol{H}_f$ is everywhere positive semidefinite (PSD) [47]. Since $\boldsymbol{H}_F = \boldsymbol{J}^\top_{\nabla F}$, using a differentiable mGradNet to parameterize a monotone ∇F requires the Jacobian of the mGradNet, with respect to its input, to be PSD everywhere.

Proposition 2: The neural network in (1) is an mGradNet if there exists convex $\psi \in C^1_\times(\mathbb{R}^m)$ such that $\sigma = \nabla\psi$. In particular, such a ψ exists if activation $\sigma : \mathbb{R}^m \to \mathbb{R}^m$ is differentiable and its Jacobian $\boldsymbol{J}_\sigma$ is everywhere PSD.

Proof: (1) is the gradient of $\psi(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{a}) + \boldsymbol{b}^\top \boldsymbol{x}$, which is convex since ψ is convex. If σ is additionally differentiable with $\boldsymbol{J}_\sigma \succeq 0$, then by Lemma 1 and the fact that a twice differentiable function is convex if and only if its Hessian is PSD [47], there exists $\psi \in C^2(\mathbb{R}^d)$ such that $\sigma = \nabla\psi$. □

We highlight three key points concerning Prop. 2: 1) A neural network with Jacobian that is everywhere PSD is guaranteed to be an mGradNet; specifically, a GradNet in Prop. 1 with elementwise, monotone activation σ is an mGradNet. 2) Most popular elementwise activations, including softplus[1], tanh, and sigmoid, are nondecreasing and can be used to specify σ in Prop. 2. 3) Similar to the discussion on GradNets in Sec. III: if the activation σ in Prop. 2 has an antiderivative, it need not be differentiable; if we consider a compact domain, continuity of an elementwise activation σ is sufficient.

We leverage the generality of Prop. 2 to propose mGradNets with group activation σ and show that they universally approximate gradients of all differentiable convex functions. These mGradNets with group activations are provably more expressive than the methods in [30], [31], [32], [33], [34], [35]. Specifically, in Sec. V-A, we prove that there exist sequences of mGradNets with softmax activation functions that can universally approximate gradients of differentiable convex functions. The generality of Prop. 2 also permits neural parameterizations of elementwise activations.

Remark 4: (1) is an mGradNet if $\sigma(\boldsymbol{x})$ is an elementwise activation where each σ_i is an mGradNet in $C^1(\mathbb{R})$.

The mGradNets specified by Rem. 4 are more amenable for direct implementation than the spline activations proposed in [34], [35], while being provably as expressive. In particular, the spline activations in [34], [35] are defined using a fixed grid and hence require careful tuning of the grid domain and a large number of knots to achieve satisfactory performance. In contrast, we observe in Sec. VII that mGradNets in Rem. 4 provide more efficient parameterizations of the activations.

Similar to techniques described in [35], [48], mGradNets can be modified to correspond to gradients of μ-strongly convex functions.

Remark 5: Let $\mu > 0$ and f be an mGradNet. The function $g(\boldsymbol{x}) = f(\boldsymbol{x}) + \mu\boldsymbol{x}$ is an mGradNet corresponding to the gradient of a μ-strongly convex function.

As gradients of strongly convex functions are invertible, the method in Rem. 5 parameterizes *invertible* mGradNets, which, for example, can be employed in normalizing flows [38]. If an mGradNet corresponds to the gradient of a strongly convex function F, then its inverse corresponds to the gradient of the Fenchel dual F^* [49]. Lastly, we note that conical combinations of convex functions are also convex [47]. Analogous to Rem. 2, mGradNets can be combined to produce other mGradNets.

Remark 6: Conical combinations (linear combinations with nonnegative coefficients) of mGradNets yield mGradNets.

V. Universal Approximation Results

In this section, we analyze the expressivity of GradNet and mGradNet architectures respectively specified by Propositions 1 and 2. We show that these architectures can *universally approximate* various function classes. We first formally define universal approximation and a set of gradient functions, which we use throughout the section.

[1]The softplus function $\frac{1}{\beta}\log(1 + \exp(\beta x))$, with scaling factor β, is a smooth approximation of the commonly used ReLU activation.

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

Definition 4 (Universal Approximation): Let $\mathcal{S} \subset \mathbb{R}^d$ be compact, $\mathcal{F} : \mathcal{S} \to \mathbb{R}^d$ be a class of continuous functions, and $\mathcal{G} : \mathcal{S} \to \mathbb{R}^d$ be a class of approximators. $\mathcal{G}$ universally approximates $\mathcal{F}$ if for any $f \in \mathcal{F}$, there exists a sequence of $g_n \in \mathcal{G}$ that uniformly converges to f.

Def. 4 is equivalent to $\mathcal{G}$ being dense in $\mathcal{F}$ with respect to the supremum norm. Since $\mathcal{S}$ must be a subset of some scaled and shifted version of $[0,1]^d$, our universal approximation proofs, without loss of generality, consider the domain $[0,1]^d$.

Definition 5 (Set of Gradient Functions): Let $\mathcal{F}$ be a set of differentiable functions. Then the set $\nabla\mathcal{F} = \{\nabla F : F \in \mathcal{F}\}$.

In Sec. V-A, we prove that the `mGradNet` in (1) with scaled softmax activation and increasing hidden dimension can universally approximate the gradient of any convex function. We extend this result to show that the difference of two `mGradNets` can universally approximate the gradient of any L-smooth function.

In Sec. V-B, we analyze the impact of the activation function on the approximation capabilities of the networks. We show that `GradNets` and `mGradNets` with nonpolynomial activations can learn the gradient of any function expressible as the sum of ridge and convex ridge functions, respectively. In Sec. V-C, we introduce a simple augmentation that enables our networks to universally approximate the gradient of a transformed sum of (convex) ridges. Finally, in Sec. V-D, we extend `mGradNet` results to subgradients of convex functions.

A. (Monotone) Gradient Functions

We start by proving that `mGradNets` of the form (1) can universally approximate monotone gradients of convex functions. The proof uses the following lemma: differentiating convex approximators of a convex potential yields monotone approximators of the monotone gradient of the potential.

Lemma 2 (Convex Function and Monotone Gradient Approximation: [49] Theorem 25.7): Let $\mathcal{S} \subseteq \mathbb{R}^d$ be open, convex and let $F \in C^1_\times(\mathcal{S})$ be finite. Let $G_i \in C^1_\times(\mathcal{S})$ be a sequence of finite functions such that $\forall x \in \mathcal{S},\ \lim_{i\to\infty} G_i(x) = F(x)$. Then $\forall x \in \mathcal{S}, \lim_{i\to\infty} \nabla G_i(x) = \nabla F(x)$. The sequence ∇G_i converges uniformly to ∇F on every compact subset of $\mathcal{S}$.

Lemma 2 is specific to convex functions and generally does not extend to arbitrary functions [25]. We prove that `mGradNets` can universally approximate monotone gradients by first constructing a sequence of convex functions that approximates any continuous convex function, and then applying Lemma 2. To construct the sequence, we use the class of scaled LogSumExp (LSE) functions with scaling factor $t > 0$:

$$\mathrm{LSE}_t(\mathcal{U}) = \frac{1}{t}\log\left(\sum_{u\in\mathcal{U}} \exp(tu)\right)$$

We first present a result of independent interest in Lemma 3 below, where we derive an upper bound on the approximation error incurred when using the scaled LSE of affine functions to approximate continuous convex functions.

Lemma 3 (Bound on Convex Function Approximation by LSE): Let convex $F \in C^0_\times\left([0,1]^d\right)$ and $\epsilon > 0$. There exist scaling factor $t > 0$ and parameters $\{w_i, b_i\}_{i=1}^n$, where $n = (2^m - 1)^d$ and $m > 0$ depend on F, such that the scaled LSE of affine functions $G(x) = \mathrm{LSE}_t(\{w_1^\top x + b_1, \dots, w_n^\top x + b_n\})$ satisfies

$$\sup_{x\in[0,1]^d} |F(x) - G(x)| < (d+1)\epsilon + \frac{\log n}{t} \tag{3}$$

Proof: Let $\epsilon > 0$ and $F \in C^0_\times([0,1]^d)$. By the Heine-Cantor Theorem [50], F is uniformly continuous on $[0,1]^d$, meaning $\exists\delta > 0$ such that $\forall x, y \in [0,1]^d$, $\|x - y\| < \delta \implies |F(x) - F(y)| < \epsilon$. We select $m \in \mathbb{N}$ such that $2^{-m} < \delta$ and define $\mathcal{X}$ as the set of points with coordinates lying in $\{i2^{-m} : 1 \le i \le 2^m - 1\}$. This means $n = |\mathcal{X}| = (2^m - 1)^d$. For each $y \in \mathcal{X}$, let $L_y(x) = v^\top(x - y) + F(y)$ be a supporting hyperplane of F at y, where v is a subgradient of F at y, i.e., $v^\top(x - y) \le F(x) - F(y)$. With $x \in [0,1]^d$, $t > 0$, and $\mathcal{U} = \{L_y(x) : y \in \mathcal{X}\}$, the triangle inequality implies

$$\begin{aligned} &|F(x) - \mathrm{LSE}_t(\mathcal{U})| \\ &\quad\le \left|F(x) - \max_{y\in\mathcal{X}} L_y(x)\right| + \left|\max_{y\in\mathcal{X}} L_y(x) - \mathrm{LSE}_t(\mathcal{U})\right| \\ &\quad< (d+1)\epsilon + \frac{d}{t}\log(2^m - 1) = (d+1)\epsilon + \frac{\log n}{t} \end{aligned} \tag{4}$$

The first term of (4) uses Appendix C Prop. 2 of [38] and the second term uses $\mathrm{LSE}_t(\mathcal{U}) < \max(\mathcal{U}) + \frac{1}{t}\log(|\mathcal{U}|)$ [51]. □

In Lemma 3, choosing small ϵ and large scaling factor t corresponds to closely approximating a convex function F with a scaled LSE of affine functions. In the bound (3), n is the number of affine functions and is equivalent to the number of hidden neurons in the neural networks we propose and analyze in Thm. 1. The number of neurons n depends on ϵ and uniform continuity properties of F.

To illustrate the utility of the error bound, we consider the case where the convex function F is L-Lipschitz on $[0,1]^d$. This encompasses a wide range of functions, including mean squared error and polynomial functions. The variables δ and ϵ in the proof of Lemma 3 can then be related by taking $\delta < \epsilon/L$, leading to the following bound on the number of neurons n: $n > \left(\frac{L}{\epsilon} - 1\right)^d$. The bound on n indicates the rate at which the number of neurons must increase when either the input dimension increases, the Lipschitz constant of F increases, or we demand closer approximations (ϵ decreases). To achieve a desired approximation error, the scaling factor t should be increased logarithmically with respect to n. If we wish to approximate F with the sum of k identical LSE_t functions, then each LSE_t function must approximate $\frac{1}{k}F$, which is $\frac{L}{k}$-Lipschitz and $n > \left(\frac{L}{k\epsilon} - 1\right)^d$. Approximating F as a sum of identical functions, each being a scaled LSE of affine functions, thus allows each individual LSE_t to use fewer affine functions. This observation motivates our design of `mGradNet-Ms` in Sec. VI-A.

