arXiv:2002.09750v2 [math.NA] 4 Mar 2020

ON SOME NEURAL NETWORK ARCHITECTURES THAT CAN REPRESENT VISCOSITY SOLUTIONS OF CERTAIN HIGH DIMENSIONAL HAMILTON–JACOBI PARTIAL DIFFERENTIAL EQUATIONS

JÉRÔME DARBON AND TINGWEI MENG

Abstract. We propose novel connections between several neural network architectures and viscosity solutions of some Hamilton–Jacobi (HJ) partial differential equations (PDEs) whose Hamiltonian is convex and only depends on the spatial gradient of the solution. To be specific, we prove that under certain assumptions, the two neural network architectures we proposed represent viscosity solutions to two sets of HJ PDEs with zero error. We also implement our proposed neural network architectures using Tensorflow and provide several examples and illustrations. Note that these neural network representations can avoid curve of dimensionality for certain HJ PDEs, since they do not involve neither grids nor discretization. Our results suggest that efficient dedicated hardware implementation for neural networks can be leveraged to evaluate viscosity solutions of certain HJ PDEs.

1. Introduction

Hamilton–Jacobi (HJ) partial differential equations (PDEs) arise in areas such as physics [5, 19, 20, 25, 75], optimal control [8, 37, 45, 46, 81], game theory [11, 18, 39, 61], and imaging sciences [27, 29, 30]. In this paper, we consider HJ PDEs with state and time independent Hamiltonian function $H: \mathbb{R}^n \to \mathbb{R}$ and initial data $J: \mathbb{R}^n \to \mathbb{R}$ that read as follows

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x},t) + H(\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)) = 0 & \text{in } \mathbb{R}^n \times (0,+\infty), \\ S(\boldsymbol{x},0) = J(\boldsymbol{x}) & \text{in } \mathbb{R}^n. \end{cases} \tag{1}$$

The partial derivative with respect to t and the gradient vector with respect to $\boldsymbol{x}$ of the solution $(\boldsymbol{x},t) \mapsto S(\boldsymbol{x},t)$ are denoted by $\frac{\partial S}{\partial t}(\boldsymbol{x},t)$ and $\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t) = \left(\frac{\partial S}{\partial x_1}(\boldsymbol{x},t), \dots, \frac{\partial S}{\partial x_n}(\boldsymbol{x},t)\right)$, respectively. Note that the Hamiltonian H only depends on $\nabla_{\boldsymbol{x}} S(\boldsymbol{x},t)$.

Recently, [28] establishes novel connections between some neural network architectures and the viscosity solution of a set of HJ PDEs in the form of (1). (We refer readers to [8, 9, 10, 26] for the definition of the viscosity solution.) In [28], the authors provided the conditions under which their proposed neural network architecture represents the viscosity solution to the corresponding HJ PDEs whose initial data J and Hamiltonian H are related to the parameters in the neural network. Note that in the HJ PDEs they considered, the initial data J is assumed to be a convex piecewise affine function, and the Hamiltonian H also satisfies certain assumptions.

In this paper, we consider the HJ PDEs in the form of (1) satisfying other assumptions. For instance, the Hamiltonian H is convex, while the initial data J is not necessarily convex. Under these assumptions, we prove that the two neural network architectures depicted in Figs. 1 and 2 represent viscosity solutions to the corresponding HJ PDEs in the form of (1) with initial data J and convex Hamiltonian H. To be specific, in the first architecture shown in Fig. 1, the convex activation function L in the neural network gives the Lagrangian function, whose Fenchel–Legendre transform gives the Hamiltonian H in the corresponding HJ PDE. The initial data equals the minimum of several functions which are shifted copies of the asymptotic function L'_∞ of L. The main result of this connection between the neural network architecture depicted in Fig. 1 and the corresponding HJ PDE is stated in Thm. 3.1. In the second architecture shown in Fig. 2, the activation function gives the initial data J in the HJ PDE. The Hamiltonian H is a piecewise affine convex function determined by the parameters in the neural network. The main result of this connection between the neural network architecture depicted in Fig. 2 and the corresponding HJ PDE is stated in Thm. 3.2.

Date: Authors' names are given in last/family name alphabetical order.

To summarize, this paper investigates the connection between several neural network architectures and some specific sets of HJ PDEs. The motivations and advantages of this work are listed as follows

- Compared with traditional grid based representations, our proposed neural network representations do not involve any discretization of space and time. Hence these neural network representations can avoid the curse of dimensionality for certain HJ PDEs if the correct parameters are provided.
- Our novel connections between certain HJ PDEs and neural networks suggest a possible direction to solve some HJ PDEs by leveraging efficient hardware technologies and silicon-based electric circuits dedicated to neural networks. LeCun mentioned in [76] that the use of neural networks has been greatly influenced by available hardware. There have been many initiatives designing and constructing new hardware for extremely efficient (in terms of speed, latency, throughput or energy) implementations of neural networks. For instance, efficient neural network implementations are developed and optimized using field programmable gate arrays [40, 41, 42], Intel's architecture [7], Google's "Tensor Processor Unit" [64], and certain building blocks [70]. To obtain better performance on neural network computation, Xilinx announced a new set of hardware called Versal AI core, while Intel enhances their processors with specific hardware instructions. In addition, there is an evolution of silicon-based electrical circuits for machine learning, for which we refer readers to [23, 55]. LeCun also suggests in [76, Sec. 3] possible new trends for hardware dedicated to neural networks. These trends for efficient neural network implementations motivate our study of the connections between neural network architectures and HJ PDEs.
- This work provides a possible interpretation of specific neural networks from the aspect of HJ PDEs.

Literature review. There is a huge body of literature on overcoming the curse of dimensionality of certain HJ PDEs. These works include, but are not limited to, max-plus algebra methods [81, 1, 2, 35, 44, 49, 82, 83, 84], dynamic programming and reinforcement learning [3, 16], tensor decomposition techniques [34, 57, 104], sparse grids [17, 48, 67], model order reduction [4, 71], polynomial approximation [65, 66], optimization methods [27, 29, 30, 111] and neural networks [28, 6, 32, 62, 51, 59, 60, 74, 89, 97, 99, 101].

Recently, because of the trends for the efficient hardware implementations, neural networks have been increasingly applied in solving PDEs [6, 32, 51, 59, 60, 74, 89, 97, 99, 101, 13, 12, 14, 15, 22, 24, 31, 33, 36, 43, 47, 50, 52, 58, 63, 68, 69, 72, 73, 77, 80, 85, 86, 106, 91, 100, 102, 103, 107, 108, 109, 110] and inverse problems involving PDEs [109, 79, 78, 87, 88, 90, 92, 93, 95, 96, 94, 105, 112, 113]. Specifically, some high-dimensional HJ PDEs have been numerically solved using neural networks [28, 51, 60, 101]. In [101], the solution to HJ PDEs is approximated by a deep neural network whose loss function is the l^2 error of the PDE, the initial condition and the boundary condition on randomly sampled points in the domain. In [51], a neural network architecture is proposed to approximate a backward stochastic differential equation which computes the solution to a second order HJ PDE via an associated stochastic representation formula. In [60], Huré *et al.* approximate the solution and its gradient using two neural networks at each discretized time step. After the neural networks at a larger time t_{j+1} are trained, the neural networks at t_j are trained with loss function given by the error of the stochastic representation formula. In [28], a neural network architecture is proposed for representing the viscosity solution to certain high dimensional HJ PDEs without error. In addition, Cárdenas and Gibou [21] use neural networks to compute the mean curvature of the implicit level set function, which is the solution to a specific HJ PDE called level set equation.

Organization of this paper. This paper investigates the connections between two neural network architectures shown in Figs. 1 and 2 and the viscosity solution of some HJ PDEs whose initial data and Hamiltonian satisfy specific assumptions. In Sec. 2, we introduce basic concepts in finite dimensional convex analysis which will be used later in this paper. In Sec. 3, we present the main results. To be specific, we propose two neural network architectures. The first architecture is analyzed in Sec. 3.1, while the second one is analyzed in Sec. 3.2. Thms. 3.1 and 3.2 state that the neural network architectures shown in Figs. 1 and 2 represent viscosity solutions to the HJ PDEs with convex Hamiltonian H and initial data J satisfying certain assumptions. We provide several examples and illustrations after each theorem. Finally, a conclusion is drawn in Sec. 4.

2. Background

In this section, we introduce related concepts in convex analysis that will be used in this paper. We refer readers to Hiriart–Urruty and Lemaréchal [53, 54] and Rockafellar [98] for comprehensive references on finite-dimensional convex analysis. For the notation, we use $\mathbb{R}^n$ to denote the n-dimensional Euclidean space, on which the Euclidean scalar product is denoted by $\langle \cdot, \cdot \rangle$.

Definition 1. *(Convex sets and the unit simplex) A set $C \subset \mathbb{R}^n$ is called convex if for any $\lambda \in [0,1]$ and any $\boldsymbol{x}, \boldsymbol{y} \in C$, the element $\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}$ is in C. The unit simplex is a specific convex set in $\mathbb{R}^n$, denoted by Λ_n, defined by*

$$\Lambda_n \coloneqq \left\{ (\alpha_1, \dots, \alpha_n) \in [0,1]^n : \sum_{i=1}^{n} \alpha_i = 1 \right\}. \tag{2}$$

Definition 2. *(Domains and proper functions) The domain of a function $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is the set*

$$\operatorname{dom} f = \{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) < +\infty\}.$$

A function $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is called proper if its domain is non-empty.

Definition 3. *(Convex functions, concave functions and lower semicontinuity) A proper function $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is called convex if the set $\operatorname{dom} f$ is convex and if for any $\boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} f$ and all $\lambda \in [0,1]$, there holds*

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leqslant \lambda f(\boldsymbol{x}) + (1-\lambda) f(\boldsymbol{y}).$$

A function $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{-\infty\}$ is called concave if $-f$ is a convex function. A proper function $f\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is called lower semicontinuous if for every sequence $\{\boldsymbol{x}_k\}_{k=1}^{+\infty}$ in $\mathbb{R}^n$ with $\lim_{k\to+\infty} \boldsymbol{x}_k = \boldsymbol{x} \in \mathbb{R}^n$, we have $\liminf_{k\to+\infty} f(\boldsymbol{x}_k) \geqslant f(\boldsymbol{x})$. The class of proper, lower semicontinuous convex functions is denoted by $\Gamma_0(\mathbb{R}^n)$.

Definition 4. *(Fenchel–Legendre transform) Let $f \in \Gamma_0(\mathbb{R}^n)$. The Fenchel–Legendre transform $f^*\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ of f is defined as*

$$f^*(\boldsymbol{p}) = \sup_{\boldsymbol{x} \in \mathbb{R}^n} \{\langle \boldsymbol{p}, \boldsymbol{x} \rangle - f(\boldsymbol{x})\}.$$

For any $f \in \Gamma_0(\mathbb{R}^n)$, the mapping $f \mapsto f^*$ is one-to-one. Moreover, there hold $f^* \in \Gamma_0(\mathbb{R}^n)$ and $(f^*)^* = f$.

Definition 5. *(Inf-convolution) Let $f, g\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be two proper convex functions satisfying*

$$f(\boldsymbol{x}) \geq \langle \boldsymbol{p}, \boldsymbol{x} \rangle + a \quad \text{and} \quad g(\boldsymbol{x}) \geq \langle \boldsymbol{p}, \boldsymbol{x} \rangle + a \text{ for every } \boldsymbol{x} \in \mathbb{R}^n, \tag{3}$$

for some $\boldsymbol{p} \in \mathbb{R}^n$ and $a \in \mathbb{R}$. The inf-convolution of f and g, denoted by $f \square g$, is defined by

$$f \square g(\boldsymbol{x}) = \inf_{\boldsymbol{u} \in \mathbb{R}^n} \{f(\boldsymbol{u}) + g(\boldsymbol{x} - \boldsymbol{u})\}.$$

Moreover, the function $f \square g\colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is a proper and convex function [53, Prop. IV.2.3.2].

