NUMERICAL ANALYSIS FOR CONVERGENCE OF A SAMPLE-WISE BACKPROPAGATION METHOD FOR TRAINING STOCHASTIC NEURAL NETWORKS

RICHARD ARCHIBALD*, FENG BAO†, YANZHAO CAO ‡, AND HUI SUN §

Abstract. The aim of this paper is to carry out convergence analysis and algorithm implementation of a novel sample-wise backpropagation method for training a class of stochastic neural networks (SNNs). The preliminary discussion on such an SNN framework was first introduced in [2]. The structure of the SNN is formulated as a discretization of a stochastic differential equation (SDE). A stochastic optimal control framework is introduced to model the training procedure, and a sample-wise approximation scheme for the adjoint backward SDE is applied to improve the efficiency of the stochastic optimal control solver, which is equivalent to the backpropagation for training the SNN. The convergence analysis is derived by introducing a novel joint conditional expectation for the gradient process. Under the convexity assumption, our result indicates that the number of SNN training steps should be proportional to the square of the number of layers in the convex optimization case. In the implementation of the sample-based SNN algorithm with the benchmark MNIST data set, we adopt the convolution neural network (CNN) architecture and demonstrated that our sample-based SNN algorithm is more robust than the conventional CNN.

Keywords. Probabilistic learning, stochastic neural networks, convergence analysis, backward stochastic differential equations, stochastic gradient descent

AMS subject classifications.

1. Introduction.

Deep neural network (DNN) based machine learning techniques are positioned to fundamentally change many sectors of society by offering decision making capabilities, which match and often exceed that of human experts [1, 14, 29–32, 35]. Despite of dramatic success that DNNs achieved, a closer examination reveals inherent challenges of applying such approaches broadly to science and engineering. A major challenge is that DNN models are sensitive to noises in data. A recent study [4] shows that effectiveness of different DNN models are affected by inaccurate data in the datasets, and science and engineering solutions typically need to be able to quantify the uncertainty of DNN outputs and to provide effective operating ranges and risks.

One successful effort to address the challenge of uncertainty quantification for DNN is probabilistic learning, which aims to incorporate randomness to DNN models and produce random output. Then, one can use the statistical behaviors of DNN's random output to study the uncertainty of data. The state-of-the-art probabilistic learning approach is the Bayesian neural network (BNN) [10,11,18,24,27,34,36,37], which treats parameters in a DNN as random variables and approximates their distributions through Bayesian inference. Instead of searching for the optimal parameters by deterministic optimization, the BNN utilizes Bayesian optimization to derive parameter distributions. As a result, the estimated random parameters generate random output, which characterizes the target model's uncertainty. Although the BNN approach provides a principled conceptual framework to quantify the uncertainty in probabilistic machine learning, carrying out Bayesian optimization to estimate a massive number of parameters in the BNN often renders the exact inference of parameter posteriors intractable. Moreover, the large size of the parameter set leads to uninterpretable parameter priors. Some of the existing Non-Bayesian methods, such as [9, 19], are simple to implement but either ad hoc or computationally prohibitive.

In this work, we consider a type of stochastic neural networks that models uncertainty in some of the well known DNN structures (e.g., residual neural networks, convolutional neural networks) through discretized stochastic (ordinary) differential equations (SDEs). Recently, an ordinary differential equation (ODE) interpretation for DNNs (named Neural-ODE) has been introduced and studied [5–7,12,13]. The central idea of the Neural-ODE is to formulate the evolution of potentially huge hidden layers in the DNN as a discretized ODE system. To characterize the randomness caused by the uncertainty of models and noises of data, we add an additive Brownian motion noise to the ODE to count for the

* Computational Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, Tennessee (archibaldrk@ornl.gov);

† Department of Mathematics, Florida State University, Tallahassee, Florida, (bao@math.fsu.edu);

‡ Department of Mathematics, Auburn University, Auburn, Alabama, (yzc0009@auburn.edu)

§ Department of Mathematics, Florida State University, Tallahassee, Florida;

model uncertainty of the DNN. This changes the ODE to an SDE, and the deterministic DNN becomes a stochastic neural network (SNN) [17, 22, 33]. In the SNN model, the drift parameters serve as the prediction of the network, and the stochastic diffusion governs the randomness of network output, which serves to quantify the uncertainty of deep learning. In comparison with the BNN, which needs to calibrate a tremendous amount of random parameters through Bayesian inference, the SNN only uses one noise term (i.e., the diffusion term) to control the amount of uncertainty at each layer. The computational cost for evaluating the diffusion coefficient is much lower than high-dimensional Bayesian inference for many unknown random parameters. On the other hand, for an SNN with multiple hidden layers, by stacking (controlled) diffusion terms together through the multi-layer structure, we would still be able to characterize sufficient probabilistic behavior of the neural network.