Before leveraging Lemma 2 to show universal approximation results for `mGradNets`, we first include the following result from [51], which states that scaled LSE functions universally approximate convex functions.

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

Lemma 4 (Scaled LSE as Universal Approximator of Convex Functions: [51] Theorem 2): Let $x \in \mathbb{R}^d$ and $\mathcal{G}$ be scaled LSE of affine functions $\mathrm{LSE}_t(\{w_1^\top x + b_1, \ldots, w_n^\top x + b_n\})$ with scaling factor $t > 0$. $\mathcal{G}$ universally approximates $C^0_\times([0,1]^d)$.

We defer the proof of Lemma 4 to [51].[2] Combining Lemma 2 with Lemma 4 enables us to show in Thm. 1 below that the `mGradNet` in Prop. 2 can universally approximate gradients of convex functions. The proof is constructive and demonstrates that the sequence of approximating functions corresponds to a sequence of `mGradNets` with softmax activation and increasing hidden dimension.

Theorem 1 (Universal Gradient Approximation for Convex Functions): Let $\mathcal{F} = C^1_\times([0-\delta, 1+\delta]^d)$ with $\delta > 0$. `mGradNets` in Prop. 2 with scaled softmax activation universally approximate $\nabla\mathcal{F}$ on $[0,1]^d$.

Proof: Let $F \in \mathcal{F}$ and let $G_n \in C^2_\times([0-\delta, 1+\delta]^d)$ be a sequence of the form $\mathrm{LSE}_t(\{w_1^\top x + b_1, \ldots, w_n^\top x + b_n\})$ that converges uniformly to F by Lemma 4. By the extreme value theorem [25], F and all G_n are finite. We consider the open convex subset $\mathcal{A} = (0-\delta, 1+\delta)^d \subset [0-\delta, 1+\delta]^d$ and observe that the G_n also converge uniformly to F on $\mathcal{A}$. By Lemma 2, $\nabla G_n \to \nabla F$ uniformly on compact subsets of $\mathcal{A}$, including $[0,1]^d$. Note that $\forall n$, ∇G_n is an `mGradNet` in Prop. 2 with scaled softmax activation and the ith row of $\boldsymbol{W}$ being $w_i^\top$: $\nabla G_n = \boldsymbol{W}^\top \mathrm{softmax}(t(\boldsymbol{W}x + \boldsymbol{b}))$ □

Using the fact that `mGradNets` can universally approximate all monotone gradient functions, we show that the difference of two `mGradNets` with softmax activations can universally approximate the gradient of any L-smooth function.

Theorem 2 (Universal Gradient Approximation for L-smooth Functions): Let $\delta > 0$ and $\mathcal{F}$ be L-smooth functions in $C^1([0-\delta, 1+\delta]^d)$. Let $g_1(x)$ and $g_2(x)$ be `mGradNets` in Prop. 2 with scaled softmax activations. `GradNets` $g(x) = g_1(x) - g_2(x)$ universally approximate $\nabla\mathcal{F}$ on $[0,1]^d$.

Proof: Let $F \in \mathcal{F}$, implying ∇F is L-Lipschitz continuous. We can write $\nabla F = Lx - (Lx - \nabla F(x))$, where Lx is clearly monotone and $Lx - \nabla F(x)$ is monotone by Rem. 3. By Thm. 1, there exist sequences of `mGradNets` uniformly converging to Lx and $Lx - \nabla F(x)$ on $[0,1]^d$. By Rem. 2, the difference of these sequences is a sequence of `GradNets` that converges uniformly to ∇F. □

B. Gradients of Sums of (Convex) Ridge Functions

Elementwise activations $\sigma(x) = \begin{bmatrix}\sigma_1(x_1) & \ldots & \sigma_n(x_n)\end{bmatrix}^\top$ are highly parallellizable in practice and are more commonly used than group activation functions like softmax. Prior works [30], [31], [32], [33], [34], [35], demonstrated the empirical success of elementwise activations used in `GradNets` of the form in (1). While we proved universal approximation for *any* (convex) gradient in Sec. V, this section demonstrates that using an elementwise activation function in (1) compromises representation power. Specifically, it leads to learning gradients of sums of (convex) ridge functions.

[2]A different approach to the proof in [51] easily follows from Lemma 3.

Definition 6 (Sum of (Convex) Ridge Functions): $F \in C^k(\mathbb{R}^d)$ is expressible as the finite sum of ridge functions if

$$F(x) = \sum_{i=1}^{N} \psi_i(a_i^\top x + b_i) \qquad (5)$$

where each profile function $\psi_i \in C^k(\mathbb{R})$. F is expressible as the sum of convex ridge functions if each $\psi_i \in C^k_\times(\mathbb{R})$.

Thm. 3 below shows that `GradNets` with scaled elementwise activations (e.g., scaled sigmoid, tanh, ReLU) can learn the gradient of any function that is the sum of ridges. It extends the result in [34], which considers approximating gradients of sums of convex ridges using learnable linear spline activations.

Theorem 3 (Universal Gradient Approximation for Sums of Ridges): Let $\mathcal{F}$ be functions in $C^1([0,1]^d)$ expressible as a finite sum of ridge functions. `GradNets` of the form in (1) with continuous, scaled, elementwise nonpolynomial activation σ universally approximate $\nabla\mathcal{F}$.

Proof: If $F \in \mathcal{F}$, then $\nabla_{\boldsymbol{x}} F(x) = \sum_{i=1}^N \boldsymbol{a}_i \psi_i'(\boldsymbol{a}_i^\top x + b_i)$. Let $\boldsymbol{A} = \begin{bmatrix}\boldsymbol{a}_1 & \ldots & \boldsymbol{a}_N\end{bmatrix}^\top$, $\Psi(\cdot) = \begin{bmatrix}\psi_1'(\cdot) & \ldots & \psi_N'(\cdot)\end{bmatrix}^\top$, and $\boldsymbol{b} = \begin{bmatrix}b_1 & \ldots & b_N\end{bmatrix}^\top$. Then $\nabla_{\boldsymbol{x}} F(x) = \boldsymbol{A}^\top \Psi(\boldsymbol{A}x + \boldsymbol{b})$, where each $\boldsymbol{a}_i^\top(\cdot) + b_i$ is a continuous affine transformation that maps $[0,1]^d$ to a compact subset $\mathcal{S}_i \subset \mathbb{R}$. We now consider the `GradNet` in Prop. 1 with $N \times d$ matrix $\boldsymbol{W} = \boldsymbol{A}$ and continuous elementwise activation $\sigma(x) = \begin{bmatrix}\sigma_1(x_1) & \ldots & \sigma_n(x_n)\end{bmatrix}^\top$. For each continuous ψ_i', let $\epsilon_i > 0$ and consider a corresponding σ_i satisfying $\forall x \in \mathcal{S}_i$, $|\psi_i'(x) - \sigma_i(x)| < \epsilon_i$. This `GradNet`'s approximation error is bounded as follows:

$$\left\| \sum_{i=1}^N \boldsymbol{a}_i(\psi_i'(\boldsymbol{a}_i^\top x + b_i) - \sigma_i(\boldsymbol{a}_i^\top x + b_i)) \right\| \leq \sum_{i=1}^N |\psi_i'(\boldsymbol{a}_i^\top x + b_i) - \sigma_i(\boldsymbol{a}_i^\top x + b_i)| \, \|\boldsymbol{a}_i\| \leq \sum_{i=1}^N \epsilon_i \|\boldsymbol{a}_i\|$$

Given any error threshold $\epsilon > 0$, there exist sufficiently small ϵ_i, and corresponding σ_i, such that the RHS of the final inequality above is bounded by ϵ.

Next, we parameterize each σ_i as a neural network of the form $\sigma_i(x) = \boldsymbol{u}_i^\top s(\boldsymbol{v}_i x + \boldsymbol{\beta}_i)$, $\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{\beta}_i \in \mathbb{R}^{m_i}$ with continuous, elementwise nonpolynomial activation s (e.g., tanh, sigmoid). By Thm. 1 in [52], for each ψ_i', there exists σ_i satisfying $\forall x \in \mathcal{S}_i$, $|\psi_i'(x) - \sigma_i(x)| < \epsilon_i$. Substituting the specified form of σ_i into the aforementioned `GradNet` yields $\sum_{i=1}^N \boldsymbol{a}_i \boldsymbol{u}_i^\top s(\boldsymbol{v}_i \cdot (\boldsymbol{a}_i^\top x + b_i) + \boldsymbol{\beta}_i)$. This can be rewritten as $\sum_{i=1}^N \boldsymbol{A}_i^\top \mathrm{diag}(\boldsymbol{u}_i) s(\mathrm{diag}(\boldsymbol{v}_i)(\boldsymbol{A}_i x + \boldsymbol{b}_i) + \boldsymbol{\beta}_i)$, where the $m_i \times d$ matrix $\boldsymbol{A}_i$ contains $\boldsymbol{a}_i^\top$ as its rows. Since s operates elementwise, we can again rewrite the `GradNet` as

$$\boldsymbol{A}^\top \boldsymbol{U} s(\boldsymbol{V}(\boldsymbol{A}x + \boldsymbol{b}) + \boldsymbol{\beta}) \qquad (6)$$

where the $(N \cdot \prod_{i=1}^N m_i) \times d$ matrix $\boldsymbol{A}$ vertically stacks the $\boldsymbol{A}_i$ matrices and the vector $\boldsymbol{\beta}$ stacks the $\boldsymbol{\beta}_i$ vectors. The diagonal matrices $\boldsymbol{U}, \boldsymbol{V}$ respectively have $\mathrm{diag}(\boldsymbol{u}_i), \mathrm{diag}(\boldsymbol{v}_i)$ along their diagonals. Therefore, (6) is a `GradNet` in (1) with intermediate diagonal matrices corresponding to scaled elementwise nonpolynomial activations. □

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

The proof of Thm. 3 uses the fact that neural networks with elementwise nonpolynomial activations universally approximate continuous functions on compact subsets of $\mathbb{R}$ [52]. In fact, the proof permits any elementwise activation that is a universal approximator.

Corollary 3.1: Let $\mathcal{F}$ be functions in $C^1([0,1]^d)$ expressible as a finite sum of ridge functions. Let σ be an elementwise activation where each $\sigma_i \in C^0(\mathbb{R})$ is a universal approximator of continuous functions on compact subsets of $\mathbb{R}$. `GradNets` in (1) with activation σ universally approximate $\nabla\mathcal{F}$.

Cor. 3.1 also applies to other architectures for approximating gradients. For example, variational networks proposed in [33] use Gaussian radial basis functions (RBFs) for elementwise activations, making them universal approximators of sums of ridge functions [53]. Similarly, the weakly convex ridge regularizers in [35], use elementwise learnable spline activations, which are also dense in the space of continuous functions.

Now, we shift focus to convex ridges and similarly prove that `mGradNets` with elementwise, nondecreasing activations can learn the gradient of any function expressible as the sum of *convex* ridges. The proof proceeds as follows: we first show that `mGradNets` in Prop. 2 with elementwise activations can universally approximate monotone functions on $\mathbb{R}$; similar to the proof of Thm. 3, we then employ this `mGradNet` to learn ψ_i', the monotone derivatives of the convex ridges. These results also motivate compositions of `mGradNets` in Sec. V-C and deeper networks in Sec. VI-B.