Definition 6. *(Asymptotic function) Let f be a function in $\Gamma_0(\mathbb{R}^n)$ and $\boldsymbol{x}_0$ be an arbitrary point in $\operatorname{dom} f$. The asymptotic function of f, denoted by f'_∞, is defined by*

$$f'_\infty(\boldsymbol{d}) = \sup_{s>0} \frac{f(\boldsymbol{x}_0 + s\boldsymbol{d}) - f(\boldsymbol{x}_0)}{s} = \lim_{s\to+\infty} \frac{f(\boldsymbol{x}_0 + s\boldsymbol{d}) - f(\boldsymbol{x}_0)}{s}, \tag{4}$$

for every $\boldsymbol{d} \in \mathbb{R}^n$. In fact, this definition does not depend on the point $\boldsymbol{x}_0$. Moreover, the asymptotic function f'_∞ is convex and positive $1-$homogeneous, i.e., $f'_\infty(\alpha \boldsymbol{d}) = \alpha f'_\infty(\boldsymbol{d})$ for every $\alpha > 0$ and $\boldsymbol{d} \in \mathbb{R}^n$. For details, see [53, Chap. IV.3.2]

We summarize some notations and definitions in Tab. 1.

TABLE 1. Notations used in this paper. Here, we use f, g to denote functions from $\mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$ and $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{p}, \boldsymbol{d}$ to denote vectors in $\mathbb{R}^n$. For simplicity, we omit the assumptions in the definitions.

Notation	Meaning	Definition
$\langle \cdot, \cdot \rangle$	Euclidean scalar product in $\mathbb{R}^n$	$\langle \boldsymbol{x}, \boldsymbol{y} \rangle := \sum_{i=1}^n x_i y_i$
Λ_n	The unit simplex in $\mathbb{R}^n$	$\{(\alpha_1, \dots, \alpha_n) \in [0,1]^n : \sum_{i=1}^n \alpha_i = 1\}$
$\operatorname{dom} f$	The domain of f	$\{\boldsymbol{x} \in \mathbb{R}^n : f(\boldsymbol{x}) < +\infty\}$
$\Gamma_0(\mathbb{R}^n)$	A useful and standard class of convex functions	The set containing all proper, convex, lower semi-continuous functions from $\mathbb{R}^n$ to $\mathbb{R} \cup \{+\infty\}$
f^*	Fenchel–Legendre transform of f	$f^*(\boldsymbol{p}) := \sup_{\boldsymbol{x} \in \mathbb{R}^n} \{\langle \boldsymbol{p}, \boldsymbol{x} \rangle - f(\boldsymbol{x})\}$
$f \square g$	Inf-convolution of f and g	$f \square g(\boldsymbol{x}) = \inf_{\boldsymbol{u} \in \mathbb{R}^n} \{f(\boldsymbol{u}) + g(\boldsymbol{x} - \boldsymbol{u})\}$
f'_∞	The asymptotic function of f	$f'_\infty(\boldsymbol{d}) = \sup_{s>0} \left\{ \frac{1}{s}(f(\boldsymbol{x}_0 + s\boldsymbol{d}) - f(\boldsymbol{x}_0)) \right\}$

3. MAIN RESULTS

In this paper, we consider the HJ PDE given by

$$\begin{cases} \dfrac{\partial S}{\partial t}(\boldsymbol{x}, t) + H(\nabla_{\boldsymbol{x}} S(\boldsymbol{x}, t)) = 0 & \text{in } \mathbb{R}^n \times (0, +\infty), \\ S(\boldsymbol{x}, 0) = J(\boldsymbol{x}) & \text{in } \mathbb{R}^n, \end{cases} \tag{5}$$

where $H \colon \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is called Hamiltonian, and $J \colon \mathbb{R}^n \to \mathbb{R}$ is the initial data. It is well-known that when H is convex, the viscosity solution is given by the Lax-Oleinik formula [9, 38, 56] stated as follows

$$S_{LO}(\boldsymbol{x}, t) = \inf_{\boldsymbol{u} \in \mathbb{R}^n} \left\{ J(\boldsymbol{u}) + tH^* \left(\frac{\boldsymbol{x} - \boldsymbol{u}}{t} \right) \right\} = \inf_{\boldsymbol{v} \in \mathbb{R}^n} \{ J(\boldsymbol{x} - t\boldsymbol{v}) + tH^*(\boldsymbol{v}) \}, \tag{6}$$

where H^* is the Fenchel–Legendre transform of H.

In this part, we represent the Lax-Oleinik formula using two neural network architectures. The first one is given by

$$f_1(\boldsymbol{x}, t) = \min_{i \in \{1, \dots, m\}} \left\{ tL \left(\frac{\boldsymbol{x} - \boldsymbol{u}_i}{t} \right) + a_i \right\}. \tag{7}$$

In this function, $\{(\boldsymbol{u}_i, a_i)\}_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$ is the set of parameters, and the function $L \colon \mathbb{R}^n \to \mathbb{R}$ is the activation function, which corresponds to the Lagrangian function in the Hamilton–Jacobi theory. An illustration is shown in Fig. 1.

The second neural network architecture is defined by

$$f_2(\boldsymbol{x}, t) = \min_{i \in \{1, \dots, m\}} \left\{ \tilde{J}(\boldsymbol{x} - t\boldsymbol{v}_i) + tb_i \right\}. \tag{8}$$

Here, $\{(\boldsymbol{v}_i, b_i)\}_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$ is the set of parameters, and $\tilde{J} \colon \mathbb{R}^n \to \mathbb{R}$ is the activation function, which corresponds to the initial function in the HJ PDE. An illustration is shown in Fig. 2.

3.1. The first architecture.

In this subsection, we analyze the first neural network architecture given by Eq. (7). Before introducing the main theorem 3.1 in this subsection, we prove the following lemma which will be used in the proof of Thm. 3.1.

Lemma 3.1. *Let f be a function in $\Gamma_0(\mathbb{R}^n)$ and f'_∞ be the asymptotic function of f. Then, we have $f \square f'_\infty = f$.*

Proof. First we consider the case when $\boldsymbol{x} \in \operatorname{dom} f$. By definition 5 we have

$$(f \square f'_\infty)(\boldsymbol{x}) = \inf_{\boldsymbol{u} \in \mathbb{R}^n} \{ f(\boldsymbol{u}) + f'_\infty(\boldsymbol{x} - \boldsymbol{u}) \} \le f(\boldsymbol{x}) + f'_\infty(\boldsymbol{0}) = f(\boldsymbol{x}),$$

where the last equality holds because $f'_\infty(\boldsymbol{0}) = 0$ by definition 6. On the other hand, taking $s = 1$, $\boldsymbol{d} = \boldsymbol{x} - \boldsymbol{u}$ and $\boldsymbol{x}_0 = \boldsymbol{u}$ in the second term in Eq. (4) in definition 6, we obtain

$$f'_\infty(\boldsymbol{x} - \boldsymbol{u}) \ge f(\boldsymbol{u} + \boldsymbol{x} - \boldsymbol{u}) - f(\boldsymbol{u}) = f(\boldsymbol{x}) - f(\boldsymbol{u}), \tag{9}$$

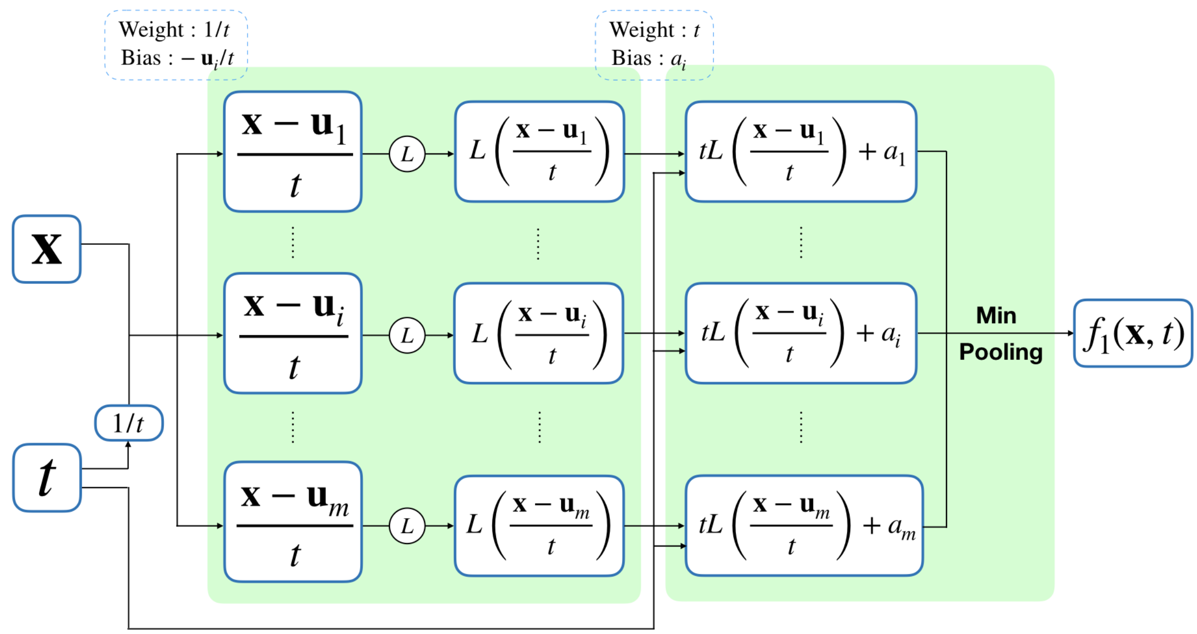


FIGURE 1. An illustration of the architecture of the neural network (7) that represents the Lax-Oleinik formula with specific initial condition $J = f_1(\cdot, 0)$ defined in (10) and the convex Hamiltonian $H = L^*$.

for every $\boldsymbol{u} \in \text{dom } f$. As a result, we have

$$(f \Box f'_\infty)(\boldsymbol{x}) = \inf_{\boldsymbol{u}\in \text{dom } f} \{f(\boldsymbol{u}) + f'_\infty(\boldsymbol{x}-\boldsymbol{u})\} \geq \inf_{\boldsymbol{u}\in \text{dom } f} \{f(\boldsymbol{u}) + f(\boldsymbol{x}) - f(\boldsymbol{u})\} = f(\boldsymbol{x}).$$

Therefore, we conclude that $(f \Box f'_\infty)(\boldsymbol{x}) = f(\boldsymbol{x})$ for every $\boldsymbol{x} \in \text{dom } f$.

Now we consider the case when $\boldsymbol{x} \notin \text{dom } f$ and prove $(f \Box f'_\infty)(\boldsymbol{x}) = +\infty$. It suffices to prove $f'_\infty(\boldsymbol{x}-\boldsymbol{u}) = +\infty$ for all $\boldsymbol{u} \in \text{dom } f$. Since $\boldsymbol{u} \in \text{dom } f$, Eq. (9) still holds. As a result, we have

$$f'_\infty(\boldsymbol{x}-\boldsymbol{u}) \geq f(\boldsymbol{x}) - f(\boldsymbol{u}) = +\infty,$$

since $\boldsymbol{x} \notin \text{dom } f$ and $\boldsymbol{u} \in \text{dom } f$. Therefore, we conclude that $(f \Box f'_\infty)(\boldsymbol{x}) = +\infty = f(\boldsymbol{x})$ for every $\boldsymbol{x} \notin \text{dom } f$. $\square$

Now, we define the initial data $f_1(\cdot, 0) : \mathbb{R}^n \to \mathbb{R}$ as follows

$$f_1(\boldsymbol{x}, 0) = \min_{i\in\{1,\dots,m\}} \{L'_\infty(\boldsymbol{x}-\boldsymbol{u}_i) + a_i\}, \tag{10}$$

where L'_∞ is the asymptotic function of L. Then, we present the main theorem stating that the function f_1 solves the HJ PDE (5) with the initial condition given by $J = f_1(\cdot, 0)$ defined in (10) and the convex Hamiltonian H which is the Fenchel–Legendre transform of L.