The bottleneck of the SNN is constructing an efficient numerical solver for the backpropagation. This process propagates the total loss back into the neural network and updates the parameters accordingly to minimize the loss. It is the essence of neural network training. For Neural-ODE, backpropagation is equivalent to solving the adjoint of the ODE, which is a backward ODE. Since its structure is no different from that of a forward ODE, backpropagation can be achieved by means of a deterministic adjoint ODE solver [5]. For SNNs, the backpropagation is equivalent to solving the adjoint of the SDE, which is a backward SDE. Because of the martingale nature of the backward SDE [8], one must create a separate backward SDE solver, which is completely different from and far more computationally intensive than forward SDE solvers. Thus the backpropagation by completely solving the backward SDE has been deemed unfeasible [33]. Several alternatives have been proposed [17, 22, 33]. But they either fail to consider the Ito's nature of stochastic differentiation or lack scalability. Thus these training frameworks have been deemed inefficient [21].

In a recent study [2], we formulated the training procedure for SNN as a stochastic optimal control problem. The central idea is to treat the random samples of the backward SDE as "pseudo data", and only solve the backward SDE partially on randomly selected pseudo data. In this way, only a tiny fraction of the computing cost of solving the entire backward SDE is required to complete one training iteration. Therefore, our novel sample-wise backpropagation method for SNNs will be a feasible tool for the uncertainty quantification of deep learning. In [2], following a standard analysis for SGD, the convergence of the continuous SNN model was proved before the sample-wised backpropagation algorithm was proved. In other words, no numerical analysis was provided. Also, the algorithm was tested only for simple one-dimensional problems.

The contribution of this work has a numerical analysis aspect and an algorithm implementation aspect. On the numerical analysis aspect, we will focus on deriving the convergence and an error estimate for the sample-wise backpropagation algorithm. For a conventional SGD, it is assumed that the adjoint equation, which is a backward SDE, is solved in the entire state space [39]. The key challenge in the convergence analysis of our sample-wise backpropagation method is that our sample-wise approximation only solves the backward SDE at a sample point in each SDG iteration, which does not carry enough information to deduce convergence of the numerical solver for the backward SDE with a standard argument for SGD convergence. Our strategy for addressing this issue is to introduce an augmented σ-algebra that contains the uncertainty information in both the SNN model and the training data. We shall show that the stochastic approximation for the gradient conditioning on the augmented σ-algebra is an unbiased estimator. This enables us to obtain the desired convergence results without the convergence of numerical solvers for backward SDEs. Under the convexity assumption, we show that the error estimate of the sample-wise backpropagation algorithm contains two terms: the first is a half-order term with respect to the depth of neural networks; the second is a quotient between the depth of SNN and the number of training steps. While the first term reflects the error of discretizing the continuous SDE formulation of probabilistic learning, the second term provides an inherent relation between the depth of a neural network and the number of iterations needed in the training procedure, which is a novelty of our numerical analysis. Without the convexity assumption, we will show that, for an SNN with a fixed depth, our sample-wise backpropagation algorithm converges with respect to training steps.

It is well known that implementation plays a key role in building a practical deep neural network. Our implementation effort for the SNN involves the selection of both activation functions and network

structures. In order for the SNN to solve high-dimensional problems, we will use a combined sigmoid activation function in place of a single sigmoid activation function. This allows the forward propagation procedure to produce rich activation behavior, which will improve the representation capability of SNN. Moreover, to implement the SNN in accomplishing benchmark machine learning tasks, we will incorporate the convolution blocks [20] into the SNN model. With such an implementation we will solve the benchmark classification problem with MNIST handwritten dataset and Fashion-MNIST dataset. This benchmark problem demonstrates the robustness advantage of SNN compared to the deterministic convolution neural network. We want to mention that other DNN structures such as the recurrent neural network can also be adopted in our algorithm similarly to solve machine learning problems in different scenarios.

The rest of this paper is organized as follows: In Section 2, we formulate the training procedure for SNNs as a stochastic optimal control problem and introduce our sample-wise backpropagation method. The main convergence theorems and their proofs will be provided in Section 3. In Section 4, we will validate our analysis results and examine the performance of SNNs through several numerical experiments.

2. A sample-wise backpropagation method for stochastic neural networks. In this section, we briefly review the sample-wise backpropagation method for training stochastic neural networks introduced in [2]. The basic idea is to first formulate the training procedure for the continuous formulation of a stochastic neural network as a stochastic optimal control problem, then solve the stochastic optimal control problem via a generalized stochastic gradient descent algorithm.

Consider the following dynamical system for a stochastic version of a deep neural network (DNN):

$$X_{n+1} = X_n + hf(X_n, u_n) + \sqrt{h}g(u_n)\omega_n, \qquad n = 0, 1, 2, \cdots, N-1, \tag{2.1}$$

where $X_n := [x_n^1, x_n^2, \cdots, x_n^L] \in \mathbb{R}^L$ is a vector containing L neurons at the n-th layer in a DNN, f is an activation function, u_n denotes the set of DNN parameters at the n-th layer, h is a positive constant that stabilizes the DNN, ω_n is a standard L-dimensional Gaussian random variable that counts for uncertainty in the neural network, and g is a coefficient function that determines the size of uncertainty in the DNN. The initial state X_0 of the dynamical system (2.1) represents the input, and X_N is the output. In this paper, we call the noise perturbed DNN model (2.1) a stochastic neural network (SNN).

