COMMUN. MATH. SCI.
Vol. 21, No. 6, pp. 1679–1705

SOBOLEV TRAINING FOR PHYSICS-INFORMED NEURAL NETWORKS*

HWIJAE SON†, JIN WOO JANG‡, WOO JIN HAN§, AND HYUNG JU HWANG¶

Abstract. Physics-Informed Neural Networks (PINNs) are promising applications of deep learning. The smooth architecture of a fully connected neural network is appropriate for finding the solutions of PDEs; the corresponding loss function can also be intuitively designed and guarantees convergence for various kinds of PDEs. However, the high computational cost required to train neural networks has been considered as a weakness of this approach. This paper proposes Sobolev-PINNs, a novel loss function for the training of PINNs, making the training substantially efficient. Inspired by the recent studies that incorporate derivative information for the training of neural networks, we develop a loss function that guides a neural network to reduce the error in the corresponding Sobolev space. Surprisingly, a simple modification of the loss function can make the training process similar to *Sobolev Training* although PINNs are not fully supervised learning tasks. We provide several theoretical justifications that the proposed loss functions upper bound the error in the corresponding Sobolev spaces for the viscous Burgers equation and the kinetic Fokker–Planck equation. We also present several simulation results, which show that compared with the traditional L^2 loss function, the proposed loss function guides the neural network to a significantly faster convergence. Moreover, we provide empirical evidence that shows that the proposed loss function, together with the iterative sampling techniques, performs better in solving high-dimensional PDEs.

Keywords. Physics-Informed Neural Networks; Sobolev Training; Partial Differential Equations; Neural Networks.

AMS subject classifications. 68T07; 65M99; 35Q84.

1. Introduction

Deep learning has achieved remarkable success in many scientific fields, including computer vision and natural language processing. In addition to engineering, deep learning has been successfully applied to the field of scientific computing. Particularly, the use of neural networks for the numerical integration of Partial Differential Equations (PDEs) has emerged as a new important application of deep learning.

Being a universal approximator [7, 12, 23], a neural network can approximate solutions of complex PDEs. To find the neural network solution of a PDE, a neural network is trained on a domain wherein the PDE is defined. Training a neural network comprises the following: Feeding the input data through forward pass and minimizing a predefined loss function with respect to the network parameters through backward pass. In the traditional supervised learning setting, the loss function is designed to guide the neural network to generate the same output as the target data for the given input data. However, while solving PDEs using neural networks, the target values that correspond to the analytic solution are not available. One possible way to guide the neural network to produce the same output as the solution of the PDE is to penalize the neural network to satisfy the PDE itself. Early approaches, for instance, [21, 22], proposed to train a

*Received: May 12, 2022; Accepted (in revised form): December 15, 2022. Communicated by Siddhartha Mishra.

†Department of Artificial Intelligence Software, Hanbat National University, Daejeon, Republic of Korea (hjson@hanbat.ac.kr).

‡Department of Mathematics, Pohang University of Science and Technology, Pohang, Republic of Korea (jangjw@postech.ac.kr).

§Department of Mathematics, Pohang University of Science and Technology, Pohang, Republic of Korea (wjhan@postech.ac.kr).

¶Department of Mathematics, Pohang University of Science and Technology, Pohang, Republic of Korea (hjhwang@postech.ac.kr).

trial function that exactly satisfies the boundary conditions on a set of predefined grid points. Later, [4, 14, 17, 28, 31] reported for various kinds of problems involving PDEs.

Unlike the traditional mesh-based schemes including the Finite Difference Method (FDM) and the Finite Element Method (FEM), neural networks are inherently mesh-free function-approximators. Advantageously, as mesh-free approximators, neural networks can be applied to solve high-dimensional PDEs [31] and approximate the solutions of PDEs on complex geometries [4]. Recently, [14] showed that, in the continuous loss setting, the neural networks could approximate the solutions of kinetic Fokker–Planck equations under not only various kinds of kinetic boundary conditions but also several irregular initial conditions. Moreover, they showed that the neural networks automatically approximate the macroscopic physical quantities including the kinetic energy, the entropy, the free energy, and the asymptotic behavior of the solutions. Additionally, [18] reported a constrained optimization formulation to impose physical constraints to PINNs for the Fokker–Planck equation and the Boltzmann equation. Further issues including the inverse problem were investigated by [19, 28].

Although the neural network approach can be used to solve several complex PDEs in various kinds of settings, it requires relatively high computational cost compared to the traditional mesh-based schemes even for very simple differential equations due to its high dimensional optimization nature. To resolve this issue, we propose a novel loss function using Sobolev norms in this paper. Inspired by a recent study that incorporated derivative information for the training of neural networks [8], we develop a loss function that efficiently guides neural networks to find the solutions of PDEs. We prove that the H^1 and H^2 norms of the approximation errors converge to zero as our loss functions tend to zero for the 1-D Heat equation, the 1-D viscous Burgers equation, and the 1-D kinetic Fokker–Planck equation. Moreover, we show via several simulation results that the number of epochs to achieve a certain accuracy is significantly reduced as the order of derivatives in the loss function gets higher, provided that the solution is smooth. This study might pave the way for overcoming the issue of high computational cost when solving PDEs using neural networks.

The main contributions of this work are threefold: (1) We propose Sobolev-PINNs, a novel training framework with new loss functions, that enables the *Sobolev Training* of PINNs. (2) We prove that the proposed Sobolev-PINNs guarantee the convergence of PINNs in the corresponding Sobolev spaces although it is not a supervised learning task. (3) We empirically demonstrate the effect of *Sobolev Training* for several regression problems and the improved performances of Sobolev-PINNs in solving several PDEs including the heat equation, Burgers' equation, the Fokker–Planck equation, and high-dimensional Poisson equation.

2. Related works

Training neural networks to approximate the solutions of PDEs has been intensively studied over the past decades. For example, [21, 22] used neural networks to solve Ordinary Differential Equations (ODEs) and PDEs on a predefined set of grid points. Subsequently, [31] proposed a method to solve high-dimensional PDEs by approximating the solution using a neural network. They focused on the fact that the traditional finite mesh-based scheme becomes computationally intractable when the dimension becomes high. However, because neural networks are mesh-free function-approximators, they can solve high-dimensional PDEs by incorporating mini-batch sampling. Furthermore, the authors showed the convergence of the neural network to the solution of quasilinear parabolic PDEs under certain conditions.

Recently, [28] reported that one can use observed data to solve PDEs using Physics-

Informed Neural Networks (PINNs). Notably, PINNs can solve a supervised regression problem on observed data while satisfying any physical properties given by nonlinear PDEs. A significant advantage of PINNs is that the data-driven discovery of PDEs, also called the inverse problem, is possible with a small change in the code. The authors provided several numerical simulations for various types of nonlinear PDEs including the Navier–Stokes equation and Burgers' equation. The first theoretical justification for PINNs was provided by [30], who showed that a sequence of neural networks converges to the solutions of linear elliptic and parabolic PDEs in L^2 sense as the number of observed data increases. After that, the authors in [26] discovered some upper bounds of generalization error in terms of training error, number of training samples, and the stability of the PDE. [9] proved that ReLU-networks can approximate solutions of scalar conservation laws and also provided an upper bound on the generalization error.

A line of research that aims to deal with the stability and convergence issue of PINNs is recently drawing attention. [25] proposed a self-adaptive loss balancing algorithm that gives more weight to the region where the solution exhibits sharp transition. [34] claimed that the stiff gradient statistics causes an imbalance in the back-propagation and proposed a learning rate annealing algorithm to resolve it. [35] investigated the training dynamics of PINNs in the neural tangent kernel regime and proposed an adaptive loss balancing algorithm based on the eigenvalues of the tangent kernels. Another branch of works considered the training of PINNs as multi-objective learning (See, [5, 29, 33] for more information). There also exists a study aiming to enhance the convergence of PINNs [32, 33].

Additionally, several works related deep neural networks with PDEs but not by the direct approximation of the solutions of PDEs. For instance, [24] attempted to discover the hidden physics model from data by learning differential operators. A fast, iterative PDE-solver was proposed by learning to modify each iteration of the existing solver [13]. A deep Backward Stochastic Differential Equation (BSDE) solver was proposed and investigated in [11, 36] for solving high-dimensional parabolic PDEs by reformulating them using BSDE.

The main strategy of the present study is to leverage derivative information while solving PDEs via neural networks. The authors of [8] first proposed *Sobolev Training* that uses derivative information of the target function when training a neural network by slightly modifying the loss function. They showed that *Sobolev Training* had lower sample complexity than regular training, and therefore it is highly efficient in many applicable fields, such as regression and policy distillation problems. We adopt the concept of *Sobolev Training* to develop Sobolev-PINNs, a novel framework for the efficient training of a neural network for solving PDEs.

3. Loss functions

We consider the following Cauchy problem of PDEs:

$$Pu = f, \ (t,x) \in [0,T] \times \Omega, \tag{3.1}$$

$$Iu = g, \ (t,x) \in \{0\} \times \Omega, \tag{3.2}$$

$$Bu = h, \ (t,x) \in [0,T] \times \partial\Omega, \tag{3.3}$$

where P denotes a differential operator; I and B denote the initial and boundary operators, respectively; f, g, and h denote the inhomogeneous term, and initial and boundary data, respectively. In most studies that reported the neural network solutions of PDEs, a neural network was trained on uniformly sampled grid points $\{(t_i,x_j)\}_{i,j=1}^{N_t,N_x} \in [0,T] \times \Omega$, which were completely determined before training. One of the most intuitive ways

to make the neural network satisfy PDEs (3.1)–(3.3) is to minimize the following loss functional:

$$\mathcal{L}_(u_{nn};p) = \|Pu_{nn} - f\|^p_{L^p([0,T]\times\Omega)} + \|Iu_{nn} - g\|^p_{L^p(\Omega)} + \|Bu_{nn} - h\|^p_{L^p([0,T]\times\partial\Omega)},$$

where u_{nn} denotes the neural network and $p=1$ or 2, as they have been the most commonly used exponents in regression problems in previous studies. Evidently, an analytic solution u satisfies $\mathcal{L}(u)=0$, and thus one can conceptualize a neural network that makes $\mathcal{L}(u_{nn})=0$ a possible solution of PDEs (3.1)–(3.3). This statement is in fact proved for second-order parabolic equations with the Dirichlet boundary condition in [19], and for the Fokker–Planck equation with inflow and specular reflective boundary conditions in [14]. Both the proofs are based on the following inequality:

$$\|u - u_{nn}\|_{L^\infty(0,T;L^2(\Omega))} \le C\mathcal{L}(u_{nn};2),$$

for some constant C, which states that minimizing the loss functional implies minimizing the approximation error.

The main concept behind *Sobolev Training* is to minimize the error between the output and the target function, and that between the derivatives of the output and those of the target function. However, unlike the traditional supervised regression problem, neither the target function nor its derivative is provided while solving PDEs via neural networks. Thus, a special treatment is required to apply *Sobolev Training* for solving PDEs using neural networks. In this and the following sections, we propose several loss functions and prove that they guarantee the convergence of the neural network to the solution of a given PDE in the corresponding Sobolev space. Therefore, the proposed loss functions play similar roles to those in *Sobolev Training*.

We define the loss function that depends on the Sobolev norm $W^{k,p}$ as follows:

$$\mathcal{L}_{GE}(u_{nn};k,p,l,q) = \left\| \|P(u_{nn}(t,\cdot)) - f(t,\cdot)\|^q_{W^{l,q}(\Omega)} \right\|^p_{W^{k,p}([0,T])}, \tag{3.4}$$

$$\mathcal{L}_{IC}(u_{nn};l,q) = \|Iu_{nn}(t,x) - g(x)\|^q_{W^{l,q}(\Omega)}, \tag{3.5}$$

$$\mathcal{L}_{BC}(u_{nn};k,p,l,q) = \left\| \|Bu_{nn}(t,\cdot) - h(t,\cdot)\|^q_{W^{l,q}(\partial\Omega)} \right\|^p_{W^{k,p}([0,T])}. \tag{3.6}$$

REMARK 3.1. Here, the total loss with zero derivatives

$$\mathcal{L}^{(0)}_{TOTAL}(u_{nn}) = \mathcal{L}_{GE}(u_{nn};0,2,0,2) + \mathcal{L}_{IC}(u_{nn};0,2) + \mathcal{L}_{BC}(u_{nn};0,2,0,2)$$

coincides with the traditional L^2 loss function employed by [4,14,28,31].