Theorem 3.1. *Let $L\colon \mathbb{R}^n \to \mathbb{R}$ be a convex uniformly Lipschitz function. Let f_1 be the function defined in* (7)*. Then $f_1 = S_{LO}$, where S_{LO} is the Lax–Oleinik formula in* (6) *with the initial condition $J = f_1(\cdot, 0)$ defined in* (10) *and the convex Hamiltonian defined by $H = L^*$. Therefore, f_1 is a viscosity solution to the corresponding HJ PDE* (5)*.*

Remark 3.1. *In the theorem above, we assume L to be a convex uniform Lipschitz function, which implies that its Fenchel–Legendre transform H has bounded domain, and hence H may take the value $+\infty$ somewhere. As a result, the uniqueness theorem of the viscosity solution in [38, Chap. 10.2] does not hold. To our knowledge, we are not aware of any uniqueness result of the viscosity solution to the HJ PDEs where* dom H *is bounded.*

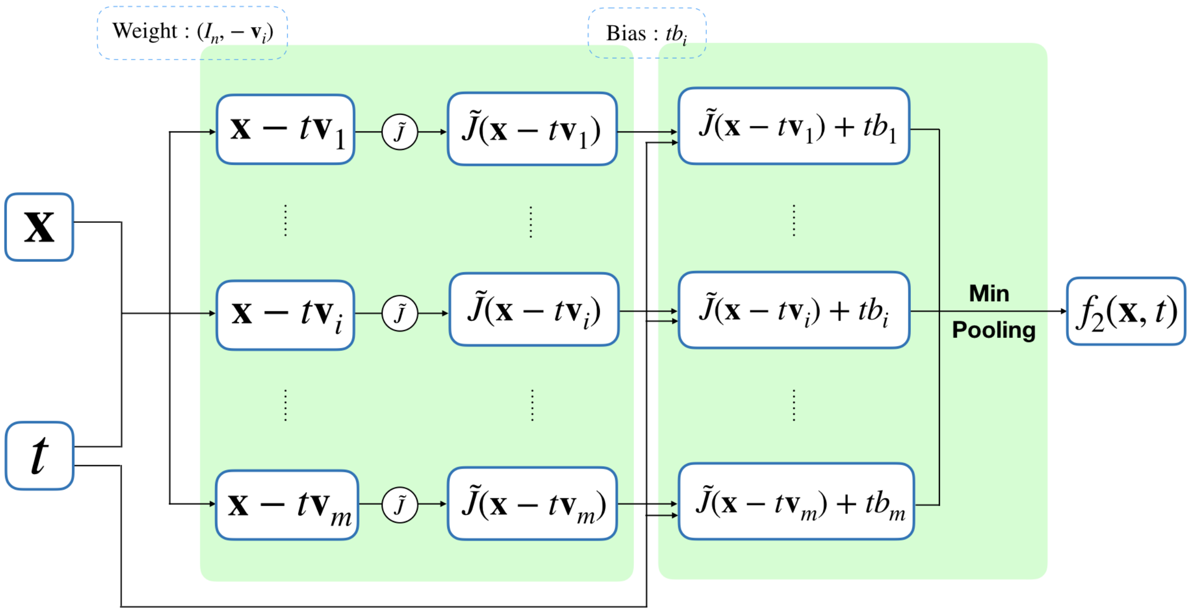


FIGURE 2. An illustration of the architecture of the neural network (8) that represents the Lax-Oleinik formula with specific initial condition $J = \tilde{J}$ and the convex Hamiltonian H defined in (12).

Proof. Since L is Lipschitz continuous, by [53, Prop. IV.3.2.7] L'_∞ is finite valued, which implies that $\mathbb{R}^n \ni \boldsymbol{x} \mapsto f_1(\boldsymbol{x}, 0)$ is finite valued and it is a valid initial condition.

Let $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$. By definition 5 and (10), we have

$$\begin{aligned}
S_{LO}(\boldsymbol{x}, t) &= \inf_{\boldsymbol{u}\in\mathbb{R}^n} \left\{ J(\boldsymbol{u}) + tH^*\left(\frac{\boldsymbol{x}-\boldsymbol{u}}{t}\right)\right\} = \inf_{\boldsymbol{u}\in\mathbb{R}^n} \left\{ \min_{i\in\{1,\dots,m\}} \{L'_\infty(\boldsymbol{u}-\boldsymbol{u}_i) + a_i\} + tH^*\left(\frac{\boldsymbol{x}-\boldsymbol{u}}{t}\right)\right\} \\
&= \min_{i\in\{1,\dots,m\}} \left\{ a_i + \inf_{\boldsymbol{u}\in\mathbb{R}^n} \left\{ L'_\infty(\boldsymbol{u}-\boldsymbol{u}_i) + tH^*\left(\frac{\boldsymbol{x}-\boldsymbol{u}}{t}\right)\right\}\right\} \\
&= \min_{i\in\{1,\dots,m\}} \left\{ a_i + \left(L'_\infty \Box tH^*\left(\frac{\cdot}{t}\right)\right)(\boldsymbol{x}-\boldsymbol{u}_i)\right\}.
\end{aligned} \tag{11}$$

Since L is convex with $\operatorname{dom} L = \mathbb{R}^n$, then the function L is continuous [53, Thm. IV.3.1.2]. As a result, L is a function in $\Gamma_0(\mathbb{R}^n)$, hence we have $L = (L^*)^*$, which equals H^* because we assume $H = L^*$. Let $t > 0$ and $h : \mathbb{R}^n \to \mathbb{R}$ be defined by $h(\boldsymbol{x}) = tH^*\left(\frac{\boldsymbol{x}}{t}\right) = tL\left(\frac{\boldsymbol{x}}{t}\right)$ for every $\boldsymbol{x} \in \mathbb{R}^n$. Let $\boldsymbol{x}_0$ be an arbitrary point in $\operatorname{dom} h$, which implies $\frac{\boldsymbol{x}_0}{t} \in \operatorname{dom} L$. By definition 6, the asymptotic function of h evaluated at $\boldsymbol{d}$ is given by

$$\begin{aligned}
h'_\infty(\boldsymbol{d}) &= \sup_{s>0} \left\{ \frac{1}{s}\left(tL\left(\frac{\boldsymbol{x}_0 + s\boldsymbol{d}}{t}\right) - tL\left(\frac{\boldsymbol{x}_0}{t}\right)\right)\right\} = \sup_{s>0} \left\{ \frac{t}{s}\left(L\left(\frac{\boldsymbol{x}_0}{t} + \frac{s}{t}\boldsymbol{d}\right) - L\left(\frac{\boldsymbol{x}_0}{t}\right)\right)\right\} \\
&= \sup_{\tau>0} \left\{ \frac{1}{\tau}\left(L\left(\frac{\boldsymbol{x}_0}{t} + \tau\boldsymbol{d}\right) - L\left(\frac{\boldsymbol{x}_0}{t}\right)\right)\right\} = L'_\infty(\boldsymbol{d}),
\end{aligned}$$

where in the third equality we set $\tau = \frac{s}{t}$. Hence, using the equality above, the definition of h and by invoking Lem. 3.1, we obtain

$$\left(L'_\infty \Box tH^*\left(\frac{\cdot}{t}\right)\right)(\boldsymbol{x}-\boldsymbol{u}_i) = (h'_\infty \Box h)(\boldsymbol{x}-\boldsymbol{u}_i) = h(\boldsymbol{x}-\boldsymbol{u}_i) = tL\left(\frac{\boldsymbol{x}-\boldsymbol{u}_i}{t}\right).$$

We combine the equality above with (11), to obtain

$$S_{LO}(\boldsymbol{x},t) = \min_{i\in\{1,\dots,m\}} \left\{ a_i + \left(L'_\infty \square t H^* \left(\frac{\cdot}{t}\right)\right)(\boldsymbol{x}-\boldsymbol{u}_i)\right\} = \min_{i\in\{1,\dots,m\}} \left\{ a_i + tL\left(\frac{\boldsymbol{x}-\boldsymbol{u}_i}{t}\right)\right\} = f_1(\boldsymbol{x},t).$$

Therefore, we conclude that $S_{LO}(\boldsymbol{x},t) = f_1(\boldsymbol{x},t)$ for each $\boldsymbol{x}\in\mathbb{R}^n$ and $t>0$. Then, using the same proof as in [38, Sec. 10.3.4, Thm. 3], we conclude that f_1 is a viscosity solution to the corresponding HJ PDE (5). $\square$

Example 3.1. *Let us consider the following one dimensional example that illustrates the function $f_1\colon \mathbb{R}\times[0,+\infty)\to\mathbb{R}$ with three neurons, i.e., we set $n=1$ and $m=3$. The Lagrangian L is defined as follows*

$$L(x) = \begin{cases} -x-\dfrac{1}{2} & x<-1,\\ \dfrac{x^2}{2} & -1\leq x\leq 2,\\ 2x-2 & x>2, \end{cases}$$

for each $x\in\mathbb{R}$. Then, by Thm. 3.1, the Hamiltonian H is given by

$$H(p) = L^*(p) = \begin{cases} \dfrac{p^2}{2} & -1\leq p\leq 2,\\ +\infty & \textit{otherwise.} \end{cases}$$

Also, by Thm. 3.1, the initial data J is given by $f_1(\cdot,0)$ defined in (10). In other words, J is the minimum of three functions, each of which is a shift of the function L'_∞, which by definition 6 reads as follows

$$L'_\infty(x) = \begin{cases} -x & x<0,\\ 2x & x\geq 0. \end{cases}$$

In this example, we choose the parameters $(u_1,a_1)=(-2,-0.5)$, $(u_2,a_2)=(0,0)$ and $(u_3,a_3)=(2,-1)$. The corresponding functions J, H and f_1 are shown in Fig. 3, where (a) shows the initial value J, (b) shows the convex Hamiltonian H, and (c) and (d) show the solution $S=f_1$ evaluated at $t=1$ and $t=3$, respectively.

The corresponding Tensorflow code reads as follows.

```
import numpy as np
import tensorflow as tf

n_data = 1000
dim = 1

def min_fn(x):
    return tf.math.reduce_min(x, axis=-1)

# L = -x-1/2 if x<-1;
#   = x^2/2 if -1<=x<=2;
#   = 2x-2 if x>2.
def L_fn(x):
    val1 = -x - 0.5
    val2 = tf.multiply(x,x)/2
    val3 = 2*x - 2
    flag1 = 1 - tf.sign(tf.maximum(x+1, 0))
    flag3 = tf.sign(tf.maximum(x-2, 0))
    flag2 = 1 - flag1 - flag3
    val = tf.multiply(flag1, val1)
    val = tf.add(val, tf.multiply(flag2, val2))
    val = tf.add(val, tf.multiply(flag3, val3))
    return tf.squeeze(val, -1)

tf.reset_default_graph()
```