If we choose a positive constant T (as a terminal time) and let $N \to \infty$ or $h \to 0$ with $h = \frac{T}{N}$, the dynamics of SNN (2.1) becomes the stochastic differential equation (SDE) which, in the integral form, is given by

$$X_T = X_0 + \int_0^T f(X_t, u_t)dt + \int_0^T g(u_t)dW_t, \tag{2.2}$$

where $W := \{W_t\}_{0 \le t \le T}$ is a standard Brownian motion corresponding to the i.i.d. Gaussian random variable sequence $\{w_n\}_n$ in (2.1), $\int_0^T g(u_t)dW_t$ is an Itô integral, and X_T corresponds to the output.

Let Γ be the random variable that generates the training data to be compared with X_T and define a loss function $\Phi(X_T, \Gamma) := \|X_T - \Gamma\|_{loss}$ corresponding to a loss error norm $\|\cdot\|_{loss}$. In this paper, we shall treat the training procedure for deep learning as a stochastic optimal control problem and the parameter u_t in (2.2) as a control process. To this end, we define the cost functional J as

$$J(u) = \mathbb{E}\left[\int_0^T r(X_t, u_t)dt + \Phi(X_T, \Gamma)\right], \tag{2.3}$$

where the integral $\int_0^T r(X_t, u_t)dt$ represents the running cost in a control problem. This way, the loss function becomes the terminal cost in the stochastic optimal control problem. Note that the stochastic process X is the "state process" for the stochastic optimal control problem, which depends on the control u. The goal of deep learning is to solve the stochastic optimal control problem, i.e., find the optimal control u^* such that

$$J(u^*) = \inf_{u \in \mathcal{U}[0,T]} J(u), \tag{2.4}$$

where $\mathcal{U}[0,T] := \{u \in L^2([0,T];\mathbb{R}^p)\}$ is a p-dimensional admissible control set.

To determine the optimal control, one can derive the following gradient process [23, 38]:

$$\nabla J_u(u_t) = \mathbb{E}\big[f_u(X_t,u_t)^\top Y_t + g_u(u_t)^\top Z_t + r_u(X_t,u_t)^\top\big], \tag{2.5}$$

where f_u, g_u and r_u are partial derivatives with respect to the control u. Here the stochastic processes Y_t and Z_t in (2.5) are the adapted solutions of the following backward SDE:

$$dY_t = \big(-f_x(X_t,u_t)^\top Y_t - r_x(X_t,u_t)^\top\big)\,dt + Z_t dW_t, \qquad Y_T = \Phi_x'(X_T,\Gamma), \tag{2.6}$$

where f_x, r_x are partial derivatives with respect to the state X, Y is the adjoint process of the state SDE X, and Z is the martingale representation of Y with respect to W. An important property of solutions of the backward SDE is that values of Y and Z depend on X, and $(Y_t,Z_t) \in \mathcal{F}_t^W$, where $\mathcal{F}_t^W := \sigma(W_s, 0 \le s \le t)$. In this paper, we use the notation ∇J_u to denote the gradient of the cost J with respect to the control process u, and the control at time t, i.e., u_t, in $\nabla J_u(\cdot)$ indicates that we are considering the gradient process at time t.

Having the gradient of J in hand, one can carry out gradient descent optimization to determine the optimal control as follows:

$$u_t^{k+1} = \mathcal{P}_{\mathcal{U}}\big(u_t^k - \eta_k \nabla J_u(u_t^k)\big), \qquad k = 0,1,2,\cdots, \quad 0 \le t \le T, \tag{2.7}$$

where u^0 is an initial guess for the optimal control, η_k is the step-size of gradient descent in the k-th iteration step, and $\mathcal{P}_{\mathcal{U}}$ is a projection operator onto the admissible control set $\mathcal{U}$. For the stochastic gradient descent method, one chooses samples X_t^k, Z_t^k of X_t and Z_t and modifies (2.7) as follows.

$$u_t^{k+1} = \mathcal{P}_{\mathcal{U}}\big(u_t^k - \eta_k\big[f_u(X_t^k,u_t^k)^\top Y_t + g_u(u_t^k)^\top Z_t^k + r_u(X_t^k,u_t^k)^\top\big]\big), \qquad k = 0,1,2,\cdots, \quad 0 \le t \le T. \tag{2.8}$$

In [2], following a standard procedure [3], we proved the convergence of the SGD iteration (2.8) under certain smoothness assumptions on the input functions.