When we train a neural network, the loss functions (3.4)–(3.6) are computed by Monte-Carlo approximation. Because the grid points are uniformly sampled, the loss functions are approximated as follows:

$$\mathcal{L}_{GE}(u_{nn};k,p,l,q) \approx \frac{T|\Omega|}{N_t N_x} \sum_{|\beta|\le k} \sum_{i=1}^{N_t} \left| \frac{d^\beta}{dt^\beta} \sum_{|\alpha|\le l} \sum_{j=1}^{N_x} |D^\alpha P(u_{nn}(t_i,x_j)) - D^\alpha f(t_i,x_j)|^q \right|^p,$$

$$\mathcal{L}_{IC}(u_{nn};l,q) \approx \frac{|\Omega|}{N_x} \sum_{|\alpha|\le l} \sum_{j=1}^{N_x} |D^\alpha u_{nn}(0,x_j) - D^\alpha g(x_j)|^q,$$

$$\mathcal{L}_{BC}(u_{nn};k,p,l,q)\approx\frac{T|\partial\Omega|}{N_tN_B}\sum_{|\beta|\le k}\sum_{i=1}^{N_t}\left|\frac{d^\beta}{dt^\beta}\sum_{|\alpha|\le l}\sum_{x_j\in\partial\Omega}|D^\alpha u_{nn}(t_i,x_j)-D^\alpha h(t_i,x_j)|^q\right|^p,$$

where α and β denote the conventional multi-indexes, and D denotes the spatial derivatives.

4. Theoretical results

In this section, we theoretically validate our claim that our loss functions guarantee the convergence of the neural network to the solution of a given PDE in the corresponding Sobolev spaces and that they play a similar role to those in *Sobolev Training* while solving PDEs via neural networks. Throughout this section, we will denote the strong solution of each equation by u, neural network solution by u_{nn}, and Sobolev spaces $W^{1,2}$ and $W^{2,2}$ by H^1 and H^2, respectively. We also assume that $u_{nn}\in C^\infty$ by considering the hyperbolic tangent function as a nonlinear activation function. The statements in this section are written in relatively intuitive forms. Mathematically rigorous statements and proofs are provided in Section 7. The proofs for the heat equation (Section 7.1) and the Poisson equation (Section 7.4) are straightforward from the well-known PDE theory.

4.1. The heat equation and Burgers' equation. We define the following three total loss functions for the heat equation and Burgers' equation:

$$\mathcal{L}^{(0)}_{TOTAL}(u_{nn})=\mathcal{L}_{GE}(u_{nn};0,2,0,2)+\mathcal{L}_{IC}(u_{nn};0,2)+\mathcal{L}_{BC}(u_{nn};0,2,0,2), \tag{4.1}$$

$$\mathcal{L}^{(1)}_{TOTAL}(u_{nn})=\mathcal{L}_{GE}(u_{nn};0,2,0,2)+\mathcal{L}_{IC}(u_{nn};1,2)+\mathcal{L}_{BC}(u_{nn};0,2,0,2), \tag{4.2}$$

$$\mathcal{L}^{(2)}_{TOTAL}(u_{nn})=\mathcal{L}_{GE}(u_{nn};1,2,0,2)+\mathcal{L}_{IC}(u_{nn};2,2)+\mathcal{L}_{BC}(u_{nn};0,2,0,2). \tag{4.3}$$

We then obtain the following convergence theorem:

THEOREM 4.1. *For the following 1-D heat and Burgers' equations:*

The heat equation	Burgers' equation
$u_t-u_{xx}=0$ in $(0,T]\times\Omega$,	$u_t+uu_x-\nu u_{xx}=0$ in $(0,T]\times\Omega$,
$u(0,x)=u_0(x)$ on Ω,	$u(0,x)=u_0(x)$ on Ω,
$u(t,x)=0$ on $[0,T]\times\partial\Omega$,	$u(t,x)=0$ on $[0,T]\times\partial\Omega$,

there hold, provided that u_{nn} is smooth,

$$\max_{0\le t\le T}\|u(t)-u_{nn}(t)\|_{L^2(\Omega)}\to 0 \text{ as } \mathcal{L}^{(0)}_{TOTAL}\to 0,$$

$$\operatorname*{esssup}_{0\le t\le T}\|u(t)-u_{nn}(t)\|_{H^1_0(\Omega)}\to 0 \text{ as } \mathcal{L}^{(1)}_{TOTAL}\to 0,$$

$$\operatorname*{esssup}_{0\le t\le T}\|u(t)-u_{nn}(t)\|_{H^2(\Omega)}\to 0 \text{ as } \mathcal{L}^{(2)}_{TOTAL}\to 0.$$

Proof. Proofs are provided in Theorem 7.1 for the heat equation, and Theorem 7.2 for Burgers' equation. ∎

4.2. The Fokker–Planck equation. For the Fokker–Planck equation, we need additional parameters for a new input variable v. We define the following two total loss functions for the Fokker–Planck equation:

$$\mathcal{L}^{(0;FP)}_{TOTAL}(u_{nn})=\mathcal{L}_{GE}(u_{nn};0,2,0,2,0,2)$$

$$+\mathcal{L}_{IC}(u_{nn};0,2,0,2)+\mathcal{L}_{BC}(u_{nn};0,2,0,2,0,2), \tag{4.4}$$

$$\begin{aligned}\mathcal{L}^{(1;FP)}_{TOTAL}(u_{nn}) =& \mathcal{L}_{GE}(u_{nn};0,2,1,2,1,2)\\ &+\mathcal{L}_{IC}(u_{nn};1,2,1,2)+\mathcal{L}_{BC}(u_{nn};0,2,0,2,0,2).\end{aligned} \tag{4.5}$$

We then have the following convergence theorem:

THEOREM 4.2. *For the 1-D Fokker–Planck equation with the periodic boundary condition:*

$$\begin{aligned}&u_t+vu_x-\beta(vu)_v-qu_{vv}=0, \text{ for } (t,x,v)\in[0,T]\times[0,1]\times\mathbb{R},\\ &u(0,x,v)=u_0(x,v), \text{ for } (x,v)\in[0,1]\times\mathbb{R},\\ &\partial^{\alpha}_{t,x,v}u(t,1,v)-\partial^{\alpha}_{t,x,v}u(t,0,v)=0, \text{ for } (t,v)\in[0,T]\times\mathbb{R},\end{aligned}$$

there hold, under assumptions (7.50) *and* (7.51),

$$\begin{aligned}&\sup_{0\le t\le T}\|u(t)-u_{nn}(t)\|_{L^2(\Omega\times[-V,V])}\to 0 \text{ as } \mathcal{L}^{(0;FP)}_{TOTAL}\to 0,\\ &\sup_{0\le t\le T}\|u(t)-u_{nn}(t)\|_{H^1(\Omega;L^2([-V,V]))}\to 0 \text{ as } \mathcal{L}^{(1;FP)}_{TOTAL}\to 0.\end{aligned}$$

Proof. Proofs are provided in Theorem 7.3 and Theorem 7.4 ∎

REMARK 4.1. The theorems in this section imply that the proposed loss functions guarantee the convergence of neural networks in the corresponding Sobolev spaces, thereby coinciding with the main idea of *Sobolev Training*. However, the theoretical results in this section imply the convergence of u_{nn} to u only when $\mathcal{L}_{TOTAL}\to 0$. In Section 5, we empirically demonstrate faster convergence of the error of the proposed Sobolev-PINNs.

REMARK 4.2. The theorems in this section cannot be directly generalized to the high-dimensional cases because even the 2-dimensional case starts involving the convexity of the boundary. Though it has also been shown that the Fokker-Planck operator has strong hypoellipticity and the solutions to the boundary problems are smooth even in the higher dimensional case, the proof requires long rigorous mathematical analysis. For more information, see [15, 16].

REMARK 4.3. Because we cannot access the label (which corresponds to the analytic solution) on the interior grid, solving PDEs using a neural network is not a fully supervised problem. Interestingly, by incorporating derivative information in the loss function, the proposed approach enables *Sobolev Training* even if neither the labels nor the derivatives of the target function are provided.

5. Experimental results

In this section, we provide experimental results for toy examples that comprise several regression problems and various kinds of differential equations, including the heat equation, Burgers' equation, the kinetic Fokker–Planck equation, and high-dimensional Poisson's equation. We employ a fully connected neural network, which is a natural choice for function approximation. We use the hyperbolic tangent function as a nonlinear activation function. Although $ReLU(x)=\max(0,x)$ is a frequent choice in modern machine learning, it is not appropriate for solving PDEs because the second derivatives of the neural network vanish.

In appreciation of Automatic Differentiation, we can easily compute derivatives of any order of a neural network with respect to input data despite the compositional

structure; see [2] and references therein. We implemented our neural network using PyTorch, a widely used deep learning library [27]. For the numerical experiments, we used a neural network with three hidden layers each of which had d-256-256-256-1 neurons, where d denotes the input dimension. We used the ADAM optimizer [20], a popular gradient-based optimizer.

To see whether our loss functions performed more efficiently than the traditional L^2 loss function introduced in Remark 3.1, we kept everything the same except the loss function. We compared the loss functions on the basis of $\|u-u_{nn}\|_{L^2(\Omega)}/\|u\|_{L^2(\Omega)}$ test error for the toy examples and the high-dimensional Poisson equation, and $\|u-u_{nn}\|_{L^\infty(0,T;L^2(\Omega))}$ test error for the heat, Burgers, Fokker–Planck equation as in the left-hand side of the estimates in Theorems 4.1, 4.2. For each loss function, we recorded the number of epochs required to meet a certain error threshold and the test error. Considering the randomness due to network initialization, we repeated the training a hundred times. Conversely, we initialized a hundred different neural networks with uniform initialization and trained them in the same manner. To compute the test error, we used analytic solutions for the Heat equation, Burgers' equation, and the high-dimensional Poisson equation, and a numerical solution from [38] for the kinetic Fokker–Planck equation.

In Figures 5.1-5.4, histograms in the first column are generated using the number of epochs required to achieve a certain accuracy, the plots in the second column are the error plots for a hundred training instances, and the third column shows the actual computation time for training in seconds. We provide the average errors over a hundred instances in the error plot with the logarithmic axis scale. Evidently, our loss function significantly reduces the number of epochs and test errors when solving various kinds of PDEs using neural networks.

5.1. Toy examples. First, we consider two simple regression problems with target functions $\sin(x)$ and $ReLU(x)$, respectively. For these toy examples, we define the loss functions as follows:

$$
\begin{aligned}
L^2 \text{ loss} &= \|u_{nn}(x)-y(x)\|_2^2,\\
H^1 \text{ loss} &= \|u_{nn}(x)-y(x)\|_2^2+\|u'_{nn}(x)-y'(x)\|_2^2,\\
H^2 \text{ loss} &= \|u_{nn}(x)-y(x)\|_2^2+\|u'_{nn}(x)-y'(x)\|_2^2+\|u''_{nn}(x)-y''(x)\|_2^2,
\end{aligned}
$$

where $y(x)$ denotes either $\sin(x)$, or $ReLU(x)$. We uniformly sampled a hundred grid points from $[0,2\pi]$ for training $\sin(x)$. Similarly, we uniformly sampled a hundred grid points from $[-1,1]$ for training $ReLU(x)$. We expected the training to become fast using higher order derivatives as many as possible when training $\sin(x)$ and $ReLU(x)$. Figure 5.1 confirms our assumption to be true. Interestingly, although $ReLU(x)$ is not twice weakly differentiable at only one point $x=0$, the H^2 loss does not facilitate the training.