Lemma 5 (Universal Approximation for Scalar Monotone Functions): Let $\mathcal{F}$ be nondecreasing functions in $C^0([0,1])$. Let σ be an elementwise activation where each $\sigma_k \in C^0(\mathbb{R})$ is bounded, nondecreasing, and has finite asymptotic end behavior. `mGradNets` of the form (1) with activation σ universally approximate $\mathcal{F}$.

Proof: Let $f \in \mathcal{F}$ and, without loss of generality, let each $\sigma_k(x) : \mathbb{R} \to [0,1]$ with $\lim_{x\to-\infty} = 0$ and $\lim_{x\to\infty} = 1$. We uniformly partition $[0,1]$ into subintervals $\mathcal{I}_{n,k} = [\frac{k-1}{2^n}, \frac{k}{2^n}]$ for $k \in \{1, 2, \ldots, 2^n\}$ and consider the approximators

$$g_{n,t}(x) = f(0) + \sum_{i=1}^{2^n} \Delta_{n,i}\sigma_i(t(2^{n+1}x - 2i + 1))$$

where $t \geq 1$, each σ_i satisfies conditions given in the lemma, and $\Delta_{n,i} = f\left(\frac{i}{2^n}\right) - f\left(\frac{i-1}{2^n}\right) \geq 0$. Next, we bound

$$\max_{x\in[0,1]} |f(x) - g_{n,t}(x)| = \max_k \max_{x\in\mathcal{I}_{n,k}} |f(x) - g_{n,t}(x)| \quad (7)$$

To do so, we introduce $\widetilde{g}_{n,t}$, which has the same form as $g_{n,t}$ except each σ_i is replaced by $\widetilde{\sigma}$:

$$\widetilde{\sigma}(x) = \Big\{ 0 \text{ if } x \leq -1;\ x \text{ if } -1 < x < 1;\ 1 \text{ if } x \geq 1$$

In each of the 2^n subintervals, $\widetilde{g}_{n,t}$ interpolates f. For a specific k on the RHS of (7), triangle inequality implies

$$\max_{x\in\mathcal{I}_{n,k}} |f(x) - g_{n,t}(x)| \leq \max_{x\in\mathcal{I}_{n,k}} |f(x) - \tilde{g}_{n,t}(x)| + |\tilde{g}_{n,t}(x) - g_{n,t}(x)| \quad (8)$$

Let $\epsilon > 0$. For any $n \in \mathbb{N}, t \geq 1$, monotonicity of f implies that the first term on the RHS of (8) is bounded as follows:

$$\max_{x\in\mathcal{I}_{n,k}} |f(x) - \widetilde{g}_{n,t}(x)| \leq \Delta_{n,k}$$

Now, by the Heine-Cantor Theorem [50], f is uniformly continuous on $[0,1]$. Hence, let $n \geq n_0 \in \mathbb{N}$ such that $\forall k,\ \Delta_{n_0,k} = f(k/2^{n_0}) - f((k-1)/2^{n_0}) < \epsilon/3$. For each k, the second term on the RHS of (8) can be bounded by separately considering the absolute difference between $\tilde{g}(x)$ and $g(x)$ in three different intervals: $i < k$, $i = k$, and $i > k$.

$$\begin{aligned}&\max_{x\in\mathcal{I}_{n,k}} |\tilde{g}_{n,t}(x) - g_{n,t}(x)| \leq \\ &\max_{x\in\mathcal{I}_{n,k}} \Bigg\{ \sum_{i<k} \Delta_{n,i}[1 - \sigma_i(t(2^{n+1}x - 2i + 1))] \\ &\quad + \Delta_{n,k}|\widetilde{\sigma}(t(2^{n+1}x - 2k + 1)) - \sigma_k(t(2^{n+1}x - 2k + 1))| \\ &\quad + \sum_{i>k} \Delta_{n,i}\sigma_i(t(2^{n+1}x - 2i + 1)) \Bigg\} \quad (9)\end{aligned}$$

By boundedness of $\widetilde{\sigma}$ and σ_k, we have $\forall x,\ |\widetilde{\sigma}(x) - \sigma_k(x)| \leq 1$ and the middle term in (9) is upper-bounded by $\Delta_{n,k} < \epsilon/3$. Using the aforementioned value of n, there exists corresponding $t > 1$ such that the sum of the remaining terms of (9) is bounded by $\epsilon/3$. Therefore, $\max_{x\in[0,1]} |f(x) - g_{n,t}(x)| < \epsilon$ and $g_{n,t}$ universally approximates $\mathcal{F}$. We note that $g_{n,t}$ can be written in the form of (1) with nondecreasing activations:

$$g_{n,t} = f(0) + \mathbf{1}^\top \overline{\sigma}(\mathbf{1}x + \boldsymbol{b}) \quad (10)$$

where $\boldsymbol{b}_k = (-2k+1)/2^{n+1}$ and $\overline{\sigma}$ is an elementwise activation composed of $\overline{\sigma}_k(x) = \Delta_{n,k}\sigma_k(2^{n+1}tx)$. □

Lemma 5 states that a sufficiently wide `mGradNet` with scaled activations, like sigmoids, can closely approximate any continuous, monotone function on $\mathbb{R}$. It shows that a broader set of functions than the linear splines considered in [34] is dense in the space of continuous nondecreasing functions on $\mathbb{R}$. Next, we give universal approximation results for sums of *convex* ridges that generalize Prop. III.5 of [34]. These results use Lemma 5 and are similar to the proofs in Thm. 3 and Cor. 3.1 that pertain to sums of general ridge functions.

Theorem 4 (Universal Gradient Approximation for Sums of Convex Ridges): Let $\mathcal{F}$ be convex functions in $C^1_\times([0,1]^d)$ expressible as a finite sum of convex ridge functions. Let σ be an elementwise activation where each $\sigma_k \in C^0(\mathbb{R})$ is bounded, nondecreasing, and has finite asymptotic end behavior. `mGradNets` in Prop. 2 with scaled activation σ universally approximate $\nabla\mathcal{F}$.

Proof: F is of the form (5), where each ψ_i is convex. By Lemma 5, an `mGradNet` of the form (10) can approximate each nondecreasing ψ_i' on a compact subset of $\mathbb{R}$ within arbitrary error. The remainder of the proof follows from the proof of Thm. 3. □

Similar to Cor. 3.1, the proof of Thm. 4 permits the activation to be any elementwise function that is a universal approximator of nondecreasing functions.

Corollary 4.1: Let $\mathcal{F}$ be convex functions in $C^1_\times([0,1]^d)$ expressible as the finite sum of convex ridge functions. Let

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

σ be an elementwise activation where each $\sigma_i \in C^0(\mathbb{R})$ is a universal approximator of continuous, nondecreasing functions on compact subsets of $\mathbb{R}$. `mGradNets` in (1) with activation σ universally approximate $\nabla\mathcal{F}$.

C. Gradients of (Convex Monotone) Transformations of Sums of (Convex) Ridges

This section presents architectures that parameterize a broader subset of (monotone) gradient functions than those discussed in Sec. V-B. We specifically examine methods for composing vector-valued gradient networks with scalar-valued networks to learn gradients of transformations of sums of ridges. We then adapt these networks to learn gradients of convex, monotone transformations of sums of convex ridges.

Definition 7 ((Convex monotone) transformation of a sum of (convex) ridge functions): $F \in C^k(\mathbb{R}^d)$ is a transformation of a finite sum of ridge functions if it can be written as

$$F(x) = T\left(\sum_{i=1}^{N} \psi_i(a_i^\top x + b_i)\right)$$

where $\psi_i, T \in C^k(\mathbb{R})$. F is a convex, monotone transformation of a finite sum of convex ridge functions if $\psi_i, T \in C^k_\times(\mathbb{R})$ and T is additionally nondecreasing [47].

Def. 7 closely resembles Def. 6 with the addition of a transformation T applied to the sum of ridges. Before describing methods for learning gradients of such functions, we prove a useful lemma which shows that approximating ∇F with ∇G implies that G can approximate F.

Lemma 6: Let $\epsilon > 0$, and $f, \widehat{f} : [0,1]^d \to \mathbb{R}$ be differentiable. If $\sup_{x\in[0,1]^d} \|\nabla \widehat{f}(x) - \nabla f(x)\| \leq \epsilon$ and there exists $x_0 \in [0,1]^d$ such that $f(x_0)$ is known, then $\widehat{f}$ can approximate f such that $\sup_{x\in[0,1]^d} |\widehat{f}(x) - f(x)| \leq \epsilon\sqrt{d}$.

Proof: Let $p(x) = \widehat{f}(x) - f(x)$, where $\widehat{f}(x)$ is the antiderivative of the known $\nabla\widehat{f}$ with a constant of integration determined by x_0. Note that $\nabla p(x) = \nabla\widehat{f}(x) - \nabla f(x)$. By the multivariate mean value theorem [25], $\forall x, y \in [0,1]^d$, $\exists c \in (0,1)$ such that $z = (1-c)x + cy$ and

$$p(x) - p(y) = \nabla p(z)^\top (x - y)$$
$$\widehat{f}(x) - f(x) - \widehat{f}(y) + f(y) = (\nabla\widehat{f}(z) - \nabla f(z))^\top (x - y)$$

Let $y = x_0$ so that $-\widehat{f}(y) + f(y) = 0$. By the Cauchy-Schwarz inequality and an upper bound on $\|x - x_0\|$,

$$|\widehat{f}(x) - f(x)| \leq \|\nabla\widehat{f}(z) - \nabla f(z)\| \|x - x_0\| \leq \epsilon\sqrt{d}$$

We note that if $\nabla\widehat{f}_n$ is a sequence of functions that converges to ∇f, with a corresponding sequence of $\epsilon_n > 0$ converging to 0, then the sequence of antiderivatives $\widehat{f}_n$ converges to f regardless of the dimension d. □

Using the above lemma, we describe a method for learning gradients of functions specified in Def. 7.

Theorem 5 (Universal Gradient Approximation for Transformed Sums of Ridges): Let $\mathcal{F}$ be functions in $C^1([0,1]^d)$ expressible as transformations of finite sums of ridge functions. Let $G \in C^1([0,1]^d)$, $g(x) = \nabla_x G(x)$, and $\gamma \in C^0(\mathbb{R})$. Let h be of the form

$$h(x) = \gamma(G(x) + \beta) \cdot g(x) \tag{11}$$

where $\beta \in \mathbb{R}$ is a learnable bias. If γ universally approximates continuous functions on any compact subset of $\mathbb{R}$ and g universally approximates gradients of sums of ridge functions on $[0,1]^d$, then functions of the form h are `GradNets` that universally approximate $\nabla\mathcal{F}$.