Numerical implementation of the stochastic gradient descent scheme (2.8) requires numerical approximations to the SDE (2.2) and backward SDE (2.6). We solve the backward SDE (2.6) over a uniform temporal partition $\Pi_N := \{0 = t_0 < t_1 < t_2 < \cdots < t_N = T\}$. Denote by h the stepsize of the partition. We adopt the classic numerical schemes for solving forward and backward SDEs (see [16, 39, 40])

$$X_{n+1}^N = X_n^N + hf(X_n^N,u_{t_n}) + g(u_{t_n})\Delta W_{t_n}, \tag{2.9}$$

$$Y_n^N = \mathbb{E}_n^X[Y_{n+1}^N] + h\mathbb{E}_n^X\big[f_x(X_{n+1}^N,u_{t_{n+1}})^\top Y_{n+1}^N + r_x(X_{n+1}^N,u_{t_{n+1}})^\top\big], \tag{2.10}$$

$$Z_n^N = \mathbb{E}_n^X\Big[\frac{Y_{n+1}^N \Delta W_{t_n}}{h}\Big], \tag{2.11}$$

where X_{n+1}^N, Y_n^N and Z_n^N are numerical approximations for $X_{t_{n+1}}$, Y_{t_n} and Z_{t_n}, respectively.

To approximate the stochastic gradient of J appeared in (2.8) with schemes (2.9)–(2.11), we need to evaluate the conditional expectation $\mathbb{E}_n^X[\cdot]$ appeared in (2.10) and (2.11) which typically involved high dimensional integrations. This is a very challenging task due to the "curse of dimensionality". To address this challenge, we adopt the stochastic approximation method to approximate gradient ∇J_u. Specifically, at the k-th iteration with an estimated optimal control u^k, we simulate the state process by choosing a sample $\omega^k \sim N(0,h)$ and evaluate $\mathbb{E}_n^X[\cdot]$ only at this sample point. Then the sample-wise numerical solutions X_n^k of X, and (Y_n^k,Z_n^k) of (Y,Z) are given by

$$X_{n+1}^k = X_n^k + hf(X_n^k,u_{t_n}^k) + g(u_{t_n}^k)\omega_n^k, \tag{2.12}$$

$$Y_n^k = Y_{n+1}^k + h\big(f_x(X_{n+1}^k,u_{t_{n+1}}^k)^\top Y_{n+1}^k + r_x(X_{n+1}^k,u_{t_{n+1}}^k)^\top\big), \qquad Z_n^k = \frac{Y_{n+1}^k \omega_n^k}{h}. \tag{2.13}$$

With (2.12) and (2.13), we approximate the gradient ∇J_u by

$$\nabla j_u^k(u_n^k) := f_u(X_n^k,u_n^k)^\top Y_n^k + g_u(u_n^k)^\top Z_n^k + r_u(X_n^k,u_n^k)^\top, \tag{2.14}$$

and derive the sample-wise stochastic gradient descent (SGD) scheme as follows.

$$u_{t_n}^{k+1} = \mathcal{P}_{\mathcal{U}_N}\left(u_{t_n}^k - \eta_k \nabla j_u^k(u_{t_n}^k)\right), \qquad k=0,1,2,\cdots, \quad 0\le n\le N, \tag{2.15}$$

where $\mathcal{U}_N := \mathcal{U}\cap\mathcal{C}_N$ with a piece-wise constant approximation set $\mathcal{C}_N := \{u | u = \sum_{n=0}^{N-1} a_n 1_{[t_n,t_{n+1})}, a_n \in \mathbb{R}^p\}$ for the control process. The above schemes (2.12) - (2.15) provide a modified SGD algorithm for solving the stochastic optimal control problem and constitute a sample-wise backpropagation framework for training the SNN. It's worth mentioning that in machine learning practice, the above sample-wise backpropagation scheme is often implemented by using a mini-batch of samples instead of a single-realization of the sample to represent the state variable.

3. Convergence Analysis. In this section, we conduct the convergence analysis for the SGD algorithm (2.12) - (2.15), and we have the following standard assumptions for backward SDEs and the stochastic optimal control problem (see page 38: HG(ii) in [26], page 26: Eq.(1.24) in [26], and Assumption 4.0.1 in [39]):

ASSUMPTION 3.1.

(a) both f and g are deterministic and $f\in C_b^{2,2}(\mathbb{R}^d\times\mathbb{R}^m;\mathbb{R}^d)$ and $g\in C_b^2(\mathbb{R}^m;\mathbb{R}^d)$.

(b) f, f_x, f_u, g, r_x, r_u are all uniformly lipschitz in x,u and uniformly bounded.

(c) g satisfies the uniform elliptic condition.

(d) The initial condition $X_0\in L^2(\mathcal{F}_0)$.

(e) The terminal (Loss) function Φ is C^1 and positive, and Φ_x has at most linear growth at infinity.

(f) $\lim_{||u||_2\to\infty} J(u) = \infty$.

We define the standard inner product of $u, v\in L^2([0,T])$ by $\langle u,v\rangle = \int_0^T u_t\cdot v_t dt$ and the standard L^2 norm $\|\cdot\|_2$ by $\|u\|_2 := \sqrt{\langle u,u\rangle}$. When there is no danger of ambiguity, we use the same notation to denote the inner product of two piece-wise constant representations u^N, $v^N\in\mathbb{R}^N$ of u, v: $\langle u^N, v^N\rangle = h\sum_{n=0}^{N-1} u_n^N\cdot v_n^N$. Also, we use $|\cdot|$ to denote the Euclidean norm or Frobenius norm.