In order to explore the nature of *Sobolev Training*, we design more complicated toy examples. Consider the target functions $\sin(kx)$, for $k=1,2,...,5$, and $ReLU(kx)=\max(0,kx)$, for $k=1,2,3...,10$. As k increases, the target functions and their derivatives contain drastic changes in their values, so it is difficult to learn those functions. We hypothesize that in *Sobolev Training*, the training becomes faster since we give explicit label for the derivatives and it becomes easier to capture the drastic changes in the derivatives. This is empirically shown to be true in Figure 5.2. We train neural networks to approximate $\sin(kx)$, and $ReLU(kx)$ for different k and record the number of training epochs to achieve certain error threshold which can be regarded as a difficulty of the

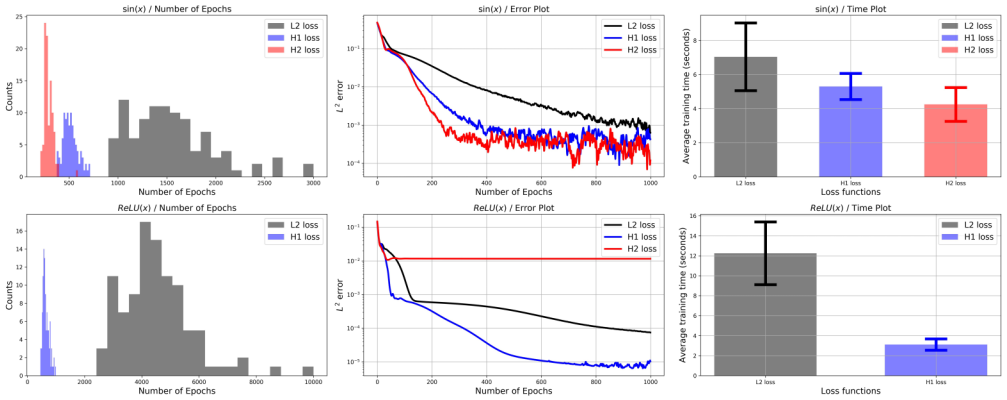

FIG. 5.1. *First row: results for* $\sin(x)$*, Second row: results for* $ReLU(x)$*. First column: Histograms generated from the repeated training of neural networks for training* $\sin(x)$*, and* $ReLU(x)$*. Second column: Test* L^2 *errors. Third column: Average training time for each loss function to achieve certain error threshold. Error bars are for standard deviations. The thresholds for the error are set to* 10^{-4}.

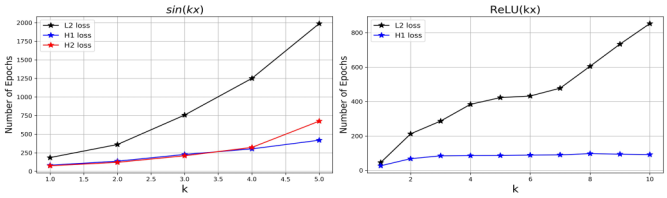

FIG. 5.2. *Average number of epochs to make error less than* 10^{-3} *increases in* L^2 *loss as* k *increases. However, when we use* H^1*, and* H^2 *losses, required number of epochs increases much more slowly or stays the same as* k *increases.*

problem. As one can see in Figure 5.2, the difficulty changes little to none when we train with H^1 and H^2 losses while the difficulty increases with k when L^2 loss is used. This implies that the difficulty of training barely changes in *Sobolev Training* even when the target function has stiff changes. The same observations are made when solving PDEs. The improvement of our loss functions as compared to L^2 loss function are more dramatic for Burgers' equation (which has stiff solution [28]) than for the heat equation, with the initial condition of f_2 (which has a higher frequency) than with the initial condition of f_1 in the Fokker–Planck equation, and as k increases for the high-dimensional Poisson equation, see Figure 5.5.

5.2. The heat equation & Burgers' equation. We now demonstrate the results of the *Sobolev Training* of the neural networks for solving PDEs. We begin with the 1-D heat equation, and Burgers' equation, which is the simplest PDE that combines both the nonlinear propagation effect and diffusive effect. Burgers' equation often appears as a simplification of a more complex and sophisticated model, such as the Navier–Stokes equation. The equations with the homogeneous Dirichlet boundary condition read as follows:

The heat equation attains a unique analytic solution $u(t,x) = \sin(x)\exp(-t)$; an analytic solution of Burgers' equation is provided in [1].

The Heat equation	Burgers' equation
$u_t - u_{xx} = 0$ in $(0,10] \times [0,\pi]$,	$u_t + uu_x - 0.2u_{xx} = 0$ in $(0,0.01] \times [0,1]$,
$u(0,x) = \sin(x)$ on $[0,\pi]$,	$u(0,x) = -\sin(\pi x)$ on $[0,1]$,
$u(t,x) = 0$ on $[0,10] \times \{0,\pi\}$.	$u(t,x) = 0$ on $[0,0.01] \times \{0,1\}$.

Although [31] indicated that iterative random sampling reduces the computational cost, we fixed the grid points before training because we aimed to compare the efficiency of our loss function with that of the traditional one. For the heat equation and the Burgers' equation, we uniformly sampled the grid points $\{t_i, x_j\}_{i,j=1}^{N_t,N_x}$ from $(0,T] \times \Omega$, where N_t and N_x denote the number of samples for interior t and x, respectively. For the initial and boundary conditions, we sampled the grid points from $\{t = 0, x_j\}_{j=1}^{N_x} \in \{0\} \times \Omega$ and $\{t_i, x_j\}_{i,j=1}^{N_t,N_B} \in [0,T] \times \partial\Omega$, respectively, where N_B denotes the number of grid points in $\partial\Omega$. Here, we set $N_t, N_x, N_B = 31$. We use 10,000 points uniformly sampled from $[0,T] \times \bar{\Omega}$ as a test dataset.

The L^2, H^1, and H^2 losses are the Monte-Carlo approximations of (4.1), (4.2), and (4.3), respectively, for the heat equation and Burgers' equation. Working on achieving a smooth solution, we observed that the H^2 loss performed the best, followed by the H^1 loss and then the L^2 loss in both accuracy, and computation time. We show the corresponding results in Figure 5.3.

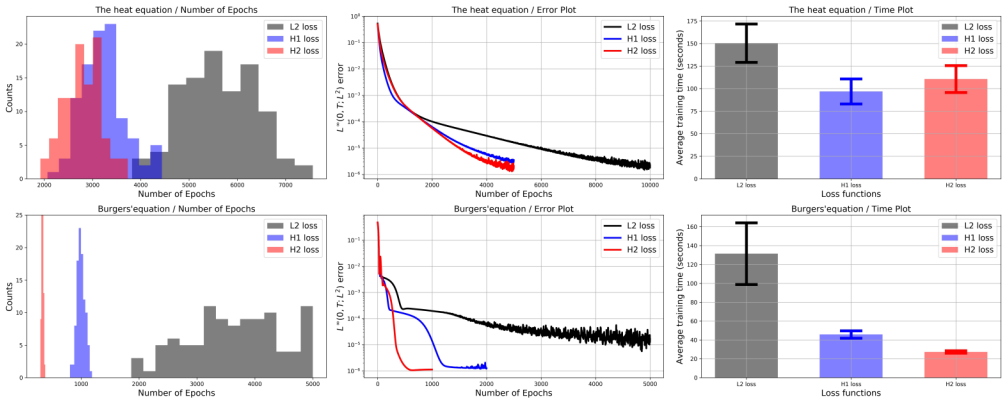

FIG. 5.3. *First row: results for the heat equation. Second row: results for Burgers' equation. First column: Histograms for the heat and Burgers' equation generated from a hundred neural networks for each loss function. Second column: Test $L^\infty(0,T;L^2(\Omega))$ errors. Third column: Average training time for each loss function to achieve certain error threshold. Error bars are for standard deviations. The thresholds for the error are set to 10^{-5}.*

5.3. The Fokker–Planck equation. The kinetic Fokker–Planck equation describes the dynamics of a particle whose behavior is similar to that of the Brownian particle. The Fokker–Planck operator has a strong regularizing effect not just in the velocity variable but also in the temporal and the spatial variables by the hypoellipticity. The Fokker–Planck equation has been considered in numerous physical circumstances including the Brownian motion described by the Uhlenbeck-Ornstein processes.

We provide two simulation results for different initial conditions for the 1-D Fokker–Planck equation with the periodic boundary condition. For the Fokker–Planck equation,

we adopted the idea of sampling from [14]. Because it is practically difficult to consider the entire space for the $v \in \mathbb{R}$ variable, we truncated the space for v as $[-5,5]$. We then uniformly sampled the grid points $\{t_i, x_j, v_k\}_{i,j,k=1}^{N_t,N_x,N_v}$ from $(0,T]\times\Omega\times[-5,5]$, where N_v denotes the number of samples for v. The grid points for the initial and periodic boundary conditions were accordingly sampled. The truncated equation reads as follows:

$$\begin{aligned}
&u_t + vu_x - \beta(vu)_v - qu_{vv} = 0, \text{ for } (t,x,v) \in (0,3]\times[0,1]\times[-5,5],\\
&u(0,x,v) = f(x,v), \text{ for } (x,v) \in [0,1]\times[-5,5],\\
&\partial^{\alpha}_{t,x,v} u(t,1,v) - \partial^{\alpha}_{t,x,v} u(t,0,v) = 0, \text{ for } (t,v) \in [0,3]\times[-5,5],
\end{aligned}$$

where $f(x,v)$ is either

$$f_1(x,v) = \frac{\exp(-v^2)}{\int_{-5}^{5}\exp(-v^2)dv}, \text{ or}$$
$$f_2(x,v) = \frac{(1+\cos(2\pi x))\exp(-v^2)}{\int_0^1\int_{-5}^{5}(1+\cos(2\pi x))\exp(-v^2)dvdx},$$

and $\beta = 0.1, q = 0.1$.

We define a test dataset by using 50,000 uniformly sampled points from $[0,3]\times[0,1]\times[-5,5]$. A numerical solution on the test data was computed by a method shown by [38] and used for computing the test error. L^2 loss and H^1 loss denote the Monte-Carlo approximations of (4.4) and (4.5), respectively. The values of N_t, N_x, and N_v were set to be 31, and the grid points were uniformly sampled. Expectedly, a solution of the Fokker–Planck equation could be estimated substantially faster using our loss function in both cases. We have provided the detailed results in Figure 5.4.