Proof: If $F \in \mathcal{F}$, then ∇F is of the same form as h:

$$\nabla F = T'\left(\sum_{i=1}^{N} \psi_i(a_i^\top x + b_i)\right) \cdot \left(\sum_{i=1}^{N} a_i \psi_i'(a_i^\top x + b_i)\right) \tag{12}$$

By definition, γ can approximate T' and $g(x)$ can approximate $\sum_{i=1}^{N} a_i \psi_i'(a_i^\top x + b_i)$ within arbitrary error. By Lemma 6, we get that $G(x) + \beta$ can simultaneously approximate $\sum_{i=1}^{N} \psi_i(a_i^\top x + b_i)$. Thus, each function in (11) can learn the corresponding function in (12) within arbitrary error. Furthermore, (12) is a `GradNet` as it corresponds to $\nabla\Gamma(G(x) + \beta)$, where, by continuity of γ on $\mathbb{R}$, Γ is the antiderivative of γ on the compact codomain of $G(x) + \beta$. □

Thm. 5 demonstrates that gradients of functions specified by Def. 7 can be learned by parameterizing γ as a scalar-valued neural network (since neural networks are universal function approximators [54]) and taking $g(x)$ to be a network of the form given in Thm. 3, which can approximate gradients of sums of ridges to arbitrary error. Furthermore, Cor. 3.1 allows us to parameterize g with a single hidden layer neural network whose antiderivative is simple to compute, hence we can easily obtain the form for G, the antiderivative of g, in Thm. 5.

We now extend Thm. 5 to specifically use `mGradNets` to learn gradients of *monotone convex* transformations of sums of *convex* ridges. The parameterization requires a function γ that universally approximates monotone nonnegative functions. The following construction of γ leverages the universality of the `mGradNet` on $\mathbb{R}$, as proved in Lemma 5.

Lemma 7: Let $\mathcal{F}$ be the set of nondecreasing functions in $C^0([0,1])$. Let τ universally approximate nondecreasing functions in $C^0([0,1])$ and ρ be uniformly continuous and strictly increasing. Then functions of the form $\rho(\tau(x))$ universally approximate $\mathcal{F}$.

Proof: Let $f \in \mathcal{F}$ and $\epsilon > 0$. Since ρ is strictly increasing, it is invertible. Since $\rho^{-1}(f(x))$ is continuous and nondecreasing, there exists τ that approximates $\rho^{-1}(f(x))$ with arbitrary error $\delta > 0$. By uniform continuity of ρ there exists $\delta > 0$ such that $\forall x$, $|\tau(x) - \rho^{-1}(f(x))| < \delta$ implies $|\rho(\tau(x)) - f(x)| < \epsilon$. □

Lemma 7 demonstrates how composing a universal approximator of differentiable monotone functions with another function can yield a universal approximator of differentiable monotone functions that map to a subset of $\mathbb{R}$. Lipschitz functions are uniformly continuous, thus taking ρ to be softplus and τ to be an `mGradNet` in Lemma 5 yields a universal approximator of differentiable, nonnegative, monotone functions γ. We extend Thm. 5 with this construction for γ.

Theorem 6 (Universal Gradient Approximation for Transformed Sums of Convex Ridges): Let $\mathcal{F}$ be functions in

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

$C^1_\times([0,1]^d)$ expressible as convex monotone transformations of finite sums of convex ridge functions. Let h be as in (11), where $G \in C^1_\times([0,1]^d)$, $\beta \in \mathbb{R}$ is a learnable bias, and $\gamma \in C^0(\mathbb{R})$ is nondecreasing and nonnegative. Let γ universally approximate nondecreasing and nonnegative functions in $C^0(\mathbb{R})$ on compact subset of $\mathbb{R}$. Let $g(x) = \nabla_x G(x)$ universally approximate gradients of sums of convex ridge functions on $[0,1]^d$. Functions of the form h are mGradNets that universally approximate $\nabla\mathcal{F}$.

Proof: Let $F \in \mathcal{F}$. Then $\nabla F(x)$ is as in (12), where convexity of ψ_i and convex monotonicity of T respectively imply that the ψ_i' are monotone and T' is monotone nonnegative. Lemma 7 provides a construction for continuous γ that can approximate T' within arbitrary error. For instance, τ can be as described in Lemma 5 and ρ can be any uniformly continuous, nondecreasing function such as softplus or ReLU. The remainder of the proof is nearly identical to that of Thm. 5 with G, g appropriately corresponding to convex ridge functions. □

Thm. 6 demonstrates that h can be parameterized using an mGradNet to yield the gradient of any function expressible as the convex monotone transformation of a sum of convex ridges. As in Thm. 5, knowledge of the antiderivative of the mGradNet $g(x)$ is required. However, as shown in Cor. 4.1, there exist mGradNets with known antiderivatives that can learn the gradient of any sum of convex ridges.

D. Learning Subgradients of Convex Functions

The previous subsections discussed parameterizing and learning gradients of differentiable convex functions. In this section, we relax the differentiability assumption and show that mGradNets can effectively characterize subgradients of *subdifferentiable* convex functions.

Definition 8 (Subgradient, Subdifferential): A subgradient of $F \in C^0_\times(\mathbb{R}^d)$ at a point x is any v that satisfies

$$v^\top(y-x) \le F(y) - F(x),\ \forall y \in \mathbb{R}^d$$

The subdifferential of F at x, denoted by $\partial F(x)$, is the set of all subgradients of F at x.

If F is convex and differentiable at a point x, then the subgradient at x is uniquely the gradient of F at x. We also note that any convex function is subdifferentiable on the interior of its domain [49]. To learn subgradients of convex functions, we use the following lemma.

Lemma 8 ([49] Theorem 24.5): Let $\mathcal{S} \subseteq \mathbb{R}^d$ be open, convex and let $F \in C^0_\times(\mathcal{S})$ be finite. Let $G_i \in C^0_\times(\mathcal{S})$ be a sequence of finite functions converging pointwise to F on $\mathcal{S}$. For any $\epsilon > 0$, there exists index i_0 such that $\forall i \ge i_0$, $\partial G_i(x) \subset \partial F(x) + \epsilon\mathcal{B}$ where $\mathcal{B}$ is the Euclidean unit ball of $\mathbb{R}^d$.

The lemma states that if a sequence of convex approximators converges pointwise to a target convex function, then the subdifferentials of the approximators become increasingly similar to those of the target function. If the approximators are additionally differentiable, the subdifferential $\partial G_i(x)$ is the gradient $\nabla G_i(x)$ and can be made arbitrarily close to an element in $\partial F(x)$. Therefore, the existing results in Sec. V based on Lemma 2 can be readily adapted to handle subdifferentiable convex functions.

VI. GradNet Architecture Variants

In this section, we propose specific neural network architectures for parameterizing GradNets and mGradNets. The architectures we propose exhibit universal approximation properties while empirically being more amenable for optimization (as seen in Sec. VII). In the following subsections, we initially describe conditions under which our proposed architectures are GradNets and then discuss stricter conditions under which the networks become mGradNets.

A. Modular (Monotone) Gradient Networks (GradNet-M, mGradNet-M)

Here we describe modular gradient networks (GradNet-M) and their monotone counterparts (mGradNet-M), which are motivated by the discussion of Lemma 3 in Sec.V. These networks achieve wide, modular architectures by respectively using GradNets and mGradNets, with different weight matrices, as building blocks. GradNet-Ms (mGradNet-Ms) can universally approximate a broader range of functions than just those expressible as the sum of (convex) ridges, and thus are more expressive than existing methods [30], [31], [32], [33], [34], [35]. GradNet-Ms universally approximate gradients of L-smooth functions and transformations of sums of ridges, as shown in Thm. 2 and Thm. 5 respectively. Similarly, mGradNets universally approximate gradients of convex functions and gradients of transformations of sums of ridges, as demonstrated in Thm. 1 and Thm. 6.

The design of GradNet-Ms leverages the facts that a linear combinations of GradNets and a composition of a GradNet with a scalar-valued, differentiable function both yield GradNets, as described respectively in Rem. 2 and Thm. 5. The following equations define the GradNet-M and generalize the architecture we previously proposed in [14]:

$$z_m = W_m x + b_m \tag{13}$$

$$\text{GradNet-M}(x) = a + \sum_{m=1}^{M} \rho_m(\phi_m(z_m)) W_m^\top \sigma_m(z_m) \tag{14}$$

where a denotes a learnable bias and M denotes the number of modules – a hyperparameter tunable based on the application. σ_m, W_m and b_m respectively denote the activation function, weight matrix, and bias corresponding to the m^{th} module. The ϕ_m are vector-to-scalar functions and ρ_m are scalar-to-scalar functions. The block diagram corresponding to a single module output is shown in Fig. 2. In the following theorem, we provide conditions for ϕ_m, ρ_m, and σ_m under which the GradNet-M is a GradNet.

Theorem 7 (GradNet-M conditions): If ρ_m, ϕ_m, σ_m are all differentiable and $\sigma_m = \nabla\phi_m$, the GradNet-M in (13)-(14) is a GradNet.

Proof: Consider the m^{th} module $\rho_m(\phi_m(z_m)) W_m^\top \sigma_m(z_m)$. The Jacobian of the m^{th} module with respect to x is

$$\begin{aligned} J_m(x) = {} & \rho_m'(\phi_m(z_m))(W_m^\top \sigma_m(z_m))(W_m^\top \sigma_m(z_m))^\top \\ & + \rho_m(\phi_m(z_m)) W_m^\top J_{\sigma_m}(z_m) W_m \end{aligned} \tag{15}$$

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

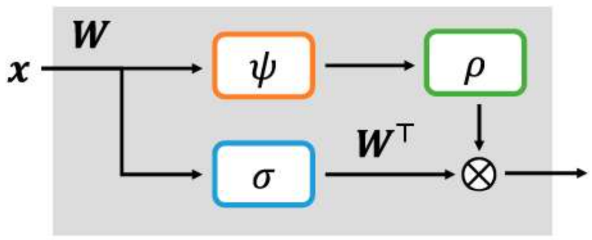


Fig. 2. Single module of the modular gradient network (GradNet-M) defined in (13)-(14).

The first term on the RHS is a Gram matrix scaled by $\rho_m'(\phi_m(z_m))$, and hence symmetric. The second term on the RHS is symmetric by Lemma 1 since $J_{\sigma_m}^\top = H_{\phi_m}$. Thus, each module is a GradNet and the result holds by Rem. 2. □

Next, we introduce the monotone GradNet-M:

Corollary 7.1 (mGradNet-M conditions): The GradNet-M defined by (13)-(14) is an mGradNet, and referred to as mGradNet-M if, for each module,

a) $\phi_m : \mathbb{R}^d \to I_m \subseteq \mathbb{R}$ is convex, twice differentiable
b) $\rho_m : I_m \to \mathbb{R}_{\geq 0}$ is differentiable and monotone
c) $\sigma_m = \nabla \phi_m$

Proof: The Jacobian of the m^{th} module is given by (15). Convexity of ϕ_m implies J_{σ_m} is everywhere PSD. Since ρ_m is nonnegative-valued, the second term on the RHS of (15) is PSD. Monotonicity of ρ_m implies that ρ_m' is everywhere nonnegative, making the first term of (15) also PSD. Hence, each module is an mGradNet and the result holds by Rem. 6. □

There are several suitable choices for ρ_m, ϕ_m, and σ_m in Cor. 7.1. For example, $\phi_m(x) = \mathrm{LSE}(x)$, $\sigma_m(x) = \mathrm{softmax}(x)$ and ρ_m can be any differentiable, monotone, and nonnegative function on $\mathbb{R}$ such as softplus. As another example, one can take σ_m to be the elementwise sigmoid activation function with $\phi_m(x) = \sum_i \mathrm{softplus}(x_i)$ and $\rho(x) = \alpha x + \beta$, where $\alpha > 0$.