Our analysis in this paper focuses on the convergence of the sample-wise SGD iteration (2.15) with respect to the temporal partition Π_N. Under the assumption that the cost functional for the optimal control is convex, we will derive a half-order convergence rate for our algorithm in the mean square sense. Without the convexity assumption, we will prove the convergence of iteration (2.15). We remark that convergence analysis regarding the number of neurons, i.e., the dimension of the state X in SDE (2.2), may require more complicated discussions on spatial dimension approximation and representability of neural networks, which is out of the scope of this paper.

3.1. Sample-wise numerical solution of the BSDE as an unbiased estimation. The foundation of the convergence analysis is based on the fact that the sample-wise solutions Y_n^k and Z_n^k introduced in (2.13) are equivalent to the classic numerical solutions Y_n^N and Z_n^N introduced in (2.10)–(2.11) under conditional expectation $\mathbb{E}_n^X[\cdot]$. Specifically, we have the following proposition.

PROPOSITION 3.2. *For given estimated control $u^k\in\mathcal{U}_N$, let $Y_n^{k,N}$ and $Z_n^{k,N}$ be the numerical solutions defined in* (2.10) *driven by u^k. Then the following identities hold:*

$$\mathbb{E}_n^X[Y_n^k] = Y_n^{k,N}|_{X_n=X}, \qquad \mathbb{E}_n^X[Z_n^k] = Z_n^{k,N}|_{X_n=X}, \qquad 0\le n\le N-1, \tag{3.1}$$

and therefore we have $\mathbb{E}[Y_n^k] = \mathbb{E}[Y_n^{k,N}]$ and $\mathbb{E}[Z_n^k] = \mathbb{E}[Z_n^{k,N}]$.

Proof. Note that the random variable ω_n^k in scheme (2.12) has the same distribution of ΔW_{t_n} appeared in (2.9). Hence, the random variable ω_n^k is equivalent to ΔW_{t_n} under expectation. More generally, for any function $\phi(\{\omega_n^k\}_n)$ of the random sample path $\{\omega_n^k\}_n$, we have $\mathbb{E}[\phi(\{\omega_n^k\}_n)] = \mathbb{E}[\phi(\{\Delta W_{t_n}\}_n)]$.

To obtain the desired results in the proposition, we first consider the case $n=N-1$ (i.e. take one step back from the terminal time), and we have

$$\mathbb{E}_{N-1}^X[Z_{N-1}^k] = \mathbb{E}_{N-1}^X\left[\frac{Y_N^k\omega_{N-1}^k}{h}\right] = \mathbb{E}_{N-1}^X\left[\frac{Y_N^{k,N}\Delta W_{t_{N-1}}}{h}\right] = Z_{N-1}^{k,N}|_{X_{N-1}=X}.$$

Following the same argument, we also have

$$\mathbb{E}^X_{N-1}[Y^k_{N-1}] = \mathbb{E}^X_{N-1}\left[Y^k_N + h\big(f_x(X^k_N, u^k_{t_N})^\top Y^k_N + r_x(X^k_N, u^k_{t_N})^\top\big)\right].$$

Since $Y^k_N = \Phi_x = Y^{k,N}_N$ and $\mathbb{E}^X_{N-1}[X^k_N] = \mathbb{E}^X_{N-1}[X^{k,N}_N]$, where $X^{k,N}_N$ is the approximated solution introduced in (2.9) with the given control u^k, the above equation becomes

$$\mathbb{E}^X_{N-1}[Y^k_{N-1}] = \mathbb{E}^X_{N-1}\left[Y^{k,N}_N + h\big(f_x(X^{k,N}_N, u^k_{t_N})^\top Y^{k,N}_N + r_x(X^{k,N}_N, u^k_{t_N})^\top\big)\right] = Y^{k,N}_{N-1}|_{x_{N-1}=X}. \tag{3.2}$$

Then, by repeatedly applying the equality (3.2) and the tower property, we obtain the desired result (3.1). □

Next, we show that the gradient of the cost functional J has a similar unbiased property at random samples. To this end, let $\mathcal{G}_k := \sigma(\omega^i, \gamma^i, 0 \le I \le k-1)$ be an augmented σ-algebra generated by the Gaussian random variables ω^i, which we use to generate state sample path X^k in the sample-wise scheme (2.12), and the data sample γ^i generated by the training data Γ. Based on the above proposition, we see that the stochastic approximation $\nabla j^k_u(u^k_{t_n})$ introduced in (2.14) is an unbiased estimator for the gradient $\nabla J^N_u(u^k_{t_n})$ given $\mathcal{G}_k$, i.e. $\mathbb{E}[\nabla j^k_u(u^k_{t_n})\big|\mathcal{G}_k] = \nabla J^N_u(u^k_{t_n})$. From scheme (2.15), we observe that the estimated optimal control u^k is $\mathcal{G}_k$ measurable. Denote $\mathbb{E}^k[\cdot] := \mathbb{E}[\cdot\big|\mathcal{G}_k]$ in the rest of this paper for convenience of presentation. Following a similar argument in the proof of Proposition 3.2, we have

$$\begin{aligned}
\mathbb{E}^k[\nabla j^k_u(u^k_{t_n})] &= \mathbb{E}^k\left[f_u(X^k_n, u^k_n)^\top Y^k_n + g_u(u^k_{t_n})^\top Z^k_n + r_u(X^k_n, u^k_{t_n})^\top\right] \\
&= \mathbb{E}^k\left[f_u(X^{k,N}_n, u^k_{t_n})^\top Y^{k,N}_n + g_u(u^k_{t_n})^\top Z^{k,N}_n + r_u(X^{k,N}_n, u^k_{t_n})^\top\right] \\
&= \nabla J^N_u(u^k_{t_n}).
\end{aligned}$$