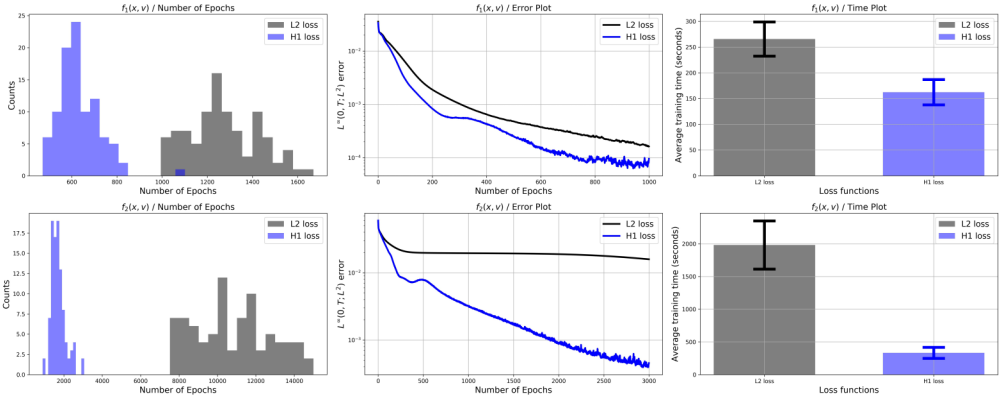

FIG. 5.4. *First row: results for f_1 initial condition. Second row: results for f_2 initial condition. First column: Histograms generated from a hundred neural networks for each loss function. Second column: Test $L^\infty(0,T;L^2(\Omega))$ errors. Third column: Average training time for each loss function to achieve certain error threshold. Error bars are for standard deviations. The thresholds for the errors for the initial conditions $f_1(x,v)$, and $f_2(x,v)$ are set to 10^{-4}, and 10^{-3}, respectively.*

5.4. The High-dimensional Poisson equation. The Poisson equation serves as an example problem in the recent literature; see [13,37,39]. In this section, we provide empirical results to demonstrate that the proposed loss functions perform satisfactorily

when equipped with iterative sampling for solving high-dimensional PDEs; see [31] for more information. Convergence result similar to those in Section 4 for the Poisson equation is given in Section 7.4. We consider the following high-dimensional Poisson equation with the Dirichlet boundary condition:

$$-\triangle u = \frac{\pi^2}{4}\sum_{i=1}^{d}\sin(\frac{\pi}{2}x_i), \text{ for } x\in\Omega=(0,1)^d,$$
$$u = \sum_{i=1}^{d}\sin(\frac{\pi}{2}x_i), \text{ for } x\in\partial\Omega,$$

where $x=(x_1,x_2,...,x_d)\in\Omega$. One can readily prove that $u(x)=\sum_{i=1}^{d}\sin(\frac{\pi}{2}x_i)$ is a strong solution. We compare the following three loss functions with each other:

$$\mathcal{L}_{TOTAL}^{(0;Poisson)}(u_{nn}) = \mathcal{L}_{GE}(u_{nn};0,2)+\mathcal{L}_{BC}(u_{nn};0,2), \tag{5.1}$$
$$\mathcal{L}_{TOTAL}^{(1;Poisson)}(u_{nn}) = \mathcal{L}_{GE}(u_{nn};1,2)+\mathcal{L}_{BC}(u_{nn};0,2), \tag{5.2}$$
$$\mathcal{L}_{TOTAL}^{(2;Poisson)}(u_{nn}) = \mathcal{L}_{GE}(u_{nn};1,2)+\mathcal{L}_{BC}(u_{nn};1,2). \tag{5.3}$$

Notably, the aforementioned loss functions have only x variable. Table 5.1 presents the relative errors on a predefined test dataset, which consists of 100,000 points uniformly sampled from Ω, for $d=10,50$, and 100. Evidently, in all cases, the proposed loss functions outperform the traditional L^2 loss function.

Dimension	$\mathcal{L}_{TOTAL}^{(0;Poisson)}$	$\mathcal{L}_{TOTAL}^{(1;Poisson)}$	$\mathcal{L}_{TOTAL}^{(2;Poisson)}$
10	0.38%	**0.22**%	**0.22**%
50	2.00%	1.74%	**1.52**%
100	3.15%	3.06%	**2.89**%

TABLE 5.1. *Average of the relative errors of a hundred neural networks for the high-dimensional Poisson equations. We uniformly sampled 500 data points from Ω for each epoch and trained the neural networks in 10000 epochs at a learning rate 10^{-4}.*

We next consider the high-dimensional Poisson equation with different boundary condition. In Subsection 5.1, we pointed out that the "difficulty" of learning $\sin(kx)$ increases as k increases. As a generalization of the argument, we consider the following PDEs:

$$-\triangle u = \frac{(k\pi)^2}{4}\sum_{i=1}^{d}\sin(\frac{k\pi}{2}x_i), \text{ for } x\in\Omega=(0,1)^d,$$
$$u = \sum_{i=1}^{d}\sin(\frac{k\pi}{2}x_i), \text{ for } x\in\partial\Omega,$$

for d=10, k = 1,3, and 5. As one can see in Figure 5.5, the improvement of Sobolev training gets bigger as k increases. This observation coincides with the one in Section 5, as we expected. Moreover, we present the comparison of training time to meet a certain error value for different loss functions in Figure 5.5. The result shows that it is advantageous to use the proposed loss functions in time, even in high-dimensional case.

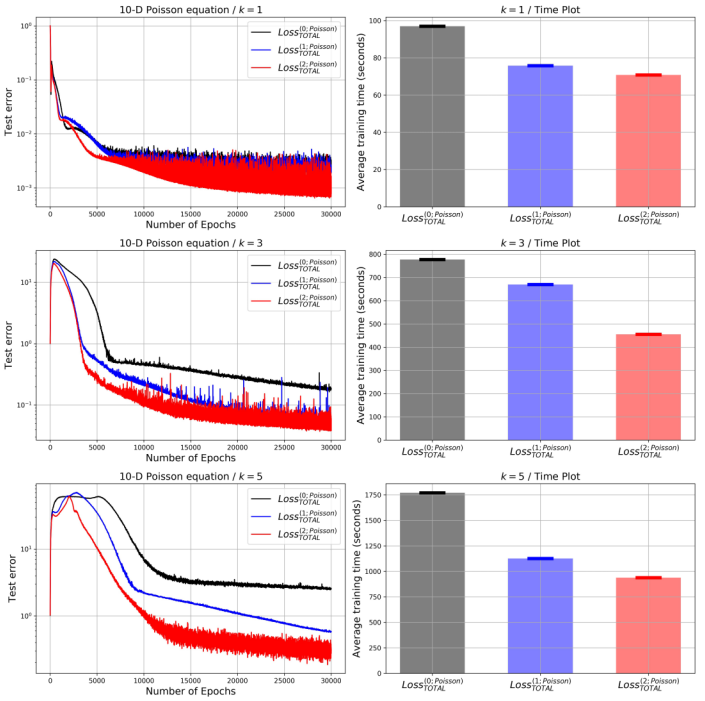

FIG. 5.5. *Left column: Test errors as training goes for different values of k. Right column: Required Training Time to achieve a certain test error.*

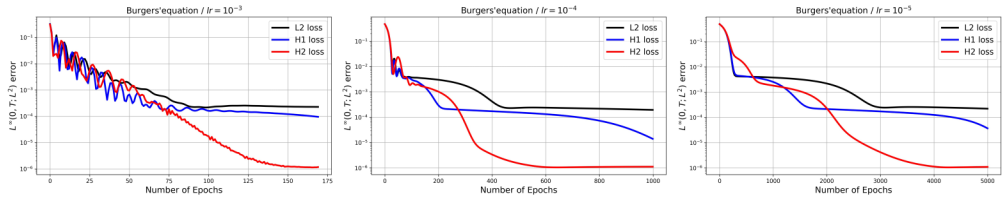

FIG. 5.6. *Test errors as training goes for different learning rates.*

5.5. Sobolev-PINNs with different learning rates. In this subsection, we provide several experiments that show the proposed loss functions generally perform better in different learning rates. We first show the results for Burgers' equation. In Figure 5.6, we show the test errors versus training epochs plot for different learning rates. We used 10^{-3},10^{-4},10^{-5} as learning rates and we observe that H^2 loss performs best followed by H^1 and L^2 loss functions.

We next present similar experiments for the high-dimensional Poisson equation. We trained 30 neural networks with different initializations with different learning rates. The average errors are presented in Figure 5.7. As in the Burgers' equation, our loss functions perform better than the traditional one in all learning rates.

5.6. Sobolev-PINNs in low-data regimes. [8] empirically demonstrated that the neural network captures the target function's shape and the effect of Sobolev

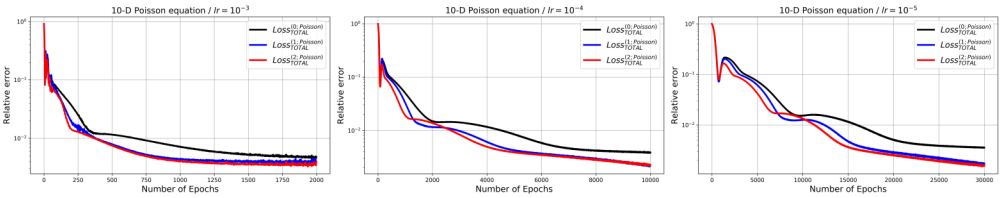

FIG. 5.7. *Test errors as training goes for different learning rates.*

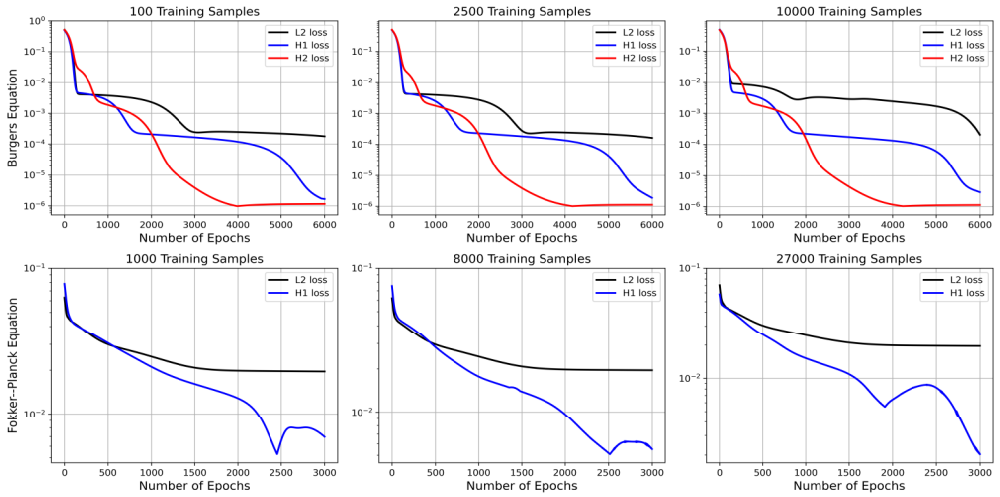

FIG. 5.8. *Test errors for different numbers of training samples.*

Training is much stronger in low-data regimes. Here, we present similar results for the proposed Sobolev-PINNs through the Burgers equation and the Fokker–Planck equation. Figure 5.8 shows test errors for different numbers of training samples for the Burgers equation and the Fokker–Planck equation. In both equations, Sobolev-trained networks successfully approximate the solutions with small test errors in extremely low-data regimes. We also observe that the effect of Sobolev Training continues from the low-data regime to the high-data regime.

6. Discussion and Conclusion

Inspired by *Sobolev Training*, we proposed Sobolev-PINNs, a novel framework involving new loss functions, which efficiently guided the training of neural networks for solving PDEs. We theoretically justified that the proposed loss functions guaranteed the convergence of a neural network to a solution of PDEs in the corresponding Sobolev spaces. We also discussed that the proposed theorems imply that the training becomes *Sobolev Training* by slightly modifying the loss function, although the process of estimating neural network solutions of PDEs is not fully supervised.

In addition to the toy examples, which showed the exceptional speed of *Sobolev Training*, we provided empirical evidence to demonstrate that Sobolev-PINNs expedited the training more than the traditional L^2 loss function. We believe that this can solve the problem associated with the high costs involved in estimating the neural network

solutions of PDEs. Moreover, our experiments on high-dimensional problems showed that the proposed loss function performed better when equipped with iterative grid sampling. The histograms in Figures 5.1-5.4 indicate that our loss function provided more stable training in that it reduced the variance in the distribution of the number of epochs (e.g., for the Burgers' equation, L^2 loss: 3651±812, H^1 loss: 995±71, and H^2 loss: 331±15). Thus, the training, when governed by our loss function, became robust to the random initialization of the weights.