B. Cascaded (Monotone) Gradient Networks (GradNet-C, mGradNet-C)

In this section we generalize the GradNet and mGradNet, described by (1), to achieve *deeper* networks. The architecture is inspired by Thm. 3, which states that the GradNet in (1) with elementwise neural network activations can universally approximate the gradient of any function expressible as the sum of ridges. The networks we propose in this section, namely GradNet-C and mGradNet-C, are consequently as expressive as the methods in [30], [31], [32], [33], [34], [35] while featuring a novel and more efficient parameterization of the elementwise activation functions. We propose a cascaded gradient network (GradNet-C), illustrated in Fig. 3 and defined by the layerwise equations:

$$z_0 = \beta_0 \odot Wx + b_0 \tag{16}$$
$$z_\ell = \beta_\ell \odot Wx + \alpha_\ell \odot \sigma_\ell(z_{\ell-1}) + b_\ell$$
$$\texttt{GradNet-C}(x) = W^\top \left[\alpha_L \odot \sigma_L\left(z_{L-1}\right)\right] + b_L \tag{17}$$

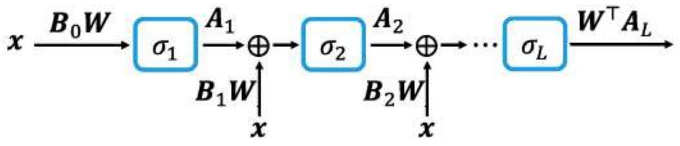


Fig. 3. Cascaded gradient network (GradNet-C), defined in (16)-(17), with $A_\ell = \mathrm{diag}(\alpha_\ell)$, $B_\ell = \mathrm{diag}(\beta_\ell)$.

where z_ℓ, b_ℓ, and σ_ℓ respectively denote the output, bias, and activation function at layer ℓ of the network and $\odot$ is the Hadamard (entrywise) vector product. The weight matrix W is shared across all layers whereas the intermediate scaling weight vectors α_ℓ, β_ℓ are unique to each layer.

Theorem 8 (GradNet-C conditions): The GradNet-C in (16)-(17) is a GradNet if the σ_ℓ are differentiable elementwise activations.

Proof: Let $A_\ell = \mathrm{diag}(\alpha_\ell), B_\ell = \mathrm{diag}(\beta_\ell)$. The Jacobian of GradNet-C(x) with respect to x is:

$$J_{\texttt{GradNet-C}}(x) = W^\top D W \tag{18}$$
$$D = \sum_{\ell=1}^{L} \left(\prod_{i=0}^{L-\ell} A_{L-i} J_{\sigma_{L-i}}\left(z_{L-i-1}\right) \right) B_{\ell-1} \tag{19}$$

The activation Jacobians J_{σ_ℓ} are diagonal matrices since each σ_ℓ operates elementwise on the input. Therefore D is a diagonal matrix and $J_{\texttt{GradNet-C}}$ is symmetric. □

The activation σ_ℓ in Thm. 8 can be a fixed function applied elementwise, e.g., tanh, softplus, and sigmoid, or a *learnable* function that operates on individual elements of the input vector. Thus, the proposed approach also permits elementwise activations parameterized by scalar-valued neural networks previously highlighted in Cor. 3.1. The proposed cascaded networks also remain GradNets if all β_ℓ are 0. However, as shown in [55], [56], skip connections accelerate training as they address the vanishing gradient problem in deep networks and smoothen the loss landscape. Next, we introduce the monotone variant of the GradNet-C.

Corollary 8.1 (mGradNet-C conditions): The GradNet-C in (16)-(17) is an mGradNet, and referred to as mGradNet-C, if, at each layer, the scaling weights α_ℓ, β_ℓ are nonnegative and the σ_ℓ are differentiable, monotonically-increasing elementwise activation functions.

Proof: Consider $J_{\texttt{GradNet-C}}$ from (18)-(19). By the assumptions in the theorem, for all ℓ, A_ℓ, B_ℓ are nonnegative diagonal matrices and the activation Jacobians J_{σ_ℓ} are also nonnegative diagonal matrices. Thus, D is everywhere PSD and it follows that $J_{\texttt{GradNet-C}}$ is everywhere PSD. □

The condition in Cor. 8.1 states that the activations σ_ℓ of an mGradNet-C should be monotonically increasing. Most popular elementwise activation functions, e.g., tanh, softplus, and sigmoid, satisfy this requirement. Cor. 8.1 also permits the use of learned, elementwise mGradNet activations as discussed in Thm. 4. When considering GradNet-C and mGradNet-C on compact domains in practice, the activations σ need only be continuous rather than differentiable. The requirement that

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

the intermediate elementwise scaling weights of mGradNet-C at each layer are nonnegative vectors is easy to parameterize in practice. Moreover, we empirically observe in Sec. VII that imposing nonnegativity constraints on the intermediate weight vectors does not impair optimization or final performance.

VII. Experiments

A. Gradient Field

We demonstrate the proficiency of our networks in learning gradients of scalar functions over the unit cube $[0,1]^d$. To visualize the error incurred by each network, we consider a low-dimensional setting with $d=2$ in Sec. VII-A1. We find that our methods result in error plots with fewer irregularities than baseline methods. In Sec. VII-A2, we consider high-dimensional settings with $d \in \{32, 256, 1024\}$ and demonstrate that our methods achieve an improvement of up to 10 dB when learning the gradient of a convex potential. For nonconvex potentials, the improvement reaches as high as 15 dB.

1) 2D Gradient Field: We start with $d=2$, where the error incurred by each method can be visualized on the unit square. For the convex test function, we consider learning the gradient of the following benchmark potential function from [29], which is convex on the unit square $x_1, x_2 \in [0,1]$:

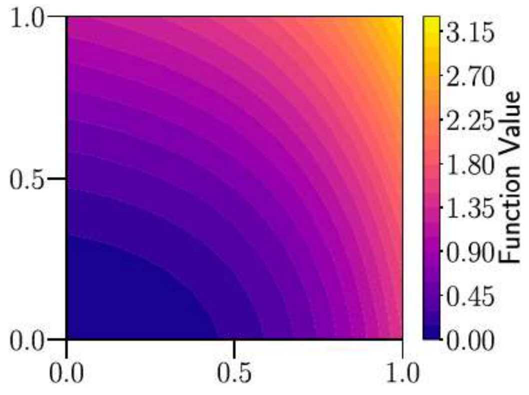
$$F(x_1,x_2) = x_1^4 + \frac{x_1^2}{2} + \frac{x_1 x_2}{2} + \frac{3x_2^2}{2} - \frac{x_2^3}{3} \qquad (20)$$

For the nonconvex test function, we consider learning the gradient of the following potential over the unit square:

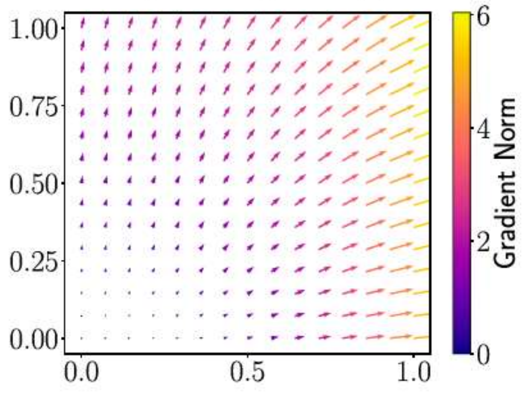
$$G(x_1,x_2) = \frac{1}{4}\sin(2\pi x_1)\cos(\pi x_2) + \frac{x_1 x_2}{2} - \frac{x_2^2}{2} \qquad (21)$$

Contour plots of F and G, along with their corresponding gradients, are shown in Fig. 4. For the convex test function $F(x)$ in (20), we compare the proposed mGradNet-C and mGradNet-M, respectively defined in (16)-(17) and (13)-(14), with the Input Convex Neural Network (ICNN) [36], Input Convex Gradient Network (ICGN) [29], and Convex Ridge Regularizer (CRR) [34]. Similarly for the nonconvex test function $G(x)$ in (21), we compare the proposed modular gradient network (GradNet-M) and cascaded gradient network (GradNet-C) to a multilayer perceptron (MLP) and to the ridge regularizer (RR) network proposed in [35]. The input to all models is a point $x \in [0,1]^2$, and all networks, excluding the ICNN and MLP, are trained to directly output $\nabla F(x)$ or $\nabla G(x)$. The input to the ICNN and MLP is also x and the gradient of the networks, with respect to x, are trained to approximate $\nabla F(x)$ or $\nabla G(x)$, respectively. The specific architecture configurations for each model are provided in Appendix A. All models are constrained to have roughly 100 parameters. We sample 100,000 training points uniformly at random on the unit square and separately sample 10,000 validation points. All models are trained using mean squared error (MSE) loss and their performance is evaluated on a uniform grid of points in the unit square. The ℓ_2 norm of the gradient prediction error is shown for each model in Fig. 5.

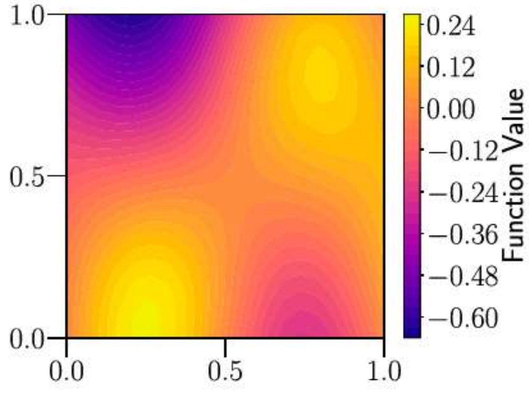
We observe from Fig. 5 that, for learning $\nabla F(x)$, the proposed mGradNet-M and mGradNet-C exhibit significantly lower error rates than that of the ICNN, ICGN, and CRR. Fig. 5