The following lemma is about the boundedness of the sample-wise solution Y^k and the linear growth property for Z^k with any approximate control $u^k \in \mathcal{U}_N$.

LEMMA 3.3. *Under Assumptions (a)-(e), for any $u^k \in \mathcal{U}_N$, we have*

$$\sup_{0\le n\le N} \mathbb{E}[(Y^k_n)^2] \le C, \qquad \sup_{0\le n\le N} \mathbb{E}[(Z^k_n)^2] \le CN$$

for some positive constant C.

Proof. We square both sides of the scheme for Y^k_n defined in (2.13) and then take expectation to obtain

$$\begin{aligned}
\mathbb{E}[(Y^k_n)^2] &\le \mathbb{E}\left[(1+h)(1+hf_x(X^k_{n+1}, u^k_{t_{n+1}})^\top)(Y^k_{n+1})^2\right] + (1+\frac{1}{h})\mathbb{E}\left[(r_x(X^k_{n+1}, u^k_{t_{n+1}})^\top)^2 h^2\right] \\
&\le (1+Ch)\mathbb{E}\left[(Y^k_{n+1})^2\right] + Ch,
\end{aligned}$$

where we have used Young's inequality. Then, by the discrete Gronwall's inequality, we have

$$\sup_{0\le n\le N} \mathbb{E}[(Y^k_n)^2] \le C\mathbb{E}[(Y^k_N)^2] + C.$$

Due to Assumption 3.1 (e) for the loss function Φ, we have that $\sup_{0\le n\le N} \mathbb{E}[(Y^k_n)^2] \le C$ as desired.

Next, we square both sides of the scheme for Z^k_n and take the expectation. Based on the boundedness for Y^k_n, we have that

$$\mathbb{E}[(Z^k_n)^2] \le \mathbb{E}\left[\left(\frac{Y^k_n \omega^k_n}{h}\right)^2\right] \le CN,$$

which is the desired boundedness for Z^k_n. □

In addition, we have the following classic boundedness property for solutions of backward SDEs (see Theorem 4.2.1 in [39]):

$$\sup_{0\le t\le T} \mathbb{E}[|Y_t|^2] + \mathbb{E}\left[\int_0^T |Z_t|^2 dt\right] \le C, \tag{3.3}$$

which will provide the following boundedness result

$$\sup_{0\le t\le T}\mathbb{E}[|Y_t^k|^2]+\mathbb{E}\left[\int_0^T|Z_t^k|^2dt\right]\le C \tag{3.4}$$

for the analytic solutions Y_t^k and Z_t^k driven by a control $u^k\in\mathcal{U}_N$, where the constant C is independent of u^k. Following the above estimates, one can also show that $\sup_{0\le t\le T}\mathbb{E}[|Z_t^k|^2]\le C$ under Assumption 3.1 (a)-(e).

Note that the boundedness of the sample-wise solution Z_n^k in the estimate of Lemma 3.3 is not as strong as the true solution Z_t^k due to the loss of regularity caused by the sample-wise representation of expectation introduced in (2.13).

The next proposition is about the convergence of the numerical solutions to backward SDEs introduced in (2.10) also holds:

PROPOSITION 3.4. *Under Assumption 3.1, for small enough step-size h, there exists $C>0$ independent of $u^k\in\mathcal{U}_N$ such that*

$$\max_{0\le n\le N}\mathbb{E}\left[\sup_{t_n}|Y_t^k-Y_n^{k,N}|^2\right]+\sum_{n=1}^{N-1}\mathbb{E}\left[\int_{t_n}^{t_{n+1}}|Z_t^k-Z_n^{k,N}|^2dt\right]\le C(1+|X_0|^2)h. \tag{3.5}$$

Proof. The inequality (3.5) is a standard extension of the regularity property, and the proof of the theorem can be derived following the proof for Theorem 5.3.3 in [39]. The fact that C does not depend on u^k is due to the uniform boundedness and the Lipschitz assumptions of f, g and r. □

The boundedness property of solutions of backward SDEs gives us the boundedness of ∇J_u. Also, the convergence result (3.5) gives the boundedness of $(Y_n^{k,N},Z_n^{k,N})$, which makes ∇J_u^N bounded, i.e.

$$\sup_{u^k\in\mathcal{U}_N}\|\nabla J_u^N(u^k)\|_2<C. \tag{3.6}$$

As a consequence of the above discussions, we have the following lemma.