7. Formal statements and proofs for the Theorems in Section 4

7.1. The heat equation.

We denote the strong solution of the heat equation

$$\begin{aligned} u_t - u_{xx} &= 0 \text{ in } (0,T]\times\Omega, \\ u(0,x) &= u_0(x) \text{ on } \Omega, \\ u(t,x) &= 0 \text{ on } [0,T]\times\partial\Omega, \end{aligned}$$

by u and the neural network solution by u_{nn}. Then, $v = u - u_{nn}$ satisfies:

$$\begin{aligned} v_t - v_{xx} &= f(t,x) \text{ in } (0,T]\times\Omega, \\ v(0,x) &= g(x) \text{ on } \Omega, \\ v(t,x) &= 0 \text{ on } [0,T]\times\partial\Omega, \end{aligned} \tag{7.1}$$

for some f, and g. Here, we can set the boundary to be zero by multiplying $B(x)$, where $B(x)$ is a smooth function satisfying $B(x)\begin{cases} = 0, & x\in\partial\Omega \\ \neq 0, & x\in\Omega \end{cases}$. Then the following holds:

THEOREM 7.1 (Theorem 7.1.5 in [10]). *If $g\in H^2(\Omega), f_t\in L^2(0,T;L^2(\Omega))$, then,*

$$\max_{0\le t\le T}\|v(t)\|_{L^2(\Omega)} \le C_1(\|f\|_{L^2(0,T;L^2(\Omega))} + \|g\|_{L^2(\Omega)}), \tag{7.2}$$

$$\operatorname*{esssup}_{0\le t\le T}\|v(t)\|_{H_0^1(\Omega)} \le C_2(\|f\|_{L^2(0,T;L^2(\Omega))} + \|g\|_{H_0^1(\Omega)}), \tag{7.3}$$

$$\operatorname*{esssup}_{0\le t\le T}\|v(t)\|_{H^2(\Omega)} \le C_3(\|f\|_{H^1(0,T;L^2(\Omega))} + \|g\|_{H^2(\Omega)}), \tag{7.4}$$

for some C_1, C_2, C_3.

By applying above theorem to (7.1), we get the results of Theorem 4.1.

REMARK 7.1. The left-hand sides in (7.2) - (7.4) are the errors of neural networks in corresponding norms, and the right-hand sides are the losses (4.1) - (4.3) for the heat equation, respectively. This implies that the proposed loss functions are the upper bounds of the errors in the Sobolev spaces, and by minimizing them, we can expect the effect of Sobolev Training when solving PDEs with neural networks.

In the rest of this section, we will show the similar results for Burgers' equation and the Fokker–Planck equation.

7.2. Burgers' equation.

We consider the strong solution u of the following Burgers equation in a bounded interval $\Omega = [a,b]$,

$$\partial_t u + u\partial_x u - \partial_x^2 u = 0 \quad \text{in } \Omega, \tag{7.5}$$

$$u = 0 \quad \text{on } \partial\Omega, \tag{7.6}$$

and the corresponding neural network solution u_{nn} satisfying

$$\partial_t u_{nn} + u_{nn}\partial_x u_{nn} - \partial_x^2 u_{nn} = f \quad \text{in } \Omega, \tag{7.7}$$

$$u_{nn} = 0 \quad \text{on } \partial\Omega, \tag{7.8}$$

with the inital data $u(0,\cdot)$ and $u_{nn}(0,\cdot)$, respectively.

The following proposition ensures the existence of a strong solution to the initial boundary value problem (7.5)–(7.6) (see [3]). Here, we multiply $B(x)$ to $u_{nn}(t,x)$ in order to meet the boundary condition. We use the notation $\lesssim$ where the relation $A \lesssim B$ stands for $A \le CB$, where C denotes a generic constant.

PROPOSITION 7.1 (Theorem 1.2 in [3]). *Let $u_0 \in H_0^1$. Then there exists a time $T^* = T^*(u_0) > 0$ such that the problem (7.5)–(7.6) with initial data u_0 has a unique solution of u satisfying*

$$u \in L^2(0,T^*;H^2(\Omega)) \cap C([0,T^*);H_0^1(\Omega)),$$
$$u_t \in L^2(0,T^*;L^2(\Omega)).$$

Furthermore, if $T^* < \infty$, then $\|u\|_{H^1(\Omega)} \to \infty$ as $t \to T^*$.

We will show that the following theorem holds.

THEOREM 7.2. *Let u and u_{nn} be strong solutions of (7.5)–(7.6) and (7.7)–(7.8) respectively, on the time interval $[0,T]$. For $w := u_{nn} - u$, following statements are valid.*

(1) There exists a continuous function

$$F_0 = F_0\left(\|w(0,\cdot)\|_2^2, \int_0^T \|f\|_2^2 dt, \int_0^T \|\partial_x u\|_2^2 dt\right)$$

such that

$$\sup_{0\le t\le T} \|w\|_2^2 + \int_0^T \|\partial_x w\|_2^2 dt \le F_0 \to 0,$$

as

$$\|w(0,\cdot)\|_2^2, \int_0^T \|f\|_2^2 dt \to 0.$$

(2) There exists a continuous function

$$F_1 = F_1\left(\|w(0,\cdot)\|_{H^1}^2, \int_0^T \|f\|_2^2 dt, \int_0^T \|\partial_x u\|_2^2 dt\right)$$

such that

$$\sup_{0\le t\le T} \|w\|_{H^1}^2 + \int_0^T \|\partial_x w\|_{H^1}^2 + \|\partial_t w\|_2^2 dt \le F_1 \to 0, \tag{7.9}$$

as

$$\|w(0,\cdot)\|_{H^1}^2, \int_0^T \|f\|_2^2 dt \to 0.$$

(3) There exists a continuous function

$$F_2 = F_2\left(\|w(0,\cdot)\|_{H^2}^2, \int_0^T \|f\|_2^2 + \|\partial_t f\|_2^2 dt, \sup_{0\le t\le T}\|\partial_x u\|_2^2 + \int_0^T \|\partial_t u\|_2^2 dt\right)$$

such that

$$\sup_{0\le t\le T}\left(\|w\|_{H^2}^2 + \|w_t\|_2^2\right) + \int_0^T \|\partial_x w\|_{H^2}^2 dt \le F_2 \to 0, \tag{7.10}$$

as

$$\|w(0,\cdot)\|_{H^2}^2, \int_0^T \|f\|_2^2 + \|\partial_t f\|_2^2 dt \to 0.$$

REMARK 7.2. By Morrey's embedding theorem and Poincare's inequality, for $f \in H_0^1(\Omega)$, we have the following inequality,

$$\|f\|_\infty^2 \le C_1(\|f\|_2^2 + \|f_x\|_2^2) \le C_2\|f_x\|_2^2, \tag{7.11}$$

for some C_1, C_2 only depending on Ω (see Theorem 5.6.4 in [10] and Proposition 8.13 in [6]). Throughout the proof, we widely use (7.11).

Proof. Subtracting (7.5) from (7.7), we get equations of w as follows.

$$w_t - w_{xx} + ww_x + wu_x + uw_x = f \quad \text{in } \Omega, \tag{7.12}$$
$$w = 0 \quad \text{on } \partial\Omega, \tag{7.13}$$
$$w(0,\cdot) = g \quad \text{in } \Omega. \tag{7.14}$$

By multiplying w to (7.12) and integrating by parts in Ω, we have

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\|w\|_2^2 + \|w_x\|_2^2 &= \int_\Omega fw - \int_\Omega w^2 w_x - \int_\Omega w^2 u_x - \int_\Omega uww_x + \int_{\partial\Omega} ww_x, \\ &= \sum_{k=1}^5 I_k^0. \end{aligned} \tag{7.15}$$

Now we estimate the terms on the right-hand side of (7.15). Applying Young's inequality, Hölder's inequality, the Sobolev inequality, and the Poincare inequality, we have

$$I_1^0 \le \|f\|_2\|w\|_2 \le \frac{1}{\epsilon}\|f\|_2^2 + \epsilon\|w\|_2^2 \overset{(7.11)}{\le} \frac{1}{\epsilon}\|f\|_2^2 + C_1\epsilon\|w_x\|_2^2, \tag{7.16}$$

$$I_2^0 = \int_{\partial\Omega}\frac{1}{3}w^3 = \frac{1}{3}w^3(b) - \frac{1}{3}w^3(a) = 0, \tag{7.17}$$

$$I_3^0 \le \|w\|_\infty\|w\|_2\|u_x\|_2 \le \frac{1}{\epsilon}\|u_x\|_2^2\|w\|_2^2 + \epsilon\|w\|_\infty, \tag{7.18}$$
$$\overset{(7.11)}{\le} \frac{1}{\epsilon}\|u_x\|_2^2\|w\|_2^2 + C_2\epsilon\|w_x\|_2^2,$$

$$I_4^0 \le \|u\|_\infty\|w\|_2\|w_x\|_2 \le \frac{1}{\epsilon}\|u\|_\infty\|w\|_2^2 + \epsilon\|w_x\|_2^2, \tag{7.19}$$
$$\overset{(7.11)}{\le} \frac{C_3}{\epsilon}\|u_x\|_2^2\|w\|_2^2 + \epsilon\|w_x\|_2^2,$$

$$I_5^0 = w(b)w_x(b) - w(a)w_x(a) = 0, \tag{7.20}$$

for any small $\epsilon > 0$ and for some constants $C_1, C_2,$ and C_3 only depending on Ω. Overall, we have

$$\frac{1}{2}\frac{d}{dt}\|w\|_2^2 + \|w_x\|_2^2 \le \frac{1}{\epsilon}\|f\|_2^2 + \epsilon(C_1 + C_2 + 1)\|w_x\|_2^2 + \frac{1+C_3}{\epsilon}\|u_x\|_2^2\|w\|_2^2.$$

If we take $\epsilon < \frac{1}{C_1+C_2+1}$, then we obtain the following inequality

$$\frac{d}{dt}\|w\|_2^2 + \|w_x\|_2^2 \lesssim \|u_x\|_2^2\|w\|_2^2 + \|f\|_2^2, \tag{7.21}$$

(7.21) and the Grönwall inequality imply that

$$\sup_{0\le t\le T}\|w\|_2^2 \lesssim e^{\int_0^T \|u_x\|_2^2 dt}\left(\|g\|_2^2 + \int_0^T \|f\|_2^2 dt\right). \tag{7.22}$$

Let us denote

$$\int_0^T \|u_x\|_2^2 dt = \beta.$$

Now we integrate (7.21) between 0 and T and drop the term $\|w(T)\|_2^2$ on the left-hand side to obtain

$$\begin{aligned}\int_0^T \|w_x\|^2 dt &\lesssim \|g\|_2^2 + \int_0^T \|u_x\|_2^2\|w\|_2^2 + \|f\|_2^2 dt \\ &\lesssim (\beta e^{\beta} + 1)\left(\|g\|_2^2 + \int_0^T \|f\|_2^2 dt\right).\end{aligned} \tag{7.23}$$

This completes the proof of (1) of Theorem 7.2.