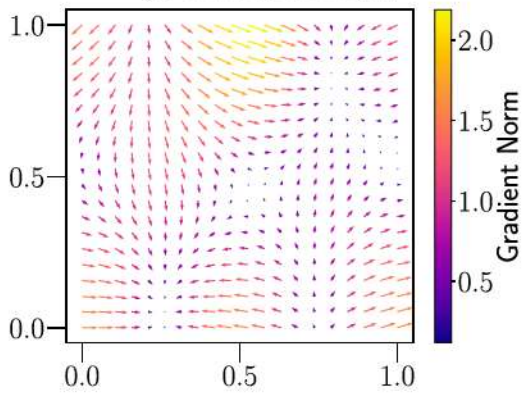
(a) Contour plot of $F(\boldsymbol{x})$



(b) Quiver plot of $\nabla F(\boldsymbol{x})$



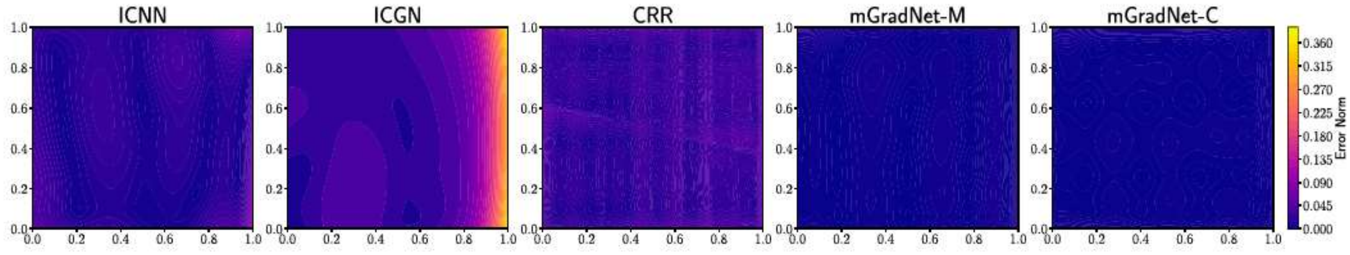
(c) Contour plot of $G(\boldsymbol{x})$



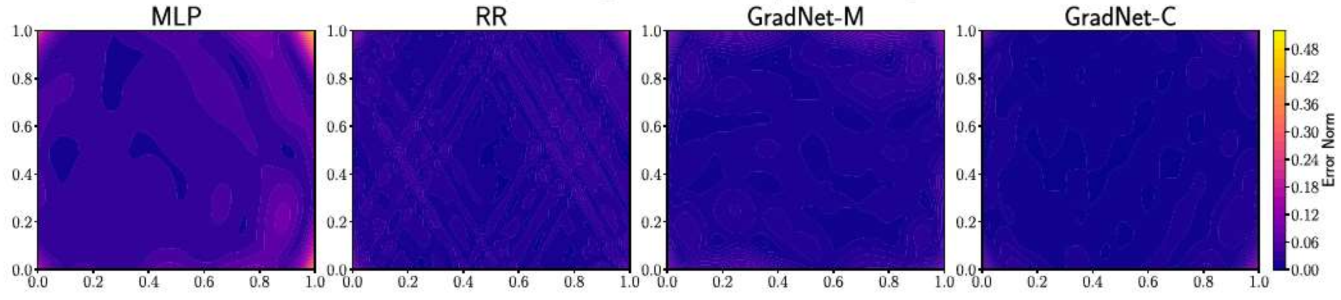
(d) Quiver plot of $\nabla G(\boldsymbol{x})$

Fig. 4. Test functions F in (20) and G in (21), along with their gradients, for $d=2$ over the unit square.

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.



(a) ℓ_2 norm between the predicted gradient and $\nabla F(\boldsymbol{x})$, the true gradient of F in (20)



(b) ℓ_2 norm between the predicted gradient and $\nabla G(\boldsymbol{x})$, the true gradient in (21)

Fig. 5. Gradient field learning results for $d = 2$.

also shows that the mGradNet-M and mGradNet-C are able to effectively learn the gradient at all areas of the unit square, whereas the ICNN, ICGN, and CRR tend to underperform around the edges of square. Similar trends are observed in the nonconvex case where the GradNet-C, GradNet-M, and RR outperform the MLP.

2) d-Dimensional Gradient Field, $d \geq 32$: We proceed to consider gradient field learning in higher dimensions. Our main findings are as follows. We observe that our mGradNet-C and GradNet-C architectures consistently outperform the CRR [34] and RR [35], both of which also use elementwise activations. Our mGradNet-M consistently outperforms all other methods for learning the gradient of a convex potential across all settings considered. Meanwhile, for the nonconvex potential, the best-performing method is consistently either our proposed GradNet-C or the GradNet-M.

For the convex test function we use the following positive definite matrices defined in [57]:

$$\boldsymbol{S}_{ij} = \frac{2+\sin(4\pi\alpha_{ij})}{(1+|i-j|\ln d)} \qquad \boldsymbol{P}_{ij} = \frac{1+2\alpha_{ij}}{(1+|i-j|\ln d)}$$
$$\boldsymbol{Q}_{ij} = \frac{3-2\alpha_{ij}}{(1+|i-j|\ln d)} \qquad \alpha_{ij} = \frac{i+j-2}{2d-2} \in [0,1]\,\forall i,j$$

where $1 \leq i, j \leq d$ and $\boldsymbol{S}_{ij}$ denotes entry ij of the matrix $\boldsymbol{S}$. We consider learning the gradient (where it exists) of the convex, piecewise quadratic function:

$$z_i = x_i - 0.5$$
$$\widetilde{F}(x) = \max\{\boldsymbol{z}^\top \boldsymbol{S}\boldsymbol{z}, \boldsymbol{z}^\top \boldsymbol{P}\boldsymbol{z}, \boldsymbol{z}^\top \boldsymbol{Q}\boldsymbol{z}\} \tag{22}$$

over the unit hypercube $x \in [0,1]^d$ for $d \in \{32, 256, 1024\}$. We also consider a nonconvex setting in which we train the appropriate models to learn the score function of a Gaussian mixture model (GMM), $\nabla_{\boldsymbol{x}} \ln \widetilde{G}(x)$, where $\ln \widetilde{G}(x)$ is the log probability density function:

$$\ln \widetilde{G}(x) = \ln\left(\sum_{i=1}^{N} \alpha_i N(x; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\right) \tag{23}$$

In (23) above, $N(x; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ denotes the probability density function of a multivariate normal distribution with mean $\boldsymbol{\mu}_i \in \mathbb{R}^d$ and $d \times d$ covariance matrix $\boldsymbol{\Sigma}_i$. Since most of the probability mass of a d-dimensional standard normal distribution lies in a thin annulus that is centered at the origin and has inner and outer radii close to $\sqrt{d}$ [58], we ensure interesting score functions in the unit hypercube by drawing each component of $\boldsymbol{\mu}_i$ uniformly at random from $[0.3, 0.7]$ and setting $\boldsymbol{\Sigma} = 2\sqrt{d}\boldsymbol{I}$. We select equal weights for the Gaussians. We train all models in these experiments for 10,000 iterations, where at each iteration we randomly sample 100 points from $[0,1]^d$ to update the model parameters. The specific model configurations and additional training details are provided in Appendix B.

The results for learning the gradients of (22) and (23) are shown in Tables I and II respectively. We observe in Table I that the mGradNet-M outperforms the baseline methods by up to 10 dB in the high-dimensional setting of $d = 1024$. Furthermore, although they parameterize the same space of functions, we observe that the mGradNet-C outperforms the CRR from [34] in the settings $d = 256$ and $d = 1024$ by margins of roughly 4 dB and 2 dB respectively.

Similarly for the score learning task, we observe in Table II that the GradNet-M and GradNet-C significantly outperform all other methods. The GradNet-C outperforms the RR by nearly 8 dB for $d = 32$ and by over 4 dB for $d = 256$. The GradNet-M outperforms the RR by a margin of over 5 dB for $d = 1024$, thus demonstrating its effectiveness in high dimensions. The MLP baseline underperformed all other

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

TABLE I
LEARNING THE GRADIENT OF CONVEX POTENTIAL (22) (15 TRIALS). *OUR METHODS

Model	$d=32$ MSE (dB)	$d=256$ MSE (dB)	$d=1024$ MSE (dB)
ICNN [36]	-12.76 ± 0.03	-9.83 ± 0.01	-0.36 ± 0.01
CRR [34]	-13.86 ± 0.03	-6.24 ± 0.04	-6.53 ± 0.01
mGradNet-C*	-12.80 ± 0.03	-10.64 ± 0.01	-8.75 ± 0.02
mGradNet-M*	**-15.34** ± 0.03	**-11.30** ± 0.02	**-10.42** ± 0.01

TABLE II
LEARNING THE NONCONVEX SCORE FUNCTION CORRESPONDING TO (23) (15 TRIALS). *OUR METHODS

Model	$d=32$ MSE (dB)	$d=256$ MSE (dB)	$d=1024$ MSE (dB)
MLP	-33.20 ± 0.04	-18.21 ± 0.14	-2.78 ± 0.01
RR [35]	-32.50 ± 0.30	-20.30 ± 2.76	-11.26 ± 0.89
GradNet-C*	**-44.43** ± 0.07	**-24.90** ± 0.07	-12.84 ± 0.04
GradNet-M*	-37.66 ± 0.02	-21.72 ± 0.14	**-17.02** ± 0.03

considered methods, especially for $d=1024$, where its performance lagged by nearly 9 dB behind the second worst model, namely the RR. In both the convex and nonconvex experiments, the modular architectures, `mGradNet-M` and `GradNet-M`, perform better than the other methods in the high-dimensional setting $d=1024$.

B. Hamiltonian Gradients for Two-Body Dynamics

In this section, we train `GradNets` to learn gradients of the Hamiltonian function for the two-body problem and subsequently use the learned models to predict the trajectories of the two objects. We observe that our `GradNet-M` outperforms the baseline Hamiltonian NN [59] by over 3 dB and the RR [35] by over 11 dB. As in the gradient field experiments, our `GradNet-C` architecture again outperforms the RR [35] – this time by over 3 dB. The performance gaps that arise in this experiment directly validate the theoretical differences in approximation capabilities between the baseline methods and `GradNets` (see Sec. V).

Hamiltonian mechanics has many fundamental applications ranging from classical mechanics and thermodynamics to quantum physics. In fact, predicting the dynamics of objects in a system can be achieved by learning the gradient of the corresponding scalar-valued Hamiltonian function. We define q and p respectively as the vectorized position and momentum information of all objects in the system. The Hamiltonian $\mathcal{H}(q,p)$ characterizes the total energy of the system and offers a complete description of the system. The time derivatives of the position q and momentum p (which are respectively the velocities and forces) of all objects can be obtained by differentiating the Hamiltonian of the system:

$$\frac{dq}{dt}=\frac{\partial \mathcal{H}}{\partial p} \qquad -\frac{dp}{dt}=\frac{\partial \mathcal{H}}{\partial q} \tag{24}$$

TABLE III
OBJECT COORDINATE MSE AND TOTAL SYSTEM ENERGY MSE OF 2-BODY PROBLEM TRAJECTORIES UNROLLED USING (24) AND HAMILTONIAN GRADIENTS ESTIMATED BY EACH MODEL (15 TRIALS). *OUR METHODS

Model	Coordinate MSE (dB)	Energy MSE (dB)
Baseline NN	-4.90 ± 2.49	-9.06 ± 1.79
Hamiltonian NN [59]	-20.01 ± 1.28	-51.43 ± 0.68
RR [34]	-12.77 ± 1.74	-19.53 ± 2.82
GradNet-C*	-16.32 ± 1.70	-44.29 ± 0.55
GradNet-M*	**-23.84** ± 1.61	**-52.31** ± 0.59

Hence, the Hamiltonian formulation of the system dynamics provides a simple method for obtaining position, momentum, velocity, and force information. Following the procedure in [59], we train models to take q, p as input and output dq/dt and $-dp/dt$. By (24), training these models corresponds to learning gradients of an unknown Hamiltonian function – a direct and quintessential application for `GradNets`.