LEMMA 3.5. *Under Assumption 3.1, for any piece-wise constant estimated control $u^k\in\mathcal{U}_N$, the following estimation holds*

$$\mathbb{E}\left[\|\nabla j_u^k(u^k)-\nabla J_u^N(u^k)\|_2^2\right]\le CN. \tag{3.7}$$

Proof. Due to Lemma 3.3 and the boundedness assumptions for f_u, g_u and r_u, we have that

$$|\nabla j_u^k(u_{t_n}^k)|^2\le C\left(|Y_n^k|^2+|Z_n^k|\right)\le CN. \tag{3.8}$$

Then, we can obtain

$$\begin{aligned}
&\mathbb{E}\left[\|\nabla j_u^k(u^k)-\nabla J_u^N(u^k)\|_2^2\right]\le 2\mathbb{E}\left[\|\nabla j_u^k(u^k)\|_2^2\right]+2\mathbb{E}\left[\|\nabla J_u^N(u^k)\|_2^2\right]\\
\le\ & CN+Ch\sum_{n=0}^{N-1}\mathbb{E}\left[|f_u(X_n^{k,N},u_{t_n}^k)^\top Y_n^{k,N}|^2+|g_u(X_n^{k,N},u_{t_n}^k)^\top Z_n^{k,N}|^2+|r_u(X_n^{k,N},u_{t_n}^k)^\top|^2\right]\\
\le\ & CN+Ch\sum_{n=0}^{N-1}\sup_{0\le n\le N-1}\mathbb{E}\left[|Y_n^{k,N}|^2+|Z_n^{k,N}|^2\right]+C\\
\le\ & CN+C,
\end{aligned}$$

where $C>0$ is a generic constant independent of N. Hence we can get the desired result of the lemma from the above analysis. □

3.2. Convergence analysis: strongly convex cost functional. We assume that the cost functional J defined in (2.3) is strongly convex in the following sense: there exists some constant $\lambda>0$ such that for any control terms $u,v\in\mathcal{U}$,

$$\langle\nabla J_u(u)-\nabla J_u(v),u-v\rangle\geq\lambda\|u-v\|_2^2. \tag{3.9}$$

By Assumption 3.1, we have the following smoothness result for J: There exists a positive constant C_L such that

$$\int_0^T|\nabla J_u(u_t)-\nabla J_u(v_t)|^2dt\leq C_L\int_0^T|u_t-v_t|^2dt,$$

or equivalently,

$$\|\nabla J_u(u)-\nabla J_u(v)\|_2^2\leq C_L\|u-v\|_2^2. \tag{3.10}$$

Before stating the main convergence theorem, we need an estimate of the error between the true gradient ∇J_u and its piecewise approximation ∇J_u^N.

LEMMA 3.6. *Assume that Assumption 3.1 holds and $f,g\in C_b^4$, $u^k\in\mathcal{U}_N$. Then there exists a constant $C>0$ such that*

$$\sup_{u^k\in\mathcal{U}_N}\|\nabla J_u(u^k)-\nabla J_u^N(u^k)\|_2^2\leq\frac{C}{N}. \tag{3.11}$$

Proof. Denote

$$\begin{aligned}\phi_t^k&:=f_u(X_t^k,u_t^k)^\top Y_t^k+g_u(X_t^k,u_t^k)^\top Z_t^k+r_u(X_t^k,u_t^k)^\top,\\ \phi_n^k&:=f_u(X_n^{k,N},u_{t_n}^k)^\top Y_n^{k,N}+g_u(X_n^{k,N},u_{t_n}^k)^\top Z_n^{k,N}+r_u(X_n^{k,N},u_{t_n}^k)^\top.\end{aligned}$$

We have that

$$\begin{aligned}&\int_0^T(\nabla J_u(u_t^k)-\nabla J_u^N(u_t^k))^2dt\\ &\leq 2\sum_{n=0}^{N-1}\int_{t_n}^{t_{n+1}}\left[(\nabla J_u(u_t^k)-\nabla J_u(u_{t_n}^k))^2+(\nabla J_u(u_{t_n}^k)-\nabla J_u^N(t_n,u_{t_n}^k))^2\right]dt\\ &=2\sum_{n=0}^{N-1}\int_{t_n}^{t_{n+1}}(\mathbb{E}[\phi_t^k-\phi_{t_n}^k])^2dt+2\sum_{n=0}^{N-1}\int_{t_n}^{t_{n+1}}(\mathbb{E}[\phi_{t_n}^k-\phi_n^k])^2dt\leq\frac{C}{N}.\end{aligned}$$

□

Let $u^{*,N}$ be the optimal control for the stochastic optimal control problem (2.4) found in the subset $\mathcal{U}_N$ of the admissible control set $\mathcal{U}$. Then, we have the following estimate.