Next, by multiplying $-w_{xx}$ to (7.12) and integrating by parts in Ω, we obtain

$$\begin{aligned}&\frac{1}{2}\frac{d}{dt}\|w_x\|_2^2 + \|w_{xx}\|_2^2 \\ &= -\int_\Omega f w_{xx} + \int_\Omega w w_x w_{xx} + \int_\Omega u_x w w_{xx} + \int_\Omega u w_x w_{xx} + \int_{\partial\Omega} w_t w_x, \\ &= \sum_{k=1}^5 I_k^1.\end{aligned} \tag{7.24}$$

Similarly to (7.16)–(7.20), we estimate the terms on the right-hand side of (7.24).

$$I_1^1 \le \|f\|_2\|w_{xx}\|_2^2 \le \frac{1}{\epsilon}\|f\|_2^2 + \epsilon\|w_{xx}\|_2^2, \tag{7.25}$$

$$I_2^1 \le \|w\|_\infty\|w_x\|_2\|w_{xx}\|_2 \overset{(7.11)}{\le} \frac{C_1}{\epsilon}\|w_x\|^4 + \epsilon\|w_{xx}\|_2^2, \tag{7.26}$$

$$I_3^1 \le \|w\|_\infty\|u_x\|_2\|w_{xx}\|_2 \overset{(7.11)}{\le} \frac{C_2}{\epsilon}\|u_x\|_2^2\|w_x\|_2^2 + \epsilon\|w_{xx}\|_2^2, \tag{7.27}$$

$$I_4^1 \le \|u\|_\infty\|w_x\|_2\|w_{xx}\|_2 \overset{(7.11)}{\le} \frac{C_3}{\epsilon}\|u_x\|_2^2\|w_x\|_2^2 + \epsilon\|w_{xx}\|_2^2, \tag{7.28}$$

$$I_5^1 = w_t(b)w_x(b) - w_t(a)w_x(a) = 0, \tag{7.29}$$

for any small $\epsilon > 0$ and some constants C_1, C_2, and C_3 only depending on Ω. Applying estimates (7.25)–(7.29) to (7.24) and choosing small enough epsilon, we have the following inequality

$$\frac{d}{dt}\|w_x\|_2^2 + \|w_{xx}\|_2^2 \lesssim (\|w_x\|_2^2 + \|u_x\|_2^2)\|w_x\|_2^2 + \|f\|_2^2. \tag{7.30}$$

It follows from (7.23), (7.30), and the Grönwall inequality that

$$\begin{aligned}\sup_{0\le t\le T}\|w_x\|_2^2 &\lesssim e^{\int_0^T \|w_x\|_2^2 + \|u_x\|_2^2 dt}\left(\|g_x\|_2^2 + \int_0^T \|f\|_2^2 dt\right)\\ &\lesssim e^{\beta}e^{F_0}\left(\|g_x\|_2^2 + \int_0^T \|f\|_2^2 dt\right).\end{aligned} \tag{7.31}$$

In a similar way to (7.23), there exists a function $\tilde{F}_1 = F_1(\|f\|_{L^2(0,T;L^2)}, \|g\|_{H^1}, \beta)$ such that

$$\int_0^T \|w_{xx}\|_2^2 dt \lesssim F_1. \tag{7.32}$$

(2) of Theorem 7.2 follows from (7.31), (7.32), and the fact that

$$w_t = f + w_{xx} - ww_x - wu_x - uw_x. \tag{7.33}$$

Finally, we differentiate (7.12) with respect to t, then we obtain

$$w_{tt} - w_{xxt} + w_t w_x + ww_{xt} + w_t u_x + wu_{xt} + u_t w_x + uw_{xt} = f_t \quad \text{in } \Omega, \tag{7.34}$$

$$w_t = 0 \quad \text{on } \partial\Omega, \tag{7.35}$$

$$w_t(0) = f + g_{xx} - gg_x - gu_{0x} - u_0 g_x \quad \text{in } \Omega. \tag{7.36}$$

By multiplying w_t to (7.34) and integrating by parts in Ω, we have

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\|w_t\|_2^2 + \|w_{xt}\|_2^2 &= \int_\Omega f_t w_t - \int_\Omega w_t^2 w_x - \int_\Omega ww_t w_{xt} - \int_\Omega w_t^2 u_x\\ &\quad - \int_\Omega ww_t u_{xt} - \int_\Omega u_t w_t w_x - \int_\Omega uw_t w_{xt} + \int_{\partial\Omega} w_t w_{xt},\\ &= \sum_{k=1}^{8} I_k^2.\end{aligned} \tag{7.37}$$

Terms on the right-hand side of (7.37) are estimated by

$$I_1^2 \le \|f_t\|_2\|w_t\|_2 \overset{(7.11)}{\le} \frac{1}{\epsilon}\|f_t\|_2^2 + C_1\epsilon\|w_{xt}\|_2^2, \tag{7.38}$$

$$I_2^2 \le \|w_t\|_\infty\|w_t\|_2\|w_x\|_2 \overset{(7.11)}{\le} \frac{1}{\epsilon}\|w_x\|_2^2\|w_t\|_2^2 + C_2\epsilon\|w_{xt}\|_2^2, \tag{7.39}$$

$$I_3^2 \le \|w\|_\infty\|w_t\|_2\|w_{xt}\|_2 \overset{(7.11)}{\le} \frac{C_3}{\epsilon}\|w_x\|_2^2\|w_t\|_2^2 + \epsilon\|w_{xt}\|_2^2, \tag{7.40}$$

$$I_4^2 \le \|w_t\|_\infty \|w_t\|_2 \|u_x\|_2 \overset{(7.11)}{\le} \frac{1}{\epsilon}\|u_x\|_2^2\|w_t\|_2^2 + C_4\epsilon\|w_{xt}\|_2^2, \tag{7.41}$$

$$I_5^2 + I_6^2 = \int_\Omega w w_{xt} u_t \le \|w\|_\infty \|u_t\|_2 \|w_{xt}\|_2 \overset{(7.11)}{\le} \frac{C_5}{\epsilon}\|u_t\|_2^2\|w_t\|_2^2 + \epsilon\|w_{xt}\|_2^2, \tag{7.42}$$

$$I_7^2 \le \|u\|_2 \|w_t\|_2 \|w_{xt}\|_2 \overset{(7.11)}{\le} \frac{C_6}{\epsilon}\|u_x\|_2^2\|w_t\|_2^2 + \epsilon\|w_{xt}\|_2^2, \tag{7.43}$$

$$I_8^2 = 0, \tag{7.44}$$

for any small $\epsilon > 0$ and for some constants $C_1, C_2, C_3, C_4, C_5$and ,C_6 only depending on Ω. Applying estimates (7.38)–(7.44) to (7.37) and choosing small enough ϵ, we have the following inequality

$$\frac{d}{dt}\|w_t\|_2^2 + \|w_{xt}\|_2^2 \lesssim \left(\|w_x\|_2^2 + \|u_x\|_2^2 + \|u_t\|_2^2\right)\|w_t\|_2^2 + \|f_t\|_2^2. \tag{7.45}$$

It follows from (7.36), (7.45) and the Grönwall inequality that

$$\begin{aligned}\sup_{0\le t\le T}\|w_t\|_2^2 &\lesssim e^{\int_0^T \|w_x\|_2^2 + \|u_x\|_2^2 + \|u_t\|_2^2 dt}\left(\|w_t(0)\|_2^2 + \int_0^T \|f_t\|_2^2\right) \\ &\lesssim e^{\gamma} e^{F_0}\left(\|f_0\|_2^2 + \|g_{xx}\|_2^2 + \|g_x\|_2^4 + \|u_{0x}\|_2^2\|g_x\|_2^2 + \int_0^T \|f_t\|_2^2 dt\right).\end{aligned} \tag{7.46}$$

where

$$\gamma = \int_0^T \|u_x\|_2^2 + \|u_t\|_2^2 dt.$$

In a similar way to the proof of (2) of Theorem 7.2, (3) of Theorem 7.2 follows from (7.33) and (7.46). This completes the proof of the theorem. ∎

7.3. The Fokker–Planck equation.

7.3.1. Boundary loss design. Define the loss function for the periodic boundary condition as

$$\begin{aligned}\mathcal{L}_{BC} &= \sum_{|\alpha|=1}\int_0^T dt \int_{-5}^5 dv \left|\partial_{t,x,v}^\alpha f^{nn}(t,1,v;m,w,b) - \partial_{t,x,v}^\alpha f^{nn}(t,0,v;m,w,b)\right|^2 \\ &\approx \frac{1}{N_{i,k}} \sum_{|\alpha|=1,i,k} \left|\partial_{t,x,v}^\alpha f^{nn}(t_i,1,v_k;m,w,b) - \partial_{t,x,v}^\alpha f^{nn}(t_i,0,v_k;m,w,b)\right|^2.\end{aligned} \tag{7.47}$$

7.3.2. The Fokker–Planck equation in a periodic interval. In this section, we introduce an L^2 energy method for the Fokker–Planck equation and introduce a regularity inequality for the solutions to the equation. Throughout the section, we will abuse the notation and use both notations $\partial_z u$ and u_z for the same derivative of u with respect to z.

We consider the Fokker–Planck equation in a periodic interval $[0,1]$:

$$\begin{aligned}&u_t + v u_x - \beta(vu)_v - q u_{vv} = 0, \text{ for } (t,x,v) \in [0,T]\times[0,1]\times\mathbb{R}, \\ &u(0,x,v) = u_0(x,v), \text{ for } (x,v) \in [0,1]\times\mathbb{R}, \text{ and} \\ &\partial_{t,x,v}^\alpha u(t,1,v) - \partial_{t,x,v}^\alpha u(t,0,v) = 0, \text{ for } (t,v) \in [0,T]\times\mathbb{R},\end{aligned} \tag{7.48}$$

for any 3-dimensional multi-index α such that $|\alpha| \leq 1$ and a given initial distribution $u_0 = u_0(x,v)$. Now we consider the Fokker–Planck equation that the corresponding neural network solution u_{nn} would satisfy:

$$\begin{aligned} &(u_{nn})_t + v(u_{nn})_x - \beta(vu_{nn})_v - q(u_{nn})_{vv} = f \text{ for } (t,x,v) \in [0,T] \times [0,1] \times [-5,5], \\ &u_{nn}(0,x,v) = g, \text{ for } (x,v) \in [0,1] \times [-5,5], \\ &\sum_{|\alpha|=1} \int_0^T dt \int_{-5}^5 dv \left| (\partial^\alpha_{t,x,v} u_{nn})(t,1,v) - (\partial^\alpha_{t,x,v} u_{nn})(t,0,v) \right|^2 \leq L, \end{aligned} \tag{7.49}$$

for any 3-dimensional multi-index α such that $|\alpha| \leq 1$ and given $f = f(t,x,v)$, $g = g(x,v)$, and a constant $L > 0$. Suppose that f, g and h are C^1 functions. Also, we suppose that the a priori solutions u and u_{nn} are sufficiently smooth; indeed, we require them to be in $C^{1,1,2}_{t,x,v}$.

For the a priori solution u and u_{nn} to (7.48) and (7.49), assume that if $|v|$ is sufficiently large, then we have that for some sufficiently small $\epsilon > 0$,

$$\sup_{t \in [0,T]} \left\| \partial^\alpha_{t,x,v} u(t,\cdot,\pm 5) - \partial^\alpha_{t,x,v} u_{nn}(t,\cdot,\pm 5) \right\|_{L^2_x([0,1])} \leq \epsilon, \tag{7.50}$$

for $|\alpha| \leq 1$ and $\alpha = (0,0,2)$. Also, suppose that

$$|\partial^\alpha_{t,x,v} u(t,x,\pm 5)|, |\partial^\alpha_{t,x,v} u_{nn}(t,x,\pm 5)| \leq C, \tag{7.51}$$

for some $C < \infty$ for $|\alpha| \leq 1$ and $\alpha = (0,0,2)$. Now we introduce the following theorem on the energy estimates:

THEOREM 7.3. *Let u and u_{nn} be the classical solutions to (7.48) and (7.49), respectively. Then we have*

$$\begin{aligned} &\sup_{0 \leq t \leq T} \|u_{nn}(t) - u(t)\|_2^2 + 2(q - \varepsilon) \int_0^T \|\partial_v u_{nn}(s) - \partial_v u(s)\|_2^2 ds \\ &\leq \left(\|g - u_0\|_2^2 + \frac{L}{2} \right) \exp\left[\left(1 + \frac{25\beta^2}{2\varepsilon} \right) T \right] + \int_0^T \|f(s)\|_2^2 ds + 2q\epsilon CT, \end{aligned}$$

for any $\varepsilon \in (0,q)$, where $L, u_0, f, g, \beta, q, m, \epsilon$, and C are given in (7.48)-(7.51).