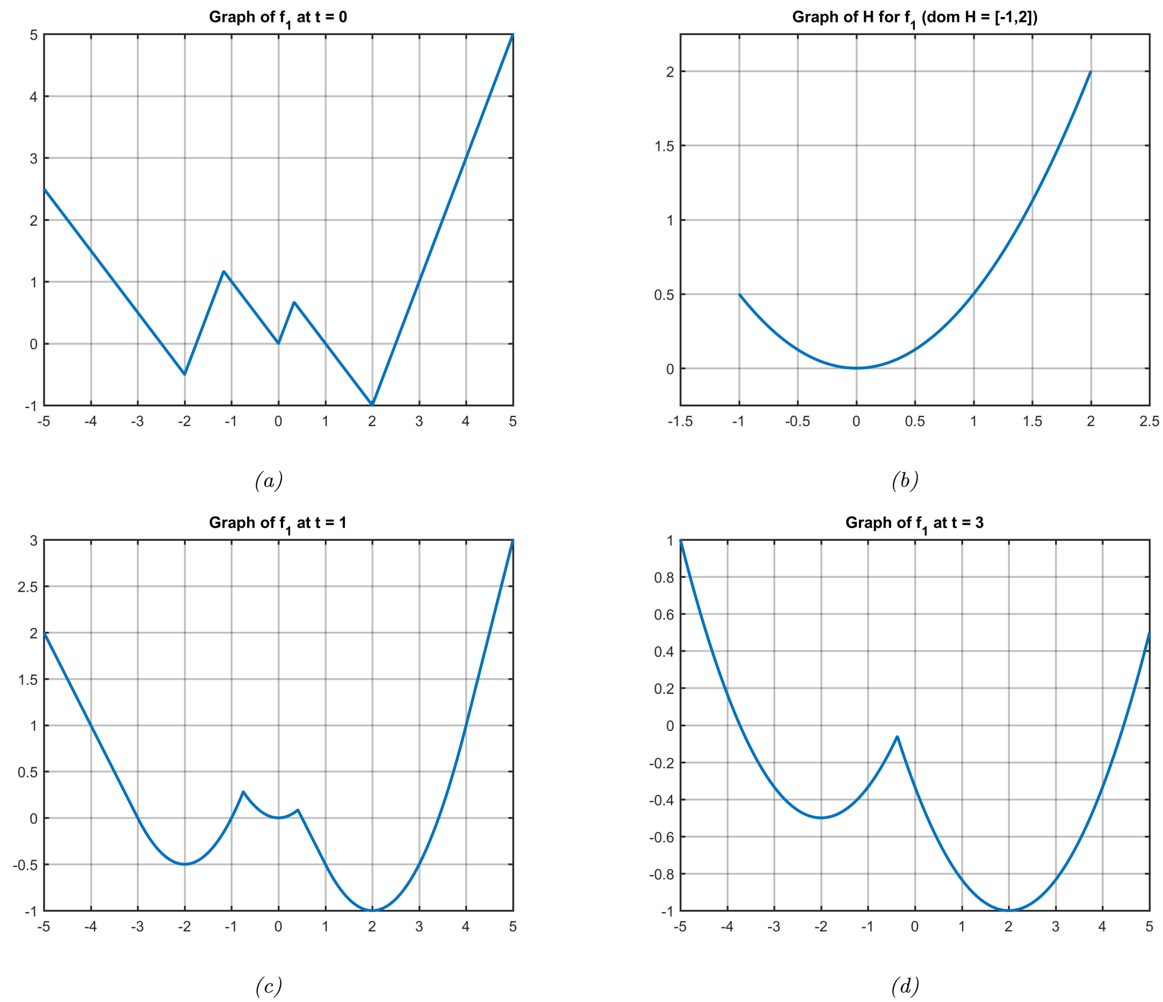


FIGURE 3. The graph of f_1 in example 3.1. The figures (a) and (b) show the initial value J and the Hamiltonian H, respectively. The figures (c) and (d) show the solution $S = f_1$ evaluated at $t = 1$ and $t = 3$, respectively.

```
u_true = np.array([[-2], [0], [2]])
a_true = np.array([-0.5, 0, -1])
u_param = tf.Variable(u_true, name = "u0", dtype = tf.float64)
a_param = tf.Variable(a_true, name = "a0", dtype = tf.float64)
x_placeholder = tf.placeholder(tf.float64, shape=(n_data, 1, dim))
t_placeholder = tf.placeholder(tf.float64, shape=(n_data, 1))

x_grid0 = np.arange(n_data) * (10.0 / n_data) - 5
t_grid0 = np.arange(n_data) * 0 + 3
x_grid = np.expand_dims(x_grid0, axis = -1)
x_grid = np.expand_dims(x_grid, axis = -1)
t_grid = np.expand_dims(t_grid0, axis = -1)

def construct_nn(x_in, t_in):
    val0 = tf.subtract(x_in, u_param)
    val1 = tf.div(val0, tf.expand_dims(t_in, -1))
    val2 = L_fn(val1)
```

```
    val3 = tf.add(tf.multiply(val2, t_in), a_param)
    y_ = min_fn(val3)
    return y_

y_nn = construct_nn(x_placeholder, t_placeholder)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
y_val = sess.run(y_nn, {x_placeholder: x_grid, t_placeholder: t_grid})
sess.close()
```

Example 3.2. *We now present a high dimensional example. To be specific, the dimension is set to be $n = 10$, and the solution $f_1 : \mathbb{R}^{10} \times [0, +\infty) \to \mathbb{R}$ is represented by a neural network with three neurons, i.e., $m = 3$. The activation function L is given by*

$$L(\boldsymbol{x}) = \max\{\|\boldsymbol{x}\|_2 - 1, 0\} = \begin{cases} \|\boldsymbol{x}\|_2 - 1 & \text{if } \|\boldsymbol{x}\|_2 > 1, \\ 0 & \text{if } \|\boldsymbol{x}\|_2 \leq 1. \end{cases}$$

The corresponding Hamiltonian is given by

$$H(\boldsymbol{p}) = L^*(\boldsymbol{p}) = \begin{cases} \|\boldsymbol{p}\|_2 & \text{if } \|\boldsymbol{p}\|_2 \leq 1, \\ +\infty & \text{if } \|\boldsymbol{p}\|_2 > 1. \end{cases}$$

The parameters are chosen to be $\boldsymbol{u}_1 = (-2, 0, 0, 0, \dots, 0)$, $\boldsymbol{u}_2 = (2, -2, -1, 0, \dots, 0)$, $\boldsymbol{u}_3 = (0, 2, 0, 0, \dots, 0)$, $a_1 = -0.5$, $a_2 = 0$ and $a_3 = -1$.

By definition 6 and straightforward computation, we obtain $L'_\infty(\boldsymbol{d}) = \|\boldsymbol{d}\|_2$. Hence, the initial condition for the corresponding HJ PDE is given by Eq. (10)*, which in this example reads*

$$J(\boldsymbol{x}) = \min_{i \in \{1,2,3\}} \{\|\boldsymbol{x} - \boldsymbol{u}_i\|_2 + a_i\}.$$

The accompanying figure 4 shows the graph of of f_1 for a 2-dimensional slice. To be specific, we fix $\boldsymbol{x} = (x_1, x_2, 0, \dots, 0)$, and compute $f_1(\boldsymbol{x}, t)$ at $t = 10^{-6}$, 1, 3 and 5. Note that the formula (7) *is not well-defined for $t = 0$, hence we use a small number 10^{-6} instead. In each figure, the color is given by the function value $f_1(\boldsymbol{x}, t)$ and the x and y axes represent the variables x_1 and x_2, respectively. The solutions evaluated at $t = 10^{-6}$, $t = 1$, $t = 3$ and $t = 5$ are shown in (a), (b), (c) and (d), respectively.*

3.2. **The second architecture.**

In this part, we analyze the second neural network architecture given by Eq. (8). Here, we assume the parameters $\{(\boldsymbol{v}_i, b_i)\}_{i=1}^m$ satisfy the following assumption

(H) There exists a convex function $\ell \colon \mathbb{R}^n \to \mathbb{R}$ satisfying $\ell(\boldsymbol{v}_i) = b_i$ for all $i \in \{1, \dots, m\}$.

Under this assumption, we present the following main theorem which states that the second architecture gives a viscosity solution to the corresponding HJ PDE, where the initial data is given by the activation function $\tilde{J}$ in the neural network, and the Hamiltonian is a convex piecewise affine function determined by the parameters $\{(\boldsymbol{v}_i, b_i)\}_{i=1}^m$.

Theorem 3.2. *Assume the function $\tilde{J} \colon \mathbb{R}^n \to \mathbb{R}$ is a concave function and the assumption (H) is satisfied. Let f_2 be the function defined in* (8)*. Then $f_2 = S_{LO}$, where S_{LO} is the Lax–Oleinik formula defined by* (6) *with initial condition $J = \tilde{J}$ and the Hamiltonian H defined by*

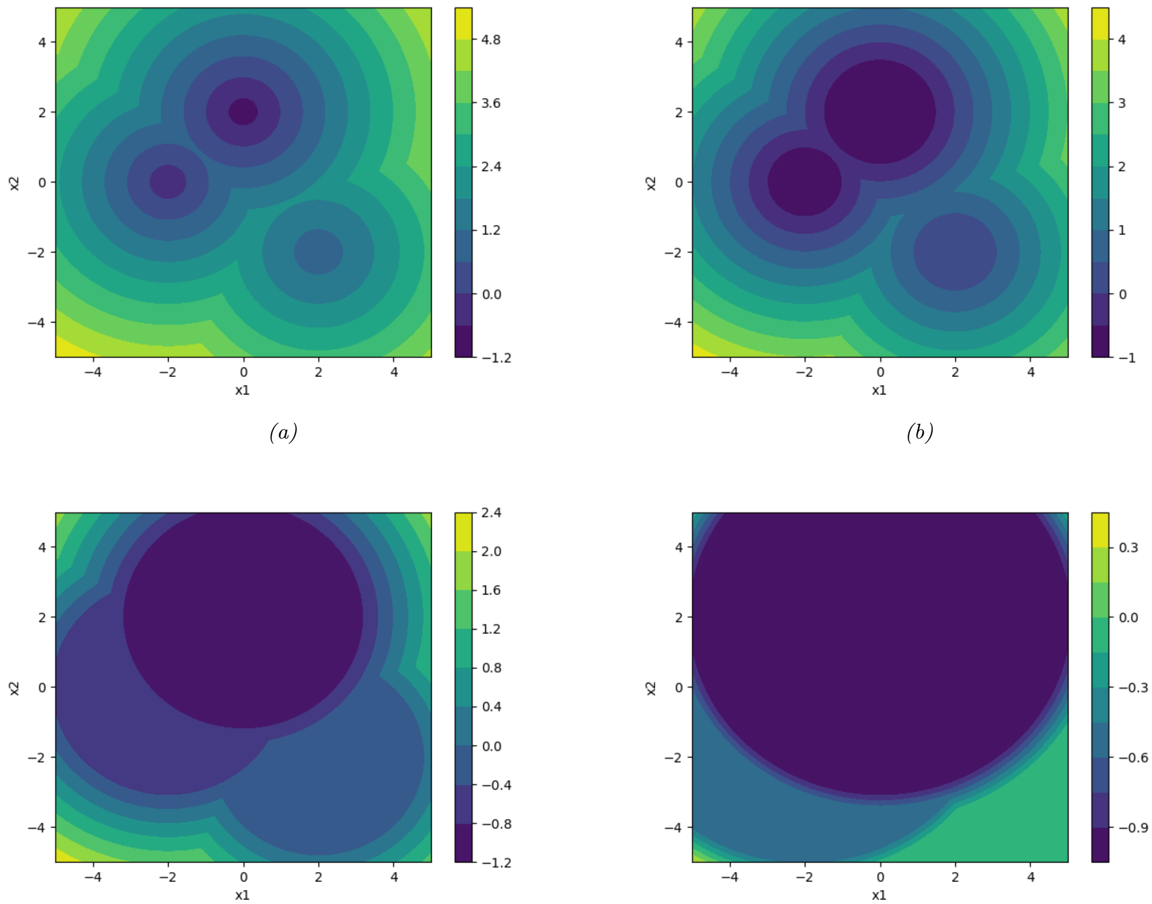
$$H(\boldsymbol{p}) = \max_{i \in \{1,\dots,m\}} \{\langle \boldsymbol{p}, \boldsymbol{v}_i \rangle - b_i\}, \tag{12}$$

for every $\boldsymbol{p} \in \mathbb{R}^n$. Hence f_2 is a concave viscosity solution to the corresponding HJ PDE (5)*.*

Proof. By assumption (H) and simply changing the notations in [28, Lem. 3.1], we have

$$H^*(\boldsymbol{v}) = \min \left\{ \sum_{i=1}^m \alpha_i b_i \colon (\alpha_1, \dots, \alpha_m) \in \Lambda_m, \sum_{i=1}^m \alpha_i \boldsymbol{v}_i = \boldsymbol{v} \right\}, \tag{13}$$

for each $\boldsymbol{v} \in \text{co}\,\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_m\} = \text{dom}\, H^*$, where Λ_m is the unit simplex defined in (2). Also, we have $H^*(\boldsymbol{v}_k) = b_k$ for each $k \in \{1, \dots, m\}$.



(a) (b) (c) (d)

FIGURE 4. A two dimensional slice of the graph of f_1 in example 3.2. In each figure, the x and y axes correspond to the variables x_1 and x_2, which are the first and second coordinates of the variable $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$. The color is given by the function value $f_1(\boldsymbol{x}, t)$. The figures (a), (b), (c) and (d) show contour lines of the solution $f_1(\boldsymbol{x}, t)$ at $t = 10^{-6}$, $t = 1$, $t = 3$ and $t = 5$, respectively.