LEMMA 3.7. *The following inequality holds:*

$$\|\nabla J_u(u^{*,N})-\nabla J_u(u^*)\|_2^2\leq\frac{C}{N}. \tag{3.12}$$

Proof. Let $\bar{u}^N$ be the projection of u^* onto $\mathcal{U}_N$, i.e. $\bar{u}^N=\mathcal{P}_{\mathcal{U}_N}(u^*)=\arg\min_{u^N\in\mathcal{U}_N}\|u^N-u^*\|_2$. Due to the boundedness of u^*, we have

$$\|\bar{u}^N-u^*\|_2\leq\frac{C}{N}.$$

Since the solutions of the backward SDEs are bounded, ∇J_u is also bounded. Thus

$$\begin{aligned}J(\bar{u}^N)-J(u^*)&=\int_0^1\langle\nabla J_u\big(u^*+\epsilon(\bar{u}^N-u^*)\big),\bar{u}^N-u^*\rangle d\epsilon\\ &\leq\sup_{u\in L^2}\|\nabla J_u(u)\|_2\|\bar{u}^N-u^*\|_2\leq C\|\bar{u}^N-u^*\|_2\leq\frac{C}{N}.\end{aligned}$$

Hence

$$J(u^{*,N}) - J(u^*) \le J(\bar{u}^N) - J(u^*) \le \frac{C}{N}.$$

By the strong convexity assumption, we have

$$\langle \nabla J_u(u^*), u^{*,N} - u^* \rangle + \frac{\lambda}{2} \|u^{*,N} - u^*\|_2^2 \le J(u^{*,N}) - J(u^*).$$

Since $\nabla J_u(u_t^*) = 0$, the above inequality leads to

$$\|u^{*,N} - u^*\|_2^2 \le \frac{C}{N}. \tag{3.13}$$

The desired result of the lemma is obtained by the above estimate and the convexity assumption (3.10). ∎

Since $\nabla J_u(u_t^*) = 0$, as a direct consequence of (3.12), we have

$$\|\nabla J_u(u^{*,N})\|_2^2 \le \frac{C}{N}. \tag{3.14}$$

Now we are ready to prove the main convergence result under the convexity assumption. First we estimate the error between the exact solution of the optimal control and the optimal control in the piece-wise constant subset $\mathcal{U}_N$ of the admissible set $\mathcal{U}$.

THEOREM 3.8. *Assume all the assumptions in Lemma 3.6 and the convexity assumption are true. Let $\eta_k = \frac{\theta}{k+M}$ for some constants θ and M such that $\lambda\theta - 4C_L\theta^2/(1+M) > 2$. Also, let $\{u^k\}_k$ be the sequence of estimated optimal control obtained by the SGD optimization scheme* (2.15)*. Then, for large enough K, the following inequality holds*

$$\mathbb{E}\left[\|u^{K+1} - u^{*,N}\|_2^2\right] \le C\left(\frac{N}{K} + \frac{1}{N}\right). \tag{3.15}$$

Proof. Recall that $u^{*,N}$ is the optimal control found in control set $\mathcal{U}_N$. Hence

$$u^{*,N} = \mathcal{P}_{\mathcal{U}_N}(u^{*,N}) = \mathcal{P}_{\mathcal{U}_N}\left(u^{*,N} - \eta_k \nabla J_u(u^{*,N}) + \eta_k \nabla J_u(u^{*,N})\right). \tag{3.16}$$

Subtracting (3.16) both sides of (2.15), we have

$$\|u^{k+1} - u^{*,N}\|_2^2 = \|\mathcal{P}_{\mathcal{U}_N}\left((u^k - u^{*,N}) - \eta_k\left(\nabla j_u^k(u_t^k) - \nabla J_u(u_t^{*,N})\right) - \eta_k \nabla J_u(u_t^{*,N})\right)\|_2^2$$

Taking conditional expectation $\mathbb{E}^k[\cdot]$ to the above equation and then applying Young's inequality, we obtain

$$\begin{aligned}
&\mathbb{E}^k\left[\|u^{k+1} - u^{*,N}\|_2^2\right] \\
&\le (1+\epsilon)\mathbb{E}^k\left[\|(u^k - u^{*,N}) - \eta_k\left(\nabla j_u^k(u^k) - \nabla J_u(u^{*,N})\right)\|_2^2\right] + (1+\frac{1}{\epsilon})\eta_k^2\mathbb{E}^k[\|\nabla J_u(u^{*,N})\|_2^2] \\
&\le (1+\epsilon)\Big(\|(u^k - u^{*,N})\|_2^2 - 2\eta_k\langle \mathbb{E}^k\left[\nabla j_u^k(u^k) - \nabla J_u(u^{*,N})\right], u^k - u^{*,N}\rangle \\
&\quad + \eta_k^2\mathbb{E}^k\left[\|\nabla j_u^k(u^k) - \nabla J_u^N(u^k) + \nabla J_u^N(u^k) - \nabla J_u(u^{*,N})\|_2^2\right]\Big) + (1+\frac{1}{\epsilon})\eta_k^2\mathbb{E}^k[\|\nabla J_u(u^{*,N})\|_2^2].
\end{aligned} \tag{3.17}$$

From the convexity assumption (3.9) and Lemma 3.6, we deduce, from Young's inequality with $\lambda/2$, and the fact that u^k is $\mathcal{G}_k$ measurable