Proof. Define $w \overset{\text{def}}{=} u_{nn} - u$. Then by (7.48) and (7.49), w satisfies

$$\begin{aligned} &w_t + vw_x - \beta v w_v - q w_{vv} = f \text{ for } (t,x,v) \in [0,T] \times [0,1] \times [-5,5], \\ &w(0,x,v) = w_0, \text{ for } (x,v) \in [0,1] \times [-5,5], \end{aligned} \tag{7.52}$$

where $w_0 \overset{\text{def}}{=} g - u_0$. By multiplying w to (7.52) and integrating with respect to $dxdv$, we have

$$\begin{aligned} &\frac{1}{2} \frac{d}{dt} \iint_{[0,1] \times [-5,5]} |w|^2 dxdv + \iint_{[0,1] \times [-5,5]} v w_x w dxdv - \iint_{[0,1] \times [-5,5]} q w_{vv} w dxdv \\ &= \iint_{[0,1] \times [-5,5]} f w dxdv + \iint_{[0,1] \times [-5,5]} \beta v w_v w dxdv. \end{aligned}$$

Then we take the integration by parts and obtain that

$$\tfrac{1}{2} \tfrac{d}{dt} \iint_{[0,1] \times [-5,5]} |w|^2 dxdv + \tfrac{1}{2} \int_{-5}^5 dv \; v(w(t,1,v)^2 - w(t,0,v)^2) + q \iint_{[0,1] \times [-5,5]} |w_v|^2 dxdv$$

$$
\begin{aligned}
&= \iint_{[0,1]\times[-5,5]} fw dxdv + \iint_{[0,1]\times[-5,5]} \beta v w_v w dxdv \\
&\quad + q\int_{[0,1]} w_v(t,x,5)w(t,x,5)dx - q\int_{[0,1]} w_v(t,x,-5)w(t,x,-5)dx \\
&\overset{\text{def}}{=} I_1 + I_2 + I_3 + I_4.
\end{aligned}
$$

We first define

$$
A(t) \overset{\text{def}}{=} \left| \frac{1}{2}\int_{-5}^{5} dv\ v(w(t,1,v)^2 - w(t,0,v)^2) \right|.
$$

We now estimate I_1-I_4 on the right-hand side. By the Hölder inequality and Young's inequality, we have

$$
|I_1| \le \|f\|_2\|w\|_2 \le \frac{1}{2}\|f\|_2^2 + \frac{1}{2}\|w\|_2^2,
$$

where we denote

$$
\|h\|_2 \overset{\text{def}}{=} \iint_{[0,1]\times[-5,5]} |h|^2 dxdv.
$$

Similarly, we observe that

$$
|I_2| \le 5\beta\|w_v\|_2\|w\|_2 \le \varepsilon\|w_v\|_2^2 + \frac{25\beta^2}{4\varepsilon}\|w\|_2^2,
$$

for a sufficiently small $\varepsilon > 0$ as $|v| \le 5$. By (7.50), we have

$$
\begin{aligned}
|I_3 + I_4| \le & q\|w_v(t,\cdot,5)\|_{L_x^2}\|w(t,\cdot,5) - w(t,\cdot,-5)\|_{L_x^2} \\
& + q\|w_v(t,\cdot,5) - w_v(t,\cdot,-5)\|_{L_x^2}\|w(t,\cdot,-5)\|_{L_x^2} \le 2q\epsilon C.
\end{aligned}
$$

Altogether, we have

$$
\frac{d}{dt}\|w\|_2^2 + 2(q-\varepsilon)\|w_v\|_2^2 \le \|f\|_2 + \left(1 + \frac{25\beta^2}{2\varepsilon}\right)\|w\|_2^2 + A(t) + 2q\epsilon C.
$$

We integrate with respect to the temporal variable on $[0,t]$ and obtain

$$
\begin{aligned}
&\|w(t)\|_2^2 + 2(q-\varepsilon)\int_0^t \|w_v(s)\|_2^2 ds \\
&\le \|w(0)\|_2^2 + \int_0^t \left(\|f(s)\|_2 + \left(1 + \frac{25\beta^2}{2\varepsilon}\right)\|w(s)\|_2^2 + A(s) + 2q\epsilon C \right) ds.
\end{aligned}
$$

By $(7.49)_3$, we have

$$
\int_0^t A(s)ds \le \frac{L}{2}.
$$

Thus, by the Grönwall inequality, we have

$$
\|w(t)\|_2^2 + 2(q-\varepsilon)\int_0^t \|w_v(s)\|_2^2 ds \le \left(\|w_0\|_2^2 + \frac{L}{2}\right)\exp\left[\left(1 + \frac{25\beta^2}{2\varepsilon}\right)t\right] + \int_0^t \|f(s)\|_2^2 ds + 2q\epsilon C t,
$$

where $w_0(x,v) = g(x,v) - u_0(x,v)$. This completes the proof for the theorem. ∎

Regarding the derivatives $\partial_t w$ and $\partial_x w$ we can obtain the similar estimates as follows.

COROLLARY 7.1. *Let u and u_{nn} be the classical solutions to (7.48) and (7.49), respectively. Assume that (7.51) holds. Then for $z = t$ or x we have*

$$\begin{aligned}&\sup_{0\le t\le T} \|\partial_z u_{nn}(t) - \partial_z u(t)\|_2^2 + 2(q-\varepsilon)\int_0^T \|\partial_v \partial_z u_{nn}(s) - \partial_v \partial_z u(s)\|_2^2 ds \\ &\le \left(\|\partial_z g - \partial_z u_0\|_2^2 + \frac{L}{2}\right)\exp\left[\left(1 + \frac{25\beta^2}{2\varepsilon}\right)T\right] + \int_0^T \|\partial_z f(s)\|_2^2 ds + 2q\epsilon CT,\end{aligned}$$

for any $\varepsilon \in (0,q)$, where $L, u_0, f, g, \beta, q, m, \epsilon$, and C are given in (7.48)-(7.51).

Proof. For both $\partial_z = \partial_t$ and ∂_x, we take ∂_z onto (7.52) and obtain

$$\begin{aligned}&(\partial_z w)_t + v(\partial_z w)_x - \beta v(\partial_z w)_v - q(\partial_z w)_{vv} = \partial_z f, \\ &\qquad\qquad \text{for } (t,x,v) \in [0,T]\times[0,1]\times[-5,5], \\ &(\partial_z w)(0,x,v) = (\partial_z w)_0, \\ &\qquad\qquad \text{for } (x,v) \in [0,1]\times[-5,5],\end{aligned} \tag{7.53}$$

where $(\partial_z w)_0 \overset{\text{def}}{=} \partial_z g - \partial_z u_0$. Then the proof is the same as the one for Theorem 7.3 with $\partial_z w$ replacing the role of w. This completes the proof. ∎

Finally, we can also obtain the regularity estimates for the derivative $\partial_v w$ as follows:

THEOREM 7.4. *Let u and u_{nn} be the classical solutions to (7.48) and (7.49), respectively. Assume that (7.51) holds. Then we have*

$$\begin{aligned}&\sup_{0\le t\le T} \|\partial_v u_{nn}(t) - \partial_v u(t)\|_2^2 + 2(q-\varepsilon)\int_0^T \|\partial_{vv} u_{nn}(s) - \partial_{vv} u(s)\|_2^2 ds \\ &\le (L + \|\partial_x g - \partial_x u_0\|_2^2 + \|\partial_v g - \partial_v u_0\|_2^2)\exp\left[\left(2 + \frac{25\beta^2}{2\varepsilon}\right)T\right] \\ &\quad + \int_0^T (\|\partial_x f(s)\|_2^2 + \|\partial_v f(s)\|_2^2) ds + 4q\epsilon CT,\end{aligned}$$

for any $\varepsilon \in (0,q)$, where $L, u_0, f, g, \beta, q, m, \epsilon$, and C are given in (7.48)-(7.51).

Proof. We take ∂_v onto (7.52) and obtain

$$\begin{aligned}&(\partial_v w)_t + w_x + v(\partial_v w)_x - \beta v(\partial_v w)_v - q(\partial_v w)_{vv} = \partial_v f, \\ &\qquad\qquad \text{for } (t,x,v) \in [0,T]\times[0,1]\times[-5,5],\end{aligned} \tag{7.54}$$

where $(\partial_v w)_0 \overset{\text{def}}{=} \partial_v g - \partial_v u_0$. By multiplying $\partial_v w$ to (7.54) and integrating with respect to $dxdv$, we have

$$\begin{aligned}&\frac{1}{2}\frac{d}{dt}\iint_{[0,1]\times[-5,5]} |\partial_v w|^2 dxdv + \iint_{[0,1]\times[-5,5]} v(\partial_v w)_x(\partial_v w) dxdv \\ &\quad - \iint_{[0,1]\times[-5,5]} q(\partial_v w)_{vv}\partial_v w dxdv\end{aligned}$$

$$= \iint_{[0,1]\times[-5,5]} (-w_x + \partial_v f)\partial_v w dx dv + \iint_{[0,1]\times[-5,5]} \beta v(\partial_v w)_v \partial_v w dx dv.$$

Then we take the integration by parts and obtain that

$$\begin{aligned}
&\frac{1}{2}\frac{d}{dt}\iint_{[0,1]\times[-5,5]} |w_v|^2 dx dv + \frac{1}{2}\int_{[-5,5]} \left(v(\partial_v w)^2(t,1,v) - v(\partial_v w)^2(t,0,v)\right) dv \\
&\quad + q\iint_{[0,1]\times[-5,5]} |w_{vv}|^2 dx dv \\
&= \iint_{[0,1]\times[-5,5]} \partial_v f w_v dx dv + \iint_{[0,1]\times[-5,5]} \beta v w_{vv} w_v dx dv \\
&\quad + q\int_{[0,1]} w_{vv}(t,x,5) w_v(t,x,5) dx - q\int_{[0,1]} w_{vv}(t,x,-5) w_v(t,x,-5) dx \\
&\quad - \iint_{[0,1]\times[-5,5]} w_x w_v dx dv \stackrel{\text{def}}{=} I_1 + I_2 + I_3 + I_4 + I_5.
\end{aligned}$$

We first define

$$B(t) \stackrel{\text{def}}{=} \left| \frac{1}{2}\int_{-5}^{5} dv\ v(\partial_v w(t,1,v)^2 - \partial_v w(t,0,v)^2) \right|.$$

We now estimate I_1-I_4 on the right-hand side. We now estimate I_1-I_5 on the right-hand side. By the Hölder inequality and Young's inequality, we have

$$|I_1| \le \|\partial_v f\|_2 \|w_v\|_2 \le \frac{1}{2}\|\partial_v f\|_2^2 + \frac{1}{2}\|w_v\|_2^2,$$

where we denote

$$\|h\|_2 \stackrel{\text{def}}{=} \iint_{[0,1]\times[-5,5]} |h|^2 dx dv.$$

Similarly, we observe that

$$|I_2| \le 5\beta \|w_{vv}\|_2 \|w_v\|_2 \le \varepsilon \|w_{vv}\|_2^2 + \frac{25\beta^2}{4\varepsilon}\|w_v\|_2^2,$$

for a sufficiently small $\varepsilon > 0$ as $|v| \le 5$. By (7.50) and (7.51), we have

$$\begin{aligned}
|I_3 + I_4| \le & q\|w_{vv}(t,\cdot,5)\|_{L_x^2} \|w_v(t,\cdot,5) - w_v(t,\cdot,-5)\|_{L_x^2} \\
& + q\|w_{vv}(t,\cdot,5) - w_{vv}(t,\cdot,-5)\|_{L_x^2} \|w_v(t,\cdot,-5)\|_{L_x^2} \le 2q\epsilon C.
\end{aligned}$$

Finally, we have

$$|I_5| \le \|w_x\|_2 \|w_v\|_2 \le \frac{1}{2}\|w_x\|_2^2 + \frac{1}{2}\|w_v\|_2^2.$$

Altogether, we have

$$\frac{d}{dt}\|w_v\|_2^2 + 2(q-\varepsilon)\|w_{vv}\|_2^2 \le \|\partial_v f\|_2 + \|w_x\|_2^2 + \left(2 + \frac{25\beta^2}{2\varepsilon}\right)\|w_v\|_2^2 + \frac{5L}{2} + 2q\epsilon C.$$

Then we take the integration with respect to the temporal variable on $[0,t]$ and obtain that

$$\|w_v(t)\|_2^2+\int_0^t ds\ 2(q-\varepsilon)\|w_{vv}(s)\|_2^2\le\|w_v(0)\|_2^2 + \int_0^t ds\ \left(\|\partial_v f(s)\|_2+\|w_x(s)\|_2^2+\left(2+\frac{25\beta^2}{2\varepsilon}\right)\|w_v(s)\|_2^2+B(s)+2q\epsilon C\right).$$