The system for the two-body problem is composed of two point particles interacting through an attractive force like gravity. It has the following nontrivial, nonconvex Hamiltonian:

$$\mathcal{H}(q,p)=\frac{\|p_{CM}\|_2^2}{m_1+m_2}+\frac{\|p_1\|_2^2+\|p_2\|_2^2}{2\mu}+\frac{gm_1m_2}{\|q_1-q_2\|_2^2} \tag{25}$$

where μ is the reduced mass and p_{CM} is the momentum of the center of mass. Using the experimental setup and training procedure[3] in [59], we consider a two-dimensional physical ambient space that induces eight degrees of freedom for the system: two-dimensional position and momentum for each object. This means $q, p \in \mathbb{R}^8$. We compare our networks with the Hamiltonian neural network (NN) and multi-layer perceptron baseline from [59]. Unlike the `GradNets` introduced in this paper, the Hamiltonian NN does not offer theoretical guarantees concerning universal approximation of gradient functions.

In Fig. 6, we plot the ground truth trajectories of the two bodies along with the trajectories unrolled using the estimated gradients of the Hamiltonian in (25). Table III shows the corresponding coordinate MSEs for the unrolled trajectories and reflects the results in Fig. 6. Since the energy in the system should remain constant, the energy MSEs in Table III reflect how effectively the Hamiltonian gradient predictions of each method conserve the total energy of the system. We see that the `GradNet-M` outperforms the Hamiltonian NN baseline by over 3 dB in terms of coordinate MSE and by approximately 1 dB in terms of energy MSE. Similarly, the `GradNet-M` outperforms the RR by 11 dB for coordinate MSE and 32 dB for energy MSE. We attribute this gain to the universal approximation capabilities of the `GradNet-M` discussed in Sec. V. The `GradNet-C` outperforms the RR and demonstrates better approximation capabilities for the gradient of (25), which is not easily approximated by a sum of ridge functions.

[3]Our results are obtained by evaluating all models trained using the hyperparameter search detailed in Appendix C.

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

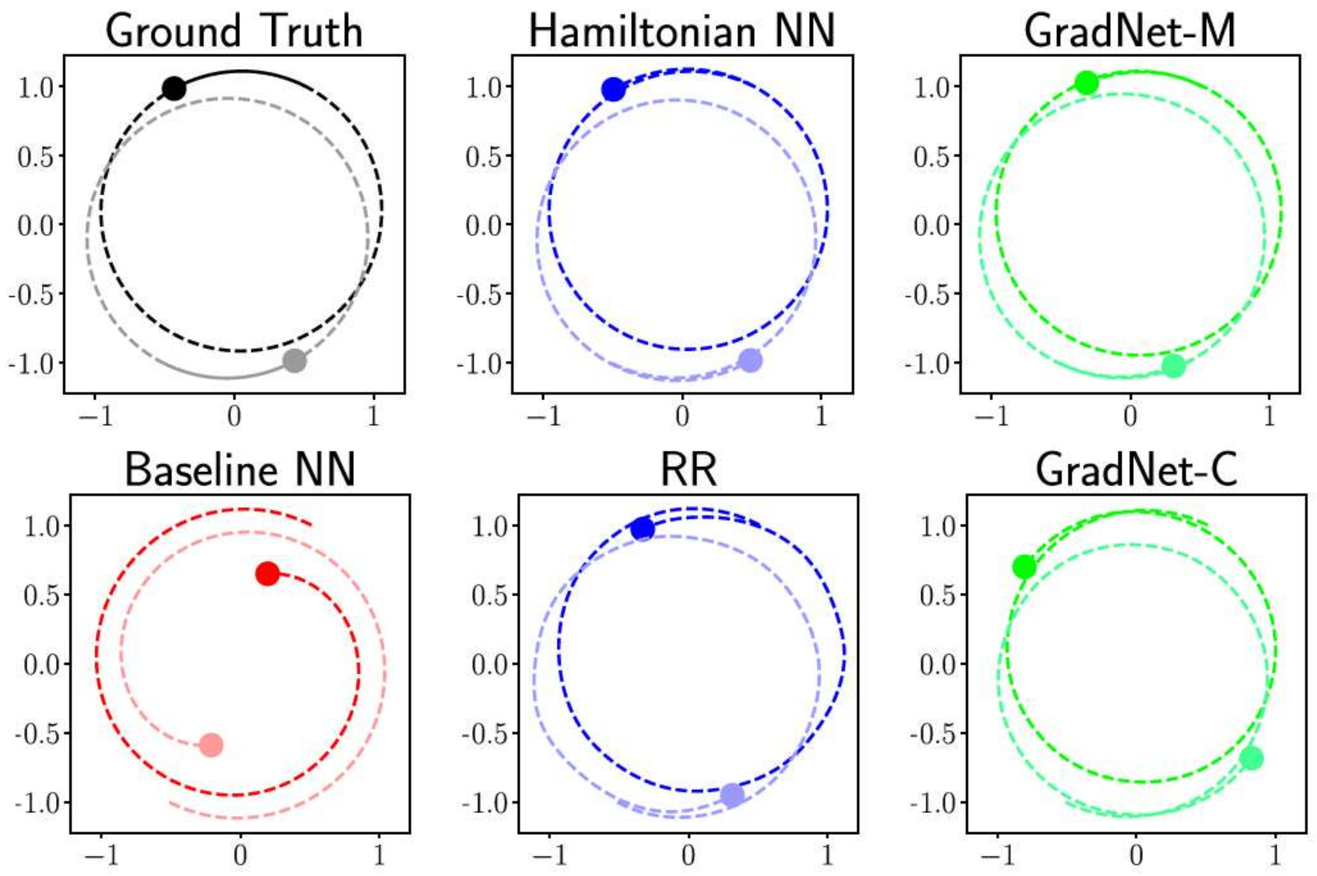


Fig. 6. 2-body problem object trajectories from unrolling dynamics using models that learn gradients of the Hamiltonian. Our proposed methods are in green.

VIII. Conclusion

In this work, we presented gradient networks (`GradNets`): novel classes of neural networks for directly parameterizing and learning gradients of functions. We first introduced gradient networks that are guaranteed to be gradients of scalar-valued functions. We subsequently demonstrated methods to convert gradient networks into monotone gradient networks (`mGradNets`) that are guaranteed to correspond to the gradients of convex functions. Notably, we proposed a general framework for designing (monotone) gradient networks and rigorously established their universal approximation capabilities for several key function classes. Our experimental results on gradient field and dynamics learning problems showcase the practical utility of `GradNets` and reinforce our theoretical findings. We observed improvements of up to 15 dB over existing methods for approximating gradient fields and achieved improvements of up to 11 dB for predicting the dynamics of the two-body problem.

Appendix

All models are trained using the Adam optimizer with standard momentum parameters 0.9 and 0.999.

A. *Gradient Field Experiment Details:* $d = 2$

The `mGradNet-M` has 4 modules, each with hidden dimension 7, $\rho(x) = 1$, and $\sigma(x)$ as the softmax function. The module outputs are conically combined via learnable, nonnegative parameters. The `GradNet-M` is identical to the described `mGradNet-M`, however the module outputs are combined linearly rather than conically. The `mGradNet-C` has $L = 3$ hidden layers, each with a hidden dimension of 7, elementwise tanh activation, and intermediate nonnegative weight vectors $\boldsymbol{\alpha}_\ell, \boldsymbol{\beta}_\ell$. The `GradNet-C` is identical to the mentioned `mGradNet-C` except α, β are unconstrained. We compare to a 2 hidden layer ICNN with hidden dimension 7 and softplus activations. The ICGN has the form $\sigma(\boldsymbol{W}x + \boldsymbol{b})$ with hidden dimension 32 and sigmoid activation. The CRR has hidden dimension 10 and each learnable spline activation has 7 knots. The RR has the same architecture as the CRR except the spline activation functions are not constrained to be nondecreasing. The MLP has 3 hidden layers and hidden dimension 6. All models are trained for 200 epochs with batch size 1000 and learning rate 0.005.

B. *Gradient Field Experiment Details:* $d \geq 32$

All models have the same architecture as described in Appendix A with some minor modifications. The CRR and RR each have 41 knots. The `mGradNet-C` has intermediate, learnable activations of the form $\alpha \tanh(x) + \beta(x - \tanh(x))$ where α, β are constrained to be nonnegative. The `GradNet-C` uses the same activation, but with unconstrained α, β. The `mGradNet-M` has intermediate, learnable

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

activation $\alpha\text{softmax}(x) - \beta\text{softmin}(x)$ where α, β are constrained to be nonnegative. The GradNet-M uses the same activation with unconstrained α, β. The hidden dimension of each model is scaled such that each model has $1024 \cdot d$ parameters. All models are trained for 10,000 iterations with randomly sampled batches of size 1,000 from $[0, 1]^d$. We test learning rates of 0.01, 0.001, and 0.0001 for each model, and report results corresponding to the learning rate that achieves the best performance. For each test trial, we evaluate the trained models on 10,000 points randomly sampled from $[0, 1]^d$.

C. Hamiltonian Experiment Details

We use the baseline MLP and Hamiltonian NN architectures described in [59] and tune over the hyperparameters listed in the appendix of [59]. To closely match the number of parameters used in the best-performing Hamiltonian NN with hidden dimension 100, we use a GradNet-M with 4 modules and hidden dimension 256, with $\rho(x) = 1$, and $\sigma(x)$ as the softmax function. The outputs of the modules are linearly combined via arbitrary learnable parameters. The GradNet-C has $L = 4$ hidden layers, each with a hidden dimension of 256, elementwise tanh activation, and intermediate weight vectors α_ℓ, β_ℓ. The RR has a hidden dimension of 220 and each learnable spline activation has 41 knots. We tuned the RR's range parameter over 0.01, 0.1, 1, and 10, and sparsity parameter over 0, 1e-1, 1e-2, 1e-4, 1e-6, and 1e-8. Like [59], we use a batch size of 200 and train for 10,000 steps. We tuned over learning rates 1e-1, 1e-2, 1e-3, 1e-4 and 1e-5 for each model, and report results corresponding to the best-performing model. For each trial at test time, we unroll from $t = 0$ to $t = 60$, where $\Delta t = 0.03$ (2000 steps).

Software Availability Statement

Code available at https://github.com/SPronav/Gradient Networks.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Adv. Neur. Inf. Process. Syst.*, vol. 25, 2012.

[2] A. Vaswani et al., "Attention is all you need," *Adv. Neur. Inf. Process. Syst.*, vol. 30, 2017.

[3] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] A. Hyvärinen and P. Dayan, "Estimation of non-normalized statistical models by score matching." *J. Mach. Learn. Res.*, vol. 6, no. 4, pp. 695–709, 2005.

[5] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Adv. Neur. Inf. Process. Syst.*, vol. 32, 2019.

[6] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *Proc. 9th Int. Conf. Learn. Represent. (ICLR), Virtual Event*, Austria, May 3–7, 2021. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=PxTIG12RRHS

[7] Y. Song and S. Ermon, "Improved techniques for training score-based generative models," *Adv. Neur. Inf. Process. Syst.*, vol. 33, pp. 12438–12448, 2020.

[8] R. Cai et al., "Learning gradient fields for shape generation," in *Proc. Comput. Vision–ECCV 2020: 16th Eur. Conf. (Part III 16)*, Glasgow, UK, August 23–28. Springer, 2020, pp. 364–381.