For each $\boldsymbol{x} \in \mathbb{R}^n$, $t > 0$ and $\boldsymbol{v} \in \mathrm{co}\,\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m\}$, let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_m) \in \Lambda_m$ be the minimizer in the minimization problem in (13) evaluated at $\boldsymbol{v}$. In other words, we have

$$\sum_{i=1}^m \alpha_i = 1, \quad \sum_{i=1}^m \alpha_i \boldsymbol{v}_i = \boldsymbol{v}, \quad \sum_{i=1}^m \alpha_i b_i = H^*(\boldsymbol{v}), \qquad \text{and } \alpha_j \in [0,1] \text{ for each } j \in \{1, \ldots, m\}. \tag{14}$$

Then, by (14) and the assumption that $J = \tilde{J}$ is concave, we have

$$\begin{aligned} J(\boldsymbol{x} - t\boldsymbol{v}) + tH^*(\boldsymbol{v}) &= J\left(\sum_{i=1}^m \alpha_i \left(\boldsymbol{x} - t\boldsymbol{v}_i\right)\right) + t\sum_{i=1}^m \alpha_i b_i \geq \sum_{i=1}^m \alpha_i J\left(\boldsymbol{x} - t\boldsymbol{v}_i\right) + \sum_{i=1}^m \alpha_i t b_i \\ &= \sum_{i=1}^m \alpha_i (J\left(\boldsymbol{x} - t\boldsymbol{v}_i\right) + tb_i) \geq \min_{i \in \{1, \ldots, m\}} \left\{J\left(\boldsymbol{x} - t\boldsymbol{v}_i\right) + tb_i\right\} = f_2(\boldsymbol{x}, t). \end{aligned}$$

As a result, we conclude that

$$S_{LO}(\boldsymbol{x}, t) = \inf_{\boldsymbol{v} \in \mathrm{dom}\ H^*} \left\{J(\boldsymbol{x} - t\boldsymbol{v}) + tH^*(\boldsymbol{v})\right\} \geq f_2(\boldsymbol{x}, t).$$

On the other hand, recall that $b_k = H^*(\boldsymbol{v}_k)$ for each $k \in \{1, \dots, m\}$, hence we obtain

$$f_2(\boldsymbol{x}, t) = \min_{i\in\{1,\dots,m\}} \{J(\boldsymbol{x} - t\boldsymbol{v}_i) + tH^*(\boldsymbol{v}_i)\} \geq \inf_{\boldsymbol{v}\in\mathbb{R}^n} \{J(\boldsymbol{x} - t\boldsymbol{v}) + tH^*(\boldsymbol{v})\} = S_{LO}(\boldsymbol{x}, t).$$

Therefore, we conclude that $f_2(\boldsymbol{x}, t) = S_{LO}(\boldsymbol{x}, t)$ for each $\boldsymbol{x} \in \mathbb{R}^n$ and $t > 0$.

Note that H is a convex function, since it is the maximum of affine functions. Then, by the same proof as in [38, Sec. 10.3.4, Thm. 3], we conclude that f_2 is a viscosity solution to the corresponding HJ PDE. Moreover, since $\tilde{J}$ is concave, f_2 is the minimum of concave functions, which implies the concavity of f_2. □

Remark 3.2. *In the second architecture, if we furthermore assume that the initial condition $J = \tilde{J}$ is uniformly Lipschitz, then f_2 is the unique uniformly continuous viscosity solution to the corresponding HJ PDE. This conclusion directly follows from [9, Thm. 2.1].*

Example 3.3. *Here, we provide a one dimensional example of the function f_2. To be specific, we consider $f_2 \colon \mathbb{R} \times [0, +\infty) \to \mathbb{R}$ represented by the neural network in Fig. 2 with three neurons, i.e., we set $n = 1$ and $m = 3$. The initial value is given by $J(x) = -\frac{x^2}{2}$ for each $x \in \mathbb{R}$, and the Hamiltonian H is given by the piecewise affine function in Eq. (12) with $(v_1, b_1) = (-2, 0.5)$, $(v_2, b_2) = (0, -5)$ and $(v_3, b_3) = (2, 1)$. The functions J, H and f_2 are shown in Fig. 5, where (a) shows the initial value J, (b) shows the convex Hamiltonian H, and (c) and (d) show the solution $S = f_2$ evaluated at $t = 1$ and $t = 3$, respectively.*

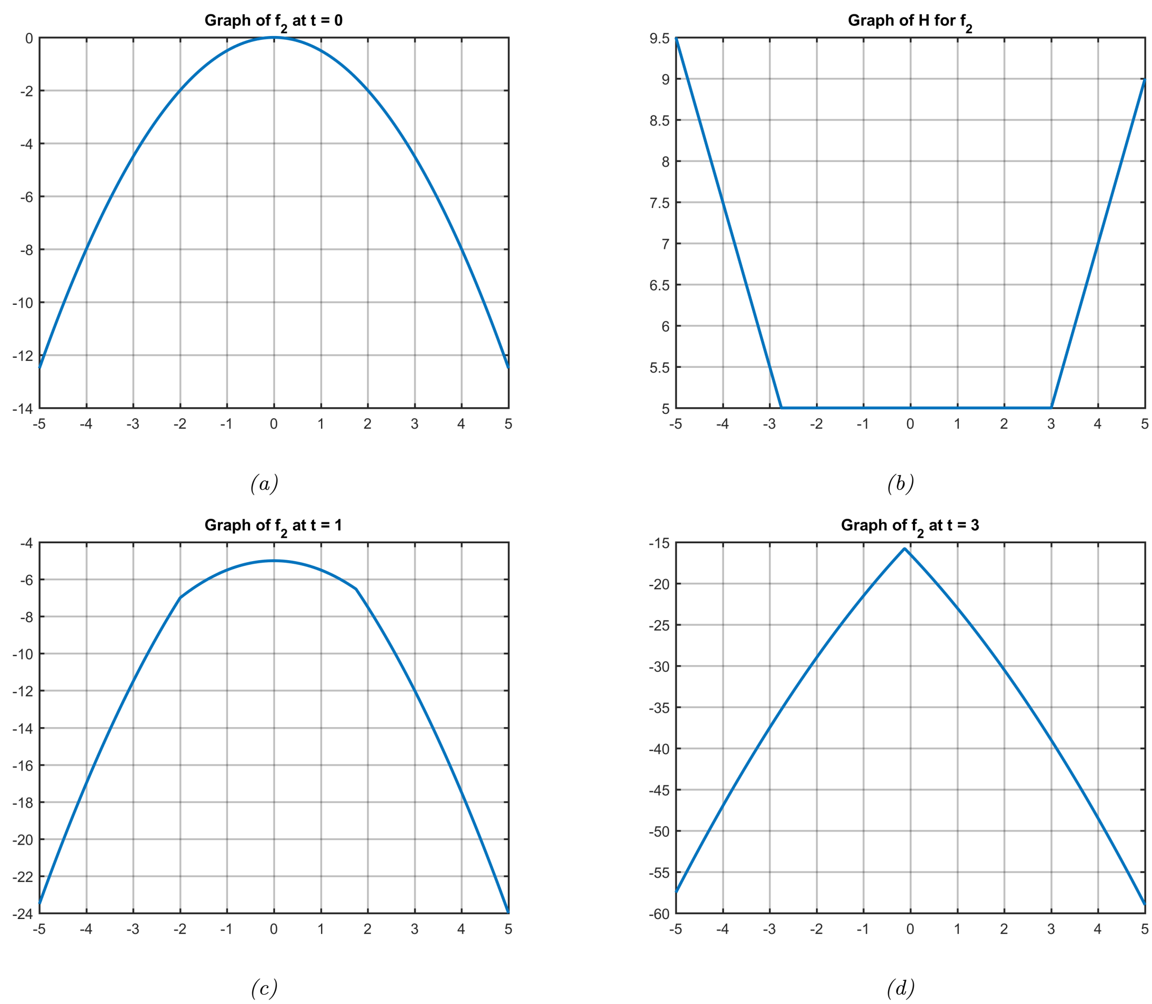


FIGURE 5. The graph of f_2 in example 3.3. The figures (a) and (b) show the initial value J and the Hamiltonian H, respectively. The figures (c) and (d) show the solution $S = f_2$ evaluated at $t = 1$ and $t = 3$, respectively.

The corresponding Tensorflow code is given as follows.

```
import numpy as np
import tensorflow as tf

n_data = 1000
dim = 1

def min_fn(x):
    return tf.math.reduce_min(x, axis=-1)

# J = -x^2/2.
def J_fn(x):
    val = -tf.multiply(x,x)/2
    return tf.squeeze(val, -1)

tf.reset_default_graph()

v_true = np.array([[-2], [0], [2]])
b_true = np.array([0.5, -5, 1])
v_param = tf.Variable(v_true, name = "v0", dtype = tf.float64)
b_param = tf.Variable(b_true, name = "b0", dtype = tf.float64)
x_placeholder = tf.placeholder(tf.float64, shape=(n_data, 1, dim))
t_placeholder = tf.placeholder(tf.float64, shape=(n_data, 1))

x_grid0 = np.arange(n_data) * (10.0 / n_data) - 5
t_grid0 = np.arange(n_data) * 0 + 3
x_grid = np.expand_dims(x_grid0, axis = -1)
x_grid = np.expand_dims(x_grid, axis = -1)
t_grid = np.expand_dims(t_grid0, axis = -1)

def construct_nn(x_in, t_in):
    t = tf.expand_dims(t_in, -1)
    val0 = tf.subtract(x_in, tf.multiply(t, v_param))
    val1 = J_fn(val0)
    val2 = tf.add(val1, tf.multiply(t_in, b_param))
    y_ = min_fn(val2)
    return y_

y_nn = construct_nn(x_placeholder, t_placeholder)
sess = tf.Session()
sess.run(tf.global_variables_initializer())
y_val = sess.run(y_nn, {x_placeholder: x_grid, t_placeholder: t_grid})
sess.close()
```

Example 3.4. *Here, we present a high dimensional example. We choose the dimension to be* $n = 10$*. We consider the solution* $f_2 : \mathbb{R}^{10} \times [0, +\infty) \to \mathbb{R}$ *represented by the neural network in Fig. 2 with three neurons, i.e., we set* $m = 3$*. Similar to the one dimensional case, the activation function* $\tilde{J}$ *is chosen to be* $\tilde{J}(\boldsymbol{x}) = -\frac{\|\boldsymbol{x}\|_2^2}{2}$ *for every* $\boldsymbol{x} \in \mathbb{R}^{10}$*. Hence, by Thm. 3.2, the initial data in the corresponding HJ PDE is given by* $J(\boldsymbol{x}) = \tilde{J}(\boldsymbol{x}) = -\frac{\|\boldsymbol{x}\|_2^2}{2}$*. The parameters are chosen to be* $\boldsymbol{v}_1 = (-2, 0, 0, 0, \dots, 0)$*,* $\boldsymbol{v}_2 = (2, -2, -1, 0, \dots, 0)$*,* $\boldsymbol{v}_3 = (0, 2, 0, 0, \dots, 0)$*,* $b_1 = 0.5$*,* $b_2 = -5$ *and* $b_3 = 1$*. Then the Hamiltonian is the corresponding convex piecewise affine function defined in* (12)*.*