By $(7.49)_3$, we have

$$\int_0^t ds\ B(s)\le\frac{L}{2}.$$

Thus, by the Grönwall inequality, we have

$$\begin{aligned}&\|w_v(t)\|_2^2+2(q-\varepsilon)\int_0^t\|w_{vv}(s)\|_2^2ds\\ &\le\left(\frac{L}{2}+\|\partial_v w_0\|_2^2\right)\exp\left[\left(2+\frac{25\beta^2}{2\varepsilon}\right)t\right]+\int_0^t(\|w_x(s)\|_2^2+\|\partial_v f(s)\|_2^2)ds+2q\epsilon Ct,\end{aligned}\tag{7.55}$$

where $\partial_v w_0(x,v)=g(x,v)-u_0(x,v)$. Then we use Corollary 7.1 for an upper-bound of $\|\partial_x w(s)\|_2^2$ and obtain that

$$\begin{aligned}&\int_0^T\|w_x(s)\|_2^2ds\\ &\le\left(\frac{L}{2}+\|\partial_x g-\partial_x u_0\|_2^2\right)\exp\left[\left(1+\frac{25\beta^2}{2\varepsilon}\right)T\right]+\int_0^T\|\partial_x f(s)\|_2^2ds+2q\epsilon CT.\end{aligned}$$

Then by (7.55), we obtain that

$$\begin{aligned}&\sup_{0\le t\le T}\|w_v(t)\|_2^2+2(q-\varepsilon)\int_0^T\|w_{vv}(s)\|_2^2ds\\ &\le(L+\|\partial_x w_0\|_2^2+\|\partial_v w_0\|_2^2)\exp\left[\left(2+\frac{25\beta^2}{2\varepsilon}\right)T\right]+\int_0^T(\|\partial_x f(s)\|_2^2+\|\partial_v f(s)\|_2^2)ds+4q\epsilon CT,\end{aligned}$$

where $\partial_v w_0(x,v)=\partial_v g(x,v)-\partial_v u_0(x,v)$. This completes the proof for the theorem. ∎

7.4. The Poisson equation. We consider the Poisson equation with Dirichlet boundary condition:

$$\begin{aligned}-\triangle u&=f\quad\text{in }\Omega,\\ u&=g\quad\text{on }\partial\Omega.\end{aligned}$$

Suppose there exists

$$\tilde g\in H^2(\bar\Omega)\ s.t.\ \tilde g|_{\partial\Omega}=g\tag{7.56}$$

Then, the equation can be written by:

$$\begin{aligned}-\triangle v&=\tilde f\quad\text{in }\Omega,\\ v&=0\quad\text{on }\partial\Omega,\end{aligned}$$

where $v=u-\tilde g,\tilde f=f-\triangle\tilde g$. Therefore, we assume the homogeneous Dirichlet boundary condition provided (7.56).

Now, let u be a strong solution of

$$\begin{aligned} -\triangle u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{7.57}$$

and let u_{nn} be a neural network such that

$$\begin{aligned} -\triangle u_{nn} &= f_{nn} \quad \text{in } \Omega, \\ u_{nn} &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{7.58}$$

Here, we can set the boundary to be zero by multiplying $B(x)$, where $B(x)$ is a smooth function satisfying $B(x) = \begin{cases} 0, & x \in \partial\Omega \\ \neq 0, & x \in \Omega \end{cases}$. By subtracting (7.58) from (7.57), we get

$$\begin{aligned} -\triangle (u - u_{nn}) &= (f - f_{nn}) \quad \text{in } \Omega, \\ u - u_{nn} &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{7.59}$$

Then we apply below theorem to (7.59) to get the convergence results.

THEOREM 7.5 (Theorem 6.3.5 in [10]). *Let m be a nonnegative integer. Suppose that $u \in H_0^1(\Omega)$ is a weak solution of the boundary-value problem (7.57). Assume $\partial\Omega$ is C^{m+2}. Then,*

$$\|u\|_{H^{m+2}(\Omega)} \leq C(\|f\|_{H^m(\Omega)} + \|u\|_{L^2(\Omega)}). \tag{7.60}$$

Furthermore, if u is the unique solution of (7.57), then

$$\|u\|_{H^{m+2}(\Omega)} \leq C\|f\|_{H^m(\Omega)}. \tag{7.61}$$

By applying (7.61) to (7.59), we obtain

$$\|u - u_{nn}\|_{H^{m+2}(\Omega)} \leq C\|f - f_{nn}\|_{H^m(\Omega)}.$$

where the right-hand side corresponds to $Loss_{GE}(u_{nn}; m, 2)$.

Acknowledgments. Hwijae Son is supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-2022R1F1A1073732) and the research fund of Hanbat National University in 2022. Jin Woo Jang is supported by the German Science Foundation (DFG) CRC 1060, by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) NRF-2022R1G1A1009044, and by the Basic Science Research Institute Fund of Korea NRF-2021R1A6A1A10042944. Hyung Ju Hwang is supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2022-00165268, NRF-2019R1A5A1028324).

REFERENCES

[1] C. Basdevant, M. Deville, P. Haldenwang, J.M. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A.T. Patera, *Spectral and finite difference solutions of the Burgers equation*, Comput. Fluids, 14(1):23–41, 1986. 5.2

[2] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, and J.M. Siskind, *Automatic differentiation in machine learning: a survey*, J. Mach. Learn. Res., 18(1):5595–5637, 2017. 5

[3] Y. Benia and B.-K. Sadallah, *Existence of solutions to Burgers equations in domains that can be transformed into rectangles*, Electron. J. Differ. Equ., 157:13, 2016. 7.2, 7.1

[4] J. Berg and K. Nyström, *A unified deep artificial neural network approach to partial differential equations in complex geometries*, Neurocomputing, 317:28–41, 2018. 1, 3.1
[5] R. Bischof and M. Kraus, *Multi-objective loss balancing for physics-informed deep learning*, arXiv preprint, arXiv:2110.09813, 2021. 2
[6] H. Brezis and H. Brézis, *Functional Analysis, Sobolev Spaces and Partial Differential Equations*, Springer, 2, 2011. 7.2
[7] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Math. Control Signals Syst., 2(4):303–314, 1989. 1
[8] W.M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu, *Sobolev training for neural networks*, Adv. Neur. Inf. Process. Syst., 4278–4287, 2017. 1, 2, 5.6
[9] T. De Ryck and S. Mishra, *Error analysis for deep neural network approximations of parametric hyperbolic conservation laws*, arXiv preprint, arXiv:2207.07362, 2022. 2
[10] L.C. Evans, *Partial Differential Equations*, Amer. Math. Soc., Providence, R.I., 2010. 7.1, 7.2, 7.5
[11] J. Han, A. Jentzen, and E. W., *Solving high-dimensional partial differential equations using deep learning*, Proc. Natl. Acad. Sci., 115(34):8505–8510, 2018. 2
[12] K. Hornik, M. Stinchcombe, and H. White, *Multilayer feedforward networks are universal approximators*, Neural Netw., 2(5):359–366, 1989. 1
[13] J-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon, *Learning neural PDE solvers with convergence guarantees*, arXiv preprint, arXiv:1906.01200, 2019. 2, 5.4
[14] H.J. Hwang, J.W. Jang, H. Jo, and J.Y. Lee, *Trend to equilibrium for the kinetic Fokker-Planck equation via the neural network approach*, J. Comput. Phys., 419:109665, 2020. 1, 3, 3.1, 5.3
[15] H.J. Hwang, J. Jang, and J. Jung, *The Fokker-Planck equation with absorbing boundary conditions in bounded domains*, SIAM J. Math. Anal., 50(2):2194–2232, 2018. 4.2
[16] H.J. Hwang, J. Jang, and J.J.L. Velázquez, *On the structure of the singular set for the kinetic Fokker-Planck equations in domains with boundaries*, Quart. Appl. Math., 77(1):19–70, 2019. 4.2
[17] H.J. Hwang, C. Kim, M.S. Park, and H. Son, *The deep minimizing movement scheme*, arXiv preprint, arXiv:2109.14851, 2021. 1
[18] H.J. Hwang and H. Son, *Lagrangian dual framework for conservative neural network solutions of kinetic equations*, Kinet. Relat. Models, 15(4):551–568, 2022. 1
[19] H. Jo, H. Son, H.J. Hwang, and E.H. Kim, *Deep neural network approach to forward-inverse problems*, Netw. Heterog. Media, 15(2):247–259, 2020. 1, 3
[20] D.P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint, arXiv:1412.6980, 2014. 5
[21] I.E. Lagaris, A. Likas, and D.I. Fotiadis, *Artificial neural networks for solving ordinary and partial differential equations*, IEEE Trans. Neural Netw., 9(5):987–1000, 1998. 1, 2
[22] I.E. Lagaris, A.C. Likas, and D.G. Papageorgiou, *Neural-network methods for boundary value problems with irregular boundaries*, IEEE Trans. Neural Netw., 11(5):1041–1049, 2000. 1, 2
[23] X. Li, *Simultaneous approximations of multivariate functions and their derivatives by neural networks with one hidden layer*, Neurocomputing, 12(4):327–343, 1996. 1
[24] Z. Long, Y. Lu, X. Ma, and B. Dong, *PDE-Net: Learning PDEs from data*, Int. Conf. Mach. Learn., 80:3208–3216, 2018. 2
[25] L. McClenny and U. Braga-Neto, *Self-adaptive physics-informed neural networks using a soft attention mechanism*, arXiv preprint, arXiv:2009.04544, 2020. 2
[26] S. Mishra and R. Molinaro, *Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs*, IMA J. Numer. Anal., 42(2):981–1022, 2022. 2
[27] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., *Pytorch: An imperative style, high-performance deep learning library*, Adv. Neur. Inf. Process. Syst., 8026–8037, 2019. 5
[28] M. Raissi, P. Perdikaris, and G. Em Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, J. Comput. Phys., 378:686–707, 2019. 1, 2, 3.1, 5.1
[29] F.M. Rohrhofer, S. Posch, and B.C. Geiger, *On the Pareto front of physics-informed neural networks*, arXiv preprint, arXiv:2105.00862, 2021. 2
[30] Y. Shin, J. Darbon, and G. Em Karniadakis, *On the convergence and generalization of physics informed neural networks*, Commun. Comput. Phys., 28:2042–2074, 2020. 2
[31] J. Sirignano and K. Spiliopoulos, *DGM: A deep learning algorithm for solving partial differential equations*, J. Comput. Phys., 375:1339–1364, 2018. 1, 2, 3.1, 5.2, 5.4
[32] H. Son, S.W. Cho, and H.J. Hwang, *AL-PINNs: Augmented Lagrangian relaxation method for physics-informed neural networks*, arXiv preprint, arXiv:2205.01059, 2022. 2

[33] R. van der Meer, C. Oosterlee, and A. Borovykh, *Optimally weighted loss functions for solving PDEs with neural networks*, J. Comput. Appl. Math., 405(15):113887, 2022. 2

[34] S. Wang, Y. Teng, and P. Perdikaris, *Understanding and mitigating gradient flow pathologies in physics-informed neural networks*, SIAM J. Sci. Comput., 43(5):A3055–A3081, 2021. 2

[35] S. Wang, X. Yu, and P. Perdikaris, *When and why PINNs fail to train: A neural tangent kernel perspective*, J. Comput. Phys., 449:110768, 2022. 2

[36] E. W., J. Han, and A. Jentzen, *Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations*, Commun. Math. Statist., 5(4):349–380, 2017. 2

[37] E. W. and B. Yu, *The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems*, Commun. Math. Statist., 6(1):1–12, 2018. 5.4

[38] S. Wollman and E. Ozizmir, *A deterministic particle method for the Vlasov–Fokker–Planck equation in one dimension*, J. Comput. Appl. Math., 213(2):316–365, 2008. 5, 5.3

[39] Y. Zang, G. Bao, X. Ye, and H. Zhou, *Weak adversarial networks for high-dimensional partial differential equations*, J. Comput. Phys., 411:109409, 2020. 5.4