[9] Y. Brenier, "Polar factorization and monotone rearrangement of vector-valued functions," *Commun. Pure Appl. Math.*, vol. 44, no. 4, pp. 375–417, 1991.

[10] F. Santambrogio, "Optimal transport for applied mathematicians," *Birkäuser, NY*, vol. 55, no. 58–63, p. 94, 2015.

[11] A. Korotin, L. Li, A. Genevay, J. M. Solomon, A. Filippov, and E. Burnaev, "Do neural optimal transport solvers work? A continuous Wasserstein-2 benchmark," *Adv. Neur. Inf. Process. Syst.*, vol. 34, pp. 14593–14605, 2021.

[12] C. Bunne, A. Krause, and M. Cuturi, "Supervised training of conditional Monge maps," *Adv. Neur. Inf. Process. Syst.*, vol. 35, pp. 6859–6872, 2022.

[13] A. Makkuva, A. Taghvaei, S. Oh, and J. Lee, "Optimal transport mapping via input convex neural networks," in *Int. Conf. Mach. Learn.* PMLR, 2020, pp. 6672–6681.

[14] S. Chaudhari, S. Pranav, and J. M. F. Moura, "Learning gradients of convex functions with monotone gradient networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2023, pp. 1–5.

[15] J. Vongkulbhisal, F. De la Torre, and J. P. Costeira, "Discriminative optimization: Theory and applications to computer vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 829–843, Apr. 2018.

[16] Y. Romano, M. Elad, and P. Milanfar, "The little engine that could: Regularization by denoising (red)," *SIAM J. Imag. Sci.*, vol. 10, no. 4, pp. 1804–1844, Apr. 2017.

[17] E. T. Reehorst and P. Schniter, "Regularization by denoising: Clarifications and new interpretations," *IEEE Trans. Comput. Imag.*, vol. 5, no. 1, pp. 52–67, Jan. 2018.

[18] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global Conf. Signal Information Process.* Piscataway, NJ, USA: IEEE Press, 2013, pp. 945–948.

[19] U. S. Kamilov, C. A. Bouman, G. T. Buzzard, and B. Wohlberg, "Plug-and-play methods for integrating physical and learned models in computational imaging: Theory, algorithms, and applications," *IEEE Signal Process. Mag.*, vol. 40, no. 1, pp. 85–97, Jan. 2023.

[20] R. Cohen, Y. Blau, D. Freedman, and E. Rivlin, "It has potential: Gradient-driven denoisers for convergent solutions to inverse problems," *Adv. Neur. Inf. Process. Syst.*, vol. 34, pp. 18152–18164, 2021.

[21] S. Hurault, A. Leclaire, and N. Papadakis, "Gradient step denoiser for convergent plug-and-play," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2022. [Online]. Available: https://openreview.net/forum?id=fPhKeld3Okz

[22] R. G. Gavaskar, C. D. Athalye, and K. N. Chaudhury, "On plug-and-play regularization using linear denoisers," *IEEE Trans. Image Process.*, vol. 30, pp. 4802–4813, 2021.

[23] R. Fermanian, M. Le Pendu, and C. Guillemot, "PnP-ReG: Learned regularizing gradient for plug-and-play gradient descent," *SIAM J. Imag. Sci.*, vol. 16, no. 2, pp. 585–613, 2023.

[24] M. Andrychowicz et al., "Learning to learn by gradient descent by gradient descent," *Adv. Neur. Inf. Process. Syst.*, vol. 29, 2016.

[25] W. Rudin, *Principles of Mathematical Analysis*. New York, NY, USA: McGraw-Hill, 1964, vol. 3.

[26] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu, "Sobolev training for neural networks," *Adv. Neur. Inf. Process. Syst.*, vol. 30, 2017.

[27] S. Saremi, "On approximating ∇f with neural networks," 2019, *arXiv:1910.12744*.

[28] L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman, "Gradients are not all you need," 2021, *arXiv:2111.05803*.

[29] J. Richter-Powell, J. Lorraine, and B. Amos, "Input convex gradient networks," 2021, *arXiv:2111.12187*.

[30] S. Roth and M. J. Black, "Fields of experts," *Int. J. Comput. Vision*, vol. 82, pp. 205–229, 2009.

[31] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, no. 3, pp. 425–455, 1994.

[32] Y. Chen, R. Ranftl, and T. Pock, "Insights into analysis operator learning: From patch-based sparse models to higher order MRFS," *IEEE Trans. Image Process.*, vol. 23, no. 3, pp. 1060–1072, 2014.

[33] E. Kobler, T. Klatzer, K. Hammernik, and T. Pock, "Variational networks: Connecting variational methods and deep learning," in *Proc. Pattern Recognit.: 39th German Conf. (GCPR)*, Basel, Switzerland, September 12–15. New York: Springer, 2017, pp. 281–293.

[34] A. Goujon, S. Neumayer, P. Bohra, S. Ducotterd, and M. Unser, "A neural-network-based convex regularizer for inverse problems," *IEEE Trans. Comput. Imag.*, vol. 9, pp. 781–795, 2023.

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.

[35] A. Goujon, S. Neumayer, and M. Unser, "Learning weakly convex regularizers for convergent image-reconstruction algorithms," *SIAM J. Imag. Sci.*, vol. 17, no. 1, pp. 91–115, 2024.
[36] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *Int. Conf. Mach. Learn.* PMLR, 2017, pp. 146–155.
[37] Y. Chen, Y. Shi, and B. Zhang, "Optimal control via neural networks: A convex approach," in *Proc. Int. Conf. Learn. Represent.*, 2018.
[38] C.-W. Huang, R. T. Q. Chen, C. Tsirigotis, and A. Courville, "Convex potential flows: Universal probability distributions with optimal transport and convex optimization," in *Proc. Int. Conf. Learn. Represent.*, 2021. [Online]. Available: https://openreview.net/forum?id=te7PVH1sPxJ
[39] S. Sivaprasad, A. Singh, N. Manwani, and V. Gandhi, "The curious case of convex neural networks," in *Proc. Mach. Learn. Knowl. Discovery Databases. Res. Track: Eur. Conf. (ECML PKDD)—Part I 21*, Bilbao, Spain, September 13–17. New York: Springer, 2021, pp. 738–754.
[40] P.-J. Hoedt and G. Klambauer, "Principled weight initialisation for input-convex neural networks," *Adv. Neur. Inf. Process. Syst.*, vol. 36, 2024.
[41] J. Lorraine and S. Hossain, "JacNet: Learning functions with structured Jacobians," 2024, *arXiv:2408.13237*.
[42] P. J. Davis and P. Rabinowitz, *Methods of Numerical Integration*. Courier Corporation, 2007.
[43] T. Tao, *Analysis II*. New York: Springer, 2016.
[44] R. T. Rockafellar and R. J.-B. Wets, *Variational Analysis*. Springer Science & Business Media, 2009, vol. 317.
[45] E. Ryu, J. Liu, S. Wang, X. Chen, Z. Wang, and W. Yin, "Plug-and-play methods provably converge with properly trained denoisers," in *Int. Conf. Mach. Learn.* PMLR, 2019, pp. 5546–5557.
[46] M. Terris, A. Repetti, J.-C. Pesquet, and Y. Wiaux, "Building firmly nonexpansive convolutional neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*. Piscataway, NJ, USA: IEEE Press, 2020, pp. 8658–8662.
[47] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[48] S. Mukherjee, S. Dittmer, Z. Shumaylov, S. Lunz, O. Öktem, and C.-B. Schönlieb, "Learned convex regularizers for inverse problems," 2020, *arXiv:2008.02839*.
[49] R. T. Rockafellar, *Convex Analysis*, vol. 18. Princeton, NJ, USA: Princeton Univ. Press, 1970.
[50] A. H. Smith and W. A. Albrecht, *Fundamental Concepts of Analysis*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1966.
[51] G. C. Calafiore, S. Gaubert, and C. Possieri, "Log-sum-exp neural networks and posynomial models for convex and log-log-convex data," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 3, pp. 827–838, Mar. 2020.
[52] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Networks*, vol. 6, no. 6, pp. 861–867, Jun. 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608005801315
[53] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural Comput.*, vol. 5, no. 2, pp. 305–316, Feb. 1993.
[54] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.
[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 770–778.
[56] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Adv. Neur. Inf. Process. Syst.*, vol. 31, 2018.
[57] K. Svanberg, "A class of globally convergent optimization methods based on conservative convex separable approximations," *SIAM J. Optim.*, vol. 12, no. 2, pp. 555–573, 2002.
[58] A. Blum, J. Hopcroft, and R. Kannan, *Foundations of Data Science*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[59] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian neural networks," in *Adv. Neur. Inf. Process. Syst.*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2019/file/26cd8ecadce0d4efd6cc8a8725cbd1f8-Paper.pdf

Shreyas Chaudhari (Student Member, IEEE) received the B.S. degree in electrical and computer engineering from The Ohio State University, Columbus, OH, USA, in 2018, and the M.S. degree in 2021 from Carnegie Mellon University, Pittsburgh, PA, USA, where he is currently working toward the Ph.D. degree, both in electrical and computer engineering. In 2021, he was awarded the National Science Foundation Graduate Research Fellowship. His research interests include the foundational aspects of deep learning as well as applications of machine learning in signal processing.

Srinivasa Pranav (Student Member, IEEE) received the B.S. degree in electrical engineering and computer sciences, the B.A. degree in applied mathematics, the B.S. degree in business administration from the University of California at Berkeley, Berkeley, CA, USA, in 2019, and the M.S. degree in 2020 from Carnegie Mellon University, Pittsburgh, PA, USA, where he is currently working toward the Ph.D. degree, both in electrical and computer engineering. He is a National Science Foundation Graduate Research Fellow and ARCS Scholar. His research interests include machine learning, distributed optimization, and peer-to-peer deep learning.

José M.F. Moura (Life Fellow, IEEE) is the Philip L. and Marsha Dowd University Professor at Carnegie Mellon University, Pittsburgh, PA, USA. His research interests include statistical, algebraic, and graph signal processing. The technology of two of his 19 patents (co-inventor Alek Kavcic) is found in the read channel of over 4 billion hard disk drives (60% of all computers sold worldwide since 2003) and the subject of a 2016 $ 750 million settlement between Carnegie Mellon University and a major chip manufacturer. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, President of the IEEE SIGNAL PROCESSING SOCIETY, and 2019 IEEE President and CEO. He is a fellow of the AAAS and the US National Academy of Inventors, and a member of the Academy of Sciences, Portugal and of the US National Academy of Engineering. He was awarded doctor honoris causa by the University of Strathclyde, Scotland, U.K. and by Universidade de Lisboa, Portugal, the Great Cross of Prince Henry from the President of the Republic of Portugal, and the IEEE Jack S. Kilby Signal Processing Medal "[f]or contributions to theory and practice of statistical, graph, and distributed signal processing." He was recognized with the IEEE Haraden-Pratt Award.

Authorized licensed use limited to: Carnegie Mellon University Libraries. Downloaded on March 12,2025 at 12:13:03 UTC from IEEE Xplore. Restrictions apply.