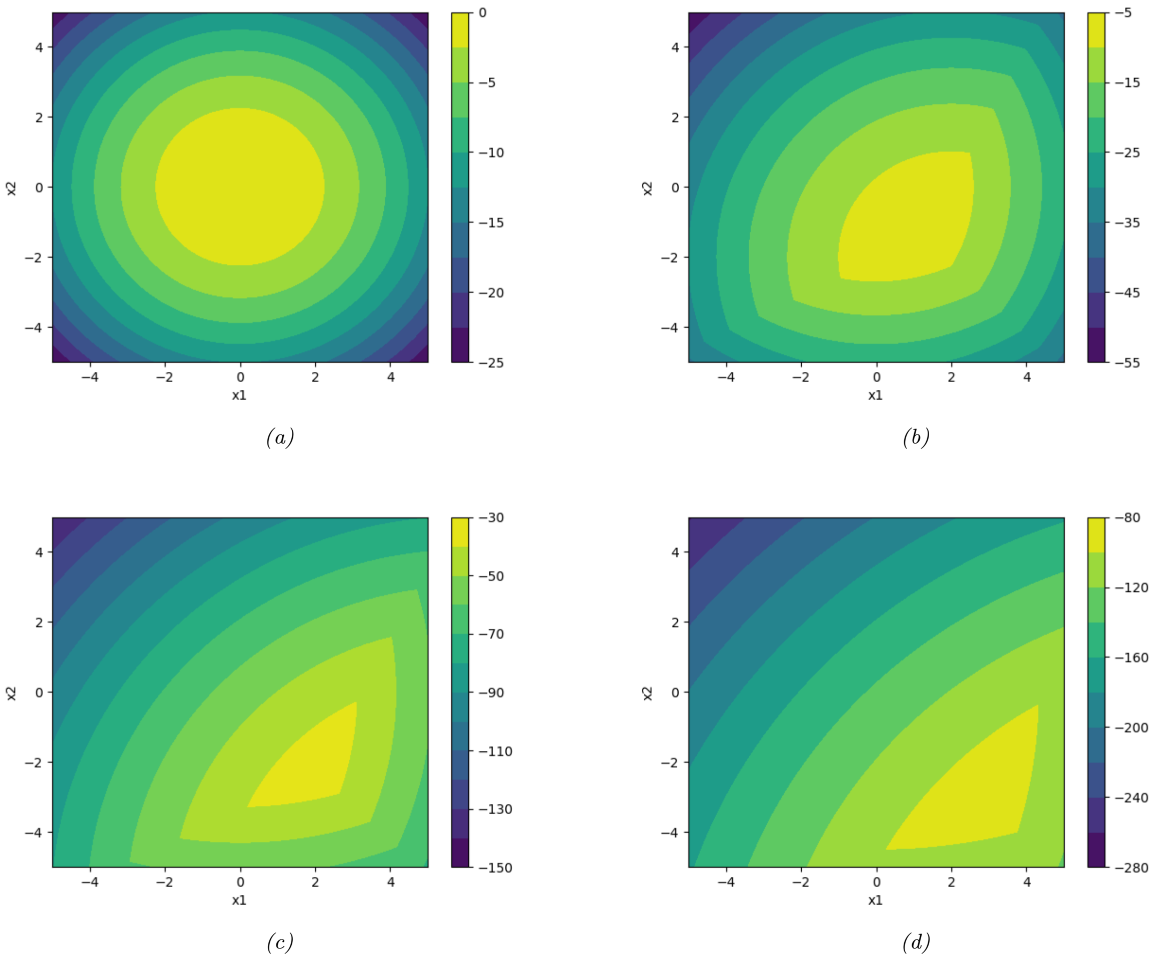


FIGURE 6. A two dimensional slice of the graph of f_2 in example 3.4. In each figure, the x and y axes correspond to the variables x_1 and x_2, which are the first and second coordinates of the variable $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$. The color is given by the function value $f_2(\boldsymbol{x}, t)$. The figures (a), (b), (c) and (d) show contour lines of the solution $f_2(\boldsymbol{x}, t)$ at $t = 0$, $t = 1$, $t = 3$ and $t = 5$, respectively.

The solution f_2 is shown in Fig. 6. We fix $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$ and compute $f_2(\boldsymbol{x}, t)$ for $t = 0$, 1, 3 and 5. In each figure, the color is given by the function value $f_2(\boldsymbol{x}, t)$ and the x and y axes represent the variables x_1 and x_2, respectively. The solutions at $t = 0$, $t = 1$, $t = 3$ and $t = 5$ are shown in (a), (b), (c) and (d), respectively.

Example 3.5. *In this example, we consider two HJ PDEs defined for $\boldsymbol{x} \in \mathbb{R}^5$, i.e., the dimension is $n = 5$. The initial data J is given by $J(\boldsymbol{x}) = -\frac{\|\boldsymbol{x}\|_2^2}{2}$ for each $\boldsymbol{x} \in \mathbb{R}^5$ and the Hamiltonian H is the l^1-norm or the l^∞-norm. The corresponding solutions f_2 are shown in Figs. 7 and 8. Similarly as in example 3.4, we consider the variable $\boldsymbol{x} = (x_1, x_2, 0, 0, 0)$ and show the 2-dimensional slice in each figure. The solutions at $t = 0$, $t = 1$, $t = 3$ and $t = 5$ are shown in (a), (b), (c) and (d), respectively, in each figure.*

When H is the l^1-norm, i.e., $H(\boldsymbol{p}) = \|\boldsymbol{p}\|_1$ for each $\boldsymbol{p} \in \mathbb{R}^5$, the Hamiltonian H can be written in the form of Eq. (12) with $m = 2^n$, $b_i = 0$ for each $i \in \{1, \ldots, m\}$ and

$$\{\boldsymbol{v}_i\}_{i=1}^m = \{(w_1, w_2, \ldots, w_n) \in \mathbb{R}^n : w_j \in \{\pm 1\} \, \forall j \in \{1, \ldots, n\}\}.$$

The corresponding function f_2 is shown in Fig. 7.

When H is the l^∞-norm, i.e., $H(\boldsymbol{p}) = \|\boldsymbol{p}\|_\infty$ for each $\boldsymbol{p} \in \mathbb{R}^5$, the Hamiltonian H can be written in the form of Eq. (12) *with $m = 2n$, $b_i = 0$ for each $i \in \{1, \dots, m\}$ and*

$$\{\boldsymbol{v}_i\}_{i=1}^m = \{\pm \boldsymbol{e}_j\}_{j=1}^n,$$

where $\boldsymbol{e}_j$ is the $j-$th coordinate basis vector in $\mathbb{R}^n$. The corresponding function f_2 is shown in Fig. 8.

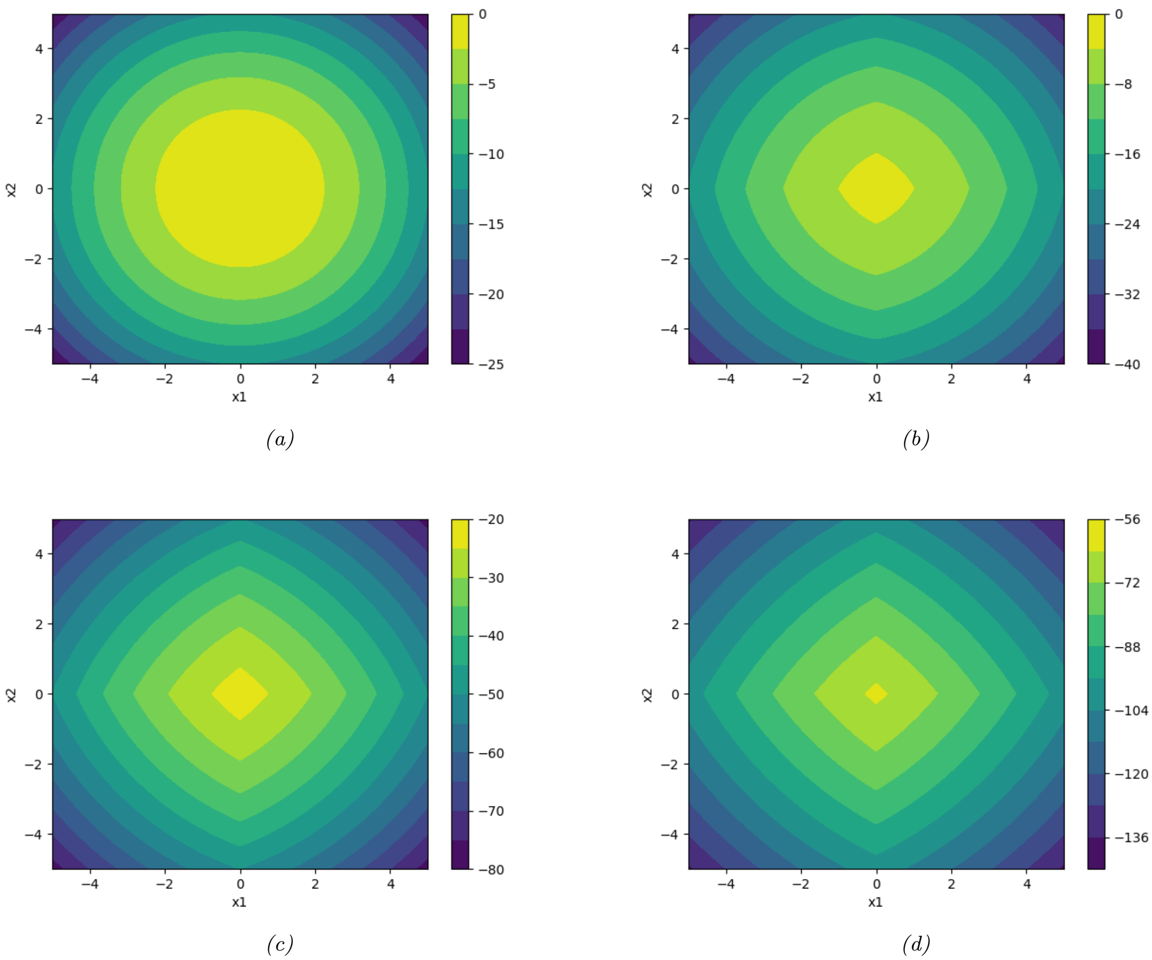


FIGURE 7. A two dimensional slice of the graph of f_2 in example 3.5. The initial data J is given by $J(\boldsymbol{x}) = -\frac{\|\boldsymbol{x}\|_2^2}{2}$ and the Hamiltonian H is the l^1 norm. In each figure, the x and y axes correspond to the variables x_1 and x_2, which are the first and second coordinates of the variable $\boldsymbol{x} = (x_1, x_2, 0, \dots, 0)$. The color is given by the function value $f_2(\boldsymbol{x}, t)$. The figures (a), (b), (c) and (d) show contour lines of the solution $f_2(\boldsymbol{x}, t)$ at $t = 0$, $t = 1$, $t = 3$ and $t = 5$, respectively.

4. CONCLUSION

In this paper, we investigated two neural network architectures shown in Figs. 1 and 2, and proved that these two architectures represent viscosity solutions to two sets of HJ PDEs whose convex Hamiltonian H and initial data J satisfy certain assumptions in Thms. 3.1 and 3.2, respectively. This connection provides a possible interpretation for some neural network architectures. Our results suggest that efficient dedicated hardware implementation for neural networks can be leveraged to evaluate viscosity solutions of certain HJ PDEs.

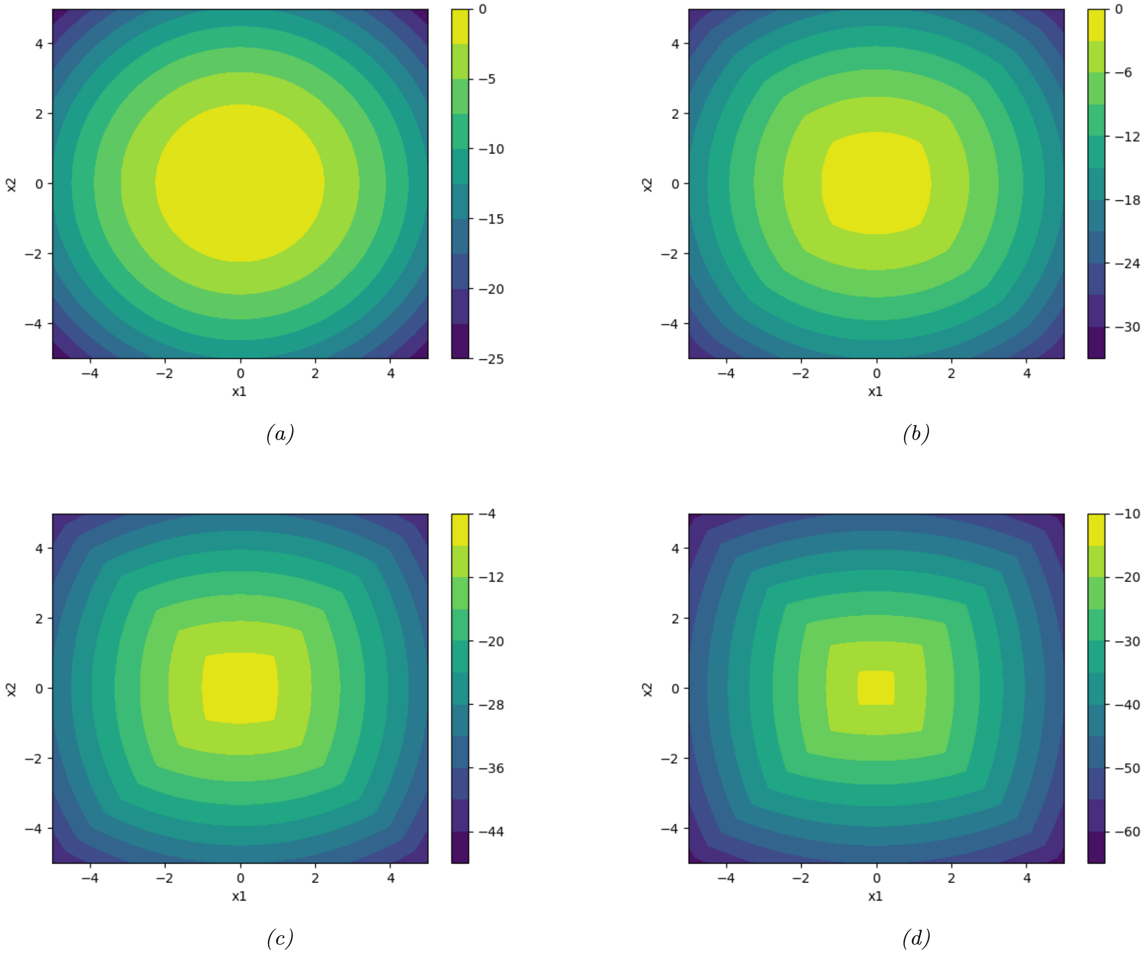


FIGURE 8. A two dimensional slice of the graph of f_2 in example 3.5. The initial data J is given by $J(\boldsymbol{x}) = -\frac{\|\boldsymbol{x}\|_2^2}{2}$ and the Hamiltonian H is the l^∞ norm. In each figure, the x and y axes correspond to the variables x_1 and x_2, which are the first and second coordinates of the variable $\boldsymbol{x} = (x_1, x_2, 0, \ldots, 0)$. The color is given by the function value $f_2(\boldsymbol{x}, t)$. The figures (a), (b), (c) and (d) show contour lines of the solution $f_2(\boldsymbol{x}, t)$ at $t = 0$, $t = 1$, $t = 3$ and $t = 5$, respectively.

Acknowledgments

This research is supported by NSF DMS 1820821.

References

[1] M. Akian, R. Bapat, and S. Gaubert, *Max-plus algebra*, Handbook of linear algebra, 39 (2006).

[2] M. Akian, S. Gaubert, and A. Lakhoua, *The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis*, SIAM Journal on Control and Optimization, 47 (2008), pp. 817–848.

[3] A. Alla, M. Falcone, and L. Saluzzi, *An efficient DP algorithm on a tree-structure for finite horizon optimal control problems*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2384–A2406.

[4] A. Alla, M. Falcone, and S. Volkwein, *Error analysis for POD approximations of infinite horizon problems via the dynamic programming approach*, SIAM Journal on Control and Optimization, 55 (2017), pp. 3091–3115.

[5] V. I. Arnol'd, *Mathematical methods of classical mechanics*, vol. 60 of Graduate Texts in Mathematics, Springer-Verlag, New York, 1989. Translated from the 1974 Russian original by K. Vogtmann and A. Weinstein, Corrected reprint of the second (1989) edition.

[6] A. Bachouch, C. Huré, N. Langrené, and H. Pham, *Deep neural networks algorithms for stochastic control problems on finite horizon: numerical applications*, arXiv preprint arXiv:1812.05916, (2018).

[7] K. Banerjee, E. Georganas, D. Kalamkar, B. Ziv, E. Segal, C. Anderson, and A. Heinecke, *Optimizing deep learning rnn topologies on intel architecture*, Supercomputing Frontiers and Innovations, 6 (2019).

[8] M. Bardi and I. Capuzzo-Dolcetta, *Optimal control and viscosity solutions of Hamilton-Jacobi-Bellman equations*, Systems & Control: Foundations & Applications, Birkhäuser Boston, Inc., Boston, MA, 1997. With appendices by Maurizio Falcone and Pierpaolo Soravia.

[9] M. Bardi and L. Evans, *On Hopf's formulas for solutions of Hamilton-Jacobi equations*, Nonlinear Analysis: Theory, Methods & Applications, 8 (1984), pp. 1373 – 1381.

[10] G. Barles, *Solutions de viscosité des équations de Hamilton-Jacobi*, Mathématiques et Applications, Springer-Verlag Berlin Heidelberg, 1994.

[11] E. Barron, L. Evans, and R. Jensen, *Viscosity solutions of Isaacs' equations and differential games with Lipschitz controls*, Journal of Differential Equations, 53 (1984), pp. 213 – 233.

[12] C. Beck, S. Becker, P. Cheridito, A. Jentzen, and A. Neufeld, *Deep splitting method for parabolic PDEs*, arXiv preprint arXiv:1907.03452, (2019).

[13] C. Beck, S. Becker, P. Grohs, N. Jaafari, and A. Jentzen, *Solving stochastic differential equations and Kolmogorov equations by means of deep learning*, arXiv preprint arXiv:1806.00421, (2018).

[14] C. Beck, E. Weinan, and A. Jentzen, *Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations*, Journal of Nonlinear Science, 29 (2019), pp. 1563–1619.

[15] J. Berg and K. Nystrm, *A unified deep artificial neural network approach to partial differential equations in complex geometries*, Neurocomputing, 317 (2018), pp. 28 – 41.

[16] D. P. Bertsekas, *Reinforcement learning and optimal control*, Athena Scientific, Belmont, Massachusetts, (2019).

[17] O. Bokanowski, J. Garcke, M. Griebel, and I. Klompmaker, *An adaptive sparse grid semi-Lagrangian scheme for first order Hamilton-Jacobi Bellman equations*, Journal of Scientific Computing, 55 (2013), pp. 575–605.

[18] R. Buckdahn, P. Cardaliaguet, and M. Quincampoix, *Some recent aspects of differential game theory*, Dynamic Games and Applications, 1 (2011), pp. 74–114.

[19] C. Carathéodory, *Calculus of variations and partial differential equations of the first order. Part I: Partial differential equations of the first order*, Translated by Robert B. Dean and Julius J. Brandstatter, Holden-Day, Inc., San Francisco-London-Amsterdam, 1965.

[20] ——, *Calculus of variations and partial differential equations of the first order. Part II: Calculus of variations*, Translated from the German by Robert B. Dean, Julius J. Brandstatter, translating editor, Holden-Day, Inc., San Francisco-London-Amsterdam, 1967.

[21] L. Á. L. Cárdenas and F. Gibou, *A deep learning approach for the computation of curvature in the level-set method*, arXiv preprint arXiv:2002.02804, (2020).

[22] Q. Chan-Wai-Nam, J. Mikael, and X. Warin, *Machine learning for semi linear PDEs*, Journal of Scientific Computing, 79 (2019), pp. 1667–1712.

[23] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. de Wilde, H.-C. R. Euler, H. Broersma, P. A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel, *Classification with a disordered dopant-atom network in silicon*, Nature, 577 (2020), pp. 341–345.

[24] T. Cheng and F. L. Lewis, *Fixed-final time constrained optimal control of nonlinear systems using neural network HJB approach*, in Proceedings of the 45th IEEE Conference on Decision and Control, Dec 2006, pp. 3016–3021.

[25] R. Courant and D. Hilbert, *Methods of mathematical physics. Vol. II*, Wiley Classics Library, John Wiley & Sons, Inc., New York, 1989. Partial differential equations, Reprint of the 1962 original, A Wiley-Interscience Publication.

[26] M. G. Crandall, H. Ishii, and P.-L. Lions, *User's guide to viscosity solutions of second order partial differential equations*, Bulletin of the American mathematical society, 27 (1992), pp. 1–67.

[27] J. Darbon, *On convex finite-dimensional variational methods in imaging sciences and Hamilton–Jacobi equations*, SIAM Journal on Imaging Sciences, 8 (2015), pp. 2268–2293.

[28] J. Darbon, G. P. Langlois, and T. Meng, *Overcoming the curse of dimensionality for some Hamilton–Jacobi partial differential equations via neural network architectures*, arXiv preprint arXiv:1910.09045, (2019).

[29] J. Darbon and T. Meng, *On decomposition models in imaging sciences and multi-time Hamilton-Jacobi partial differential equations*, arXiv preprint arXiv:1906.09502, (2019).

[30] J. Darbon and S. Osher, *Algorithms for overcoming the curse of dimensionality for certain Hamilton–Jacobi equations arising in control theory and elsewhere*, Research in the Mathematical Sciences, 3 (2016), p. 19.

[31] M. W. M. G. Dissanayake and N. Phan-Thien, *Neural-network-based approximations for solving partial differential equations*, Communications in Numerical Methods in Engineering, 10 (1994), pp. 195–201.

[32] B. Djeridane and J. Lygeros, *Neural approximation of PDE solutions: An application to reachability computations*, in Proceedings of the 45th IEEE Conference on Decision and Control, Dec 2006, pp. 3034–3039.

[33] T. Dockhorn, *A discussion on solving partial differential equations using neural networks*, arXiv preprint arXiv:1904.07200, (2019).

[34] S. Dolgov, D. Kalise, and K. Kunisch, *A tensor decomposition approach for high-dimensional Hamilton-Jacobi-Bellman equations*, arXiv preprint arXiv:1908.01533, (2019).

[35] P. M. Dower, W. M. McEneaney, and H. Zhang, *Max-plus fundamental solution semigroups for optimal control problems*, in 2015 Proceedings of the Conference on Control and its Applications, SIAM, 2015, pp. 368–375.

[36] W. E, J. Han, and A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations*, Communications in Mathematics and Statistics, 5 (2017), pp. 349–380.

[37] R. J. Elliott, *Viscosity solutions and optimal control*, vol. 165 of Pitman Research Notes in Mathematics Series, Longman Scientific & Technical, Harlow; John Wiley & Sons, Inc., New York, 1987.

[38] L. C. Evans, *Partial differential equations*, vol. 19 of Graduate Studies in Mathematics, American Mathematical Society, Providence, RI, second ed., 2010.

[39] L. C. Evans and P. E. Souganidis, *Differential games and representation formulas for solutions of Hamilton-Jacobi-Isaacs equations*, Indiana University Mathematics Journal, 33 (1984), pp. 773–797.

[40] C. Farabet, Y. LeCun, K. Kavukcuoglu, E. Culurciello, B. Martini, P. Akselrod, and S. Talay, *Large-scale fpga-based convolutional networks*, in Scaling up Machine Learning: Parallel and Distributed Approaches, R. Bekkerman, M. Bilenko, and J. Langford, eds., Cambridge University Press, 2011.

[41] C. Farabet, C. poulet, J. Han, and Y. LeCun, *Cnp: An fpga-based processor for convolutional networks*, in International Conference on Field Programmable Logic and Applications, Prague, September 2009, IEEE.

[42] C. Farabet, C. Poulet, and Y. LeCun, *An fpga-based stream processor for embedded real-time vision with convolutional networks*, in 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, Los Alamitos, CA, USA, oct 2009, IEEE Computer Society, pp. 878–885.

[43] A. B. Farimani, J. Gomes, and V. S. Pande, *Deep Learning the Physics of Transport Phenomena*, arXiv e-prints, (2017).

[44] W. Fleming and W. McEneaney, *A max-plus-based algorithm for a Hamilton–Jacobi–Bellman equation of nonlinear filtering*, SIAM Journal on Control and Optimization, 38 (2000), pp. 683–710.

[45] W. H. Fleming and R. W. Rishel, *Deterministic and stochastic optimal control*, Bulletin of the American Mathematical Society, 82 (1976), pp. 869–870.

[46] W. H. Fleming and H. M. Soner, *Controlled Markov processes and viscosity solutions*, vol. 25, Springer Science & Business Media, 2006.

[47] M. Fujii, A. Takahashi, and M. Takahashi, *Asymptotic expansion as prior knowledge in deep learning method for high dimensional BSDEs*, Asia-Pacific Financial Markets, 26 (2019), pp. 391–408.
[48] J. Garcke and A. Kröner, *Suboptimal feedback control of PDEs by solving HJB equations on adaptive sparse grids*, Journal of Scientific Computing, 70 (2017), pp. 1–28.
[49] S. Gaubert, W. McEneaney, and Z. Qu, *Curse of dimensionality reduction in max-plus based approximation methods: Theoretical estimates and improved pruning algorithms*, in 2011 50th IEEE Conference on Decision and Control and European Control Conference, IEEE, 2011, pp. 1054–1061.
[50] P. Grohs, A. Jentzen, and D. Salimova, *Deep neural network approximations for Monte Carlo algorithms*, arXiv preprint arXiv:1908.10828, (2019).
[51] J. Han, A. Jentzen, and W. E, *Solving high-dimensional partial differential equations using deep learning*, Proceedings of the National Academy of Sciences, 115 (2018), pp. 8505–8510.
[52] J. Han, L. Zhang, and E. Weinan, *Solving many-electron Schrödinger equation using deep neural networks*, Journal of Computational Physics, (2019), p. 108929.
[53] J.-B. Hiriart-Urruty and C. Lemaréchal, *Convex analysis and minimization algorithms I: Fundamentals*, vol. 305, Springer science & business media, 1993.
[54] ——, *Convex analysis and minimization algorithms II: Advanced Theory and Bundle Methods*, vol. 306, Springer science & business media, 1993.
[55] C. Hirjibehedin, *Evolution of circuits for machine learning*, Nature, 577 (2020), pp. 320–321.
[56] E. Hopf, *Generalized solutions of non-linear equations of first order*, J. Math. Mech., 14 (1965), pp. 951–973.
[57] M. B. Horowitz, A. Damle, and J. W. Burdick, *Linear Hamilton Jacobi Bellman equations in high dimensions*, in 53rd IEEE Conference on Decision and Control, IEEE, 2014, pp. 5880–5887.
[58] J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon, *Learning neural PDE solvers with convergence guarantees*, in International Conference on Learning Representations, 2019.
[59] C. Huré, H. Pham, A. Bachouch, and N. Langrené, *Deep neural networks algorithms for stochastic control problems on finite horizon, part I: convergence analysis*, arXiv preprint arXiv:1812.04300, (2018).
[60] C. Huré, H. Pham, and X. Warin, *Some machine learning schemes for high-dimensional nonlinear PDEs*, arXiv preprint arXiv:1902.01599, (2019).
[61] H. Ishii, *Representation of solutions of Hamilton-Jacobi equations*, Nonlinear Analysis: Theory, Methods & Applications, 12 (1988), pp. 121 – 146.
[62] F. Jiang, G. Chou, M. Chen, and C. J. Tomlin, *Using neural networks to compute approximate and guaranteed feasible Hamilton-Jacobi-Bellman PDE solutions*, arXiv preprint arXiv:1611.03158, (2016).
[63] L. Jianyu, L. Siwei, Q. Yingjian, and H. Yaping, *Numerical solution of elliptic partial differential equation using radial basis function neural networks*, Neural Networks, 16 (2003), pp. 729–734.
[64] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, and et al., *In-datacenter performance analysis of a tensor processing unit*, in Proceedings of the 44th Annual International Symposium on Computer Architecture, ISCA 17, New York, NY, USA, 2017, Association for Computing Machinery, p. 112.
[65] D. Kalise, S. Kundu, and K. Kunisch, *Robust feedback control of nonlinear PDEs by numerical approximation of high-dimensional Hamilton-Jacobi-Isaacs equations*, arXiv preprint arXiv:1905.06276, (2019).
[66] D. Kalise and K. Kunisch, *Polynomial approximation of high-dimensional Hamilton–Jacobi–Bellman equations and applications to feedback control of semilinear parabolic PDEs*, SIAM Journal on Scientific Computing, 40 (2018), pp. A629–A652.
[67] W. Kang and L. C. Wilcox, *Mitigating the curse of dimensionality: sparse grid characteristics method for optimal feedback control and HJB equations*, Computational Optimization and Applications, 68 (2017), pp. 289–315.
[68] Y. Khoo, J. Lu, and L. Ying, *Solving parametric PDE problems with artificial neural networks*, arXiv preprint arXiv:1707.03351, (2017).
[69] ——, *Solving for high-dimensional committor functions using artificial neural networks*, Research in the Mathematical Sciences, 6 (2019), p. 1.

[70] A. Kundu, S. Srinivasan, E. C. Qin, D. Kalamkar, N. K. Mellempudi, D. Das, K. Banerjee, B. Kaul, and P. Dubey, *K-tanh: Hardware efficient activations for deep learning*, arXiv preprint arXiv:1909.07729, (2019).

[71] K. Kunisch, S. Volkwein, and L. Xie, *HJB-POD-based feedback design for the optimal control of evolution problems*, SIAM Journal on Applied Dynamical Systems, 3 (2004), pp. 701–722.

[72] I. E. Lagaris, A. Likas, and D. I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Transactions on Neural Networks, 9 (1998), pp. 987–1000.

[73] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, *Neural-network methods for boundary value problems with irregular boundaries*, IEEE Transactions on Neural Networks, 11 (2000), pp. 1041–1049.

[74] P. Lambrianides, Q. Gong, and D. Venturi, *A new scalable algorithm for computational optimal control under uncertainty*, arXiv preprint arXiv:1909.07960, (2019).

[75] L. Landau and E. Lifschic, *Course of theoretical physics. vol. 1: Mechanics*, Oxford, 1978.

[76] Y. LeCun, *1.1 deep learning hardware: Past, present, and future*, in 2019 IEEE International Solid-State Circuits Conference - (ISSCC), Feb 2019, pp. 12–19.

[77] H. Lee and I. S. Kang, *Neural algorithm for solving differential equations*, Journal of Computational Physics, 91 (1990), pp. 110–131.

[78] Z. Long, Y. Lu, and B. Dong, *PDE-net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network*, Journal of Computational Physics, 399 (2019), p. 108925.

[79] Z. Long, Y. Lu, X. Ma, and B. Dong, *PDE-net: Learning PDEs from data*, arXiv preprint arXiv:1710.09668, (2017).

[80] K. O. Lye, S. Mishra, and D. Ray, *Deep learning observables in computational fluid dynamics*, arXiv preprint arXiv:1903.03040, (2019).

[81] W. McEneaney, *Max-plus methods for nonlinear control and estimation*, Springer Science & Business Media, 2006.

[82] ——, *A curse-of-dimensionality-free numerical method for solution of certain HJB PDEs*, SIAM Journal on Control and Optimization, 46 (2007), pp. 1239–1276.

[83] W. M. McEneaney, A. Deshpande, and S. Gaubert, *Curse-of-complexity attenuation in the curse-of-dimensionality-free method for HJB PDEs*, in 2008 American Control Conference, IEEE, 2008, pp. 4684–4690.

[84] W. M. McEneaney and L. J. Kluberg, *Convergence rate for a curse-of-dimensionality-free method for a class of HJB PDEs*, SIAM Journal on Control and Optimization, 48 (2009), pp. 3052–3079.

[85] K. S. McFall and J. R. Mahan, *Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions*, IEEE Transactions on Neural Networks, 20 (2009), pp. 1221–1233.

[86] A. Meade and A. Fernandez, *The numerical solution of linear ordinary differential equations by feedforward neural networks*, Mathematical and Computer Modelling, 19 (1994), pp. 1 – 25.

[87] X. Meng and G. E. Karniadakis, *A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse PDE problems*, arXiv preprint arXiv:1903.00104, (2019).

[88] X. Meng, Z. Li, D. Zhang, and G. E. Karniadakis, *PPINN: Parareal physics-informed neural network for time-dependent PDEs*, arXiv preprint arXiv:1909.10145, (2019).

[89] K. N. Niarchos and J. Lygeros, *A neural approximation to continuous time reachability computations*, in Proceedings of the 45th IEEE Conference on Decision and Control, Dec 2006, pp. 6313–6318.

[90] G. Pang, L. Lu, and G. E. Karniadakis, *fPINNs: Fractional physics-informed neural networks*, SIAM Journal on Scientific Computing, 41 (2019), pp. A2603–A2626.

[91] H. Pham, H. Pham, and X. Warin, *Neural networks-based backward scheme for fully nonlinear PDEs*, arXiv preprint arXiv:1908.00412, (2019).

[92] M. Raissi, *Deep hidden physics models: Deep learning of nonlinear partial differential equations*, The Journal of Machine Learning Research, 19 (2018), pp. 932–955.

[93] ——, *Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations*, arXiv preprint arXiv:1804.07010, (2018).

[94] M. Raissi, P. Perdikaris, and G. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational Physics, 378 (2019), pp. 686 – 707.

[95] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations*, arXiv preprint arXiv:1711.10561, (2017).
[96] ——, *Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations*, arXiv preprint arXiv:1711.10566, (2017).
[97] C. Reisinger and Y. Zhang, *Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems*, arXiv preprint arXiv:1903.06652, (2019).
[98] R. T. Rockafellar, *Convex analysis*, Princeton university press, 1970.
[99] V. R. Royo and C. Tomlin, *Recursive regression with neural networks: Approximating the HJI PDE solution*, arXiv preprint arXiv:1611.02739, (2016).
[100] K. Rudd, G. D. Muro, and S. Ferrari, *A constrained backpropagation approach for the adaptive solution of partial differential equations*, IEEE Transactions on Neural Networks and Learning Systems, 25 (2014), pp. 571–584.
[101] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, Journal of Computational Physics, 375 (2018), pp. 1339 – 1364.
[102] W. Tang, T. Shan, X. Dang, M. Li, F. Yang, S. Xu, and J. Wu, *Study on a Poisson's equation solver based on deep learning technique*, in 2017 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS), Dec 2017, pp. 1–3.
[103] Y. Tassa and T. Erez, *Least squares solutions of the HJB equation with neural network value-function approximators*, IEEE Transactions on Neural Networks, 18 (2007), pp. 1031–1041.
[104] E. Todorov, *Efficient computation of optimal actions*, Proceedings of the national academy of sciences, 106 (2009), pp. 11478–11483.
[105] T. Uchiyama and N. Sonehara, *Solving inverse problems in nonlinear PDEs by recurrent neural networks*, in IEEE International Conference on Neural Networks, IEEE, 1993, pp. 99–102.
[106] B. P. van Milligen, V. Tribaldos, and J. A. Jiménez, *Neural network differential equation and plasma equilibrium solver*, Phys. Rev. Lett., 75 (1995), pp. 3594–3597.
[107] E. Weinan and B. Yu, *The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems*, Communications in Mathematics and Statistics, 6 (2018), pp. 1–12.
[108] N. Yadav, A. Yadav, and M. Kumar, *An introduction to neural network methods for differential equations*, SpringerBriefs in Applied Sciences and Technology, Springer, Dordrecht, 2015.
[109] L. Yang, D. Zhang, and G. E. Karniadakis, *Physics-informed generative adversarial networks for stochastic differential equations*, arXiv preprint arXiv:1811.02033, (2018).
[110] Y. Yang and P. Perdikaris, *Adversarial uncertainty quantification in physics-informed neural networks*, Journal of Computational Physics, 394 (2019), pp. 136–152.
[111] I. Yegorov and P. M. Dower, *Perspectives on characteristics based curse-of-dimensionality-free numerical approaches for solving Hamilton–Jacobi equations*, Applied Mathematics & Optimization, (2017), pp. 1–49.
[112] D. Zhang, L. Guo, and G. E. Karniadakis, *Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks*, arXiv preprint arXiv:1905.01205, (2019).
[113] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis, *Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems*, Journal of Computational Physics, 397 (2019), p. 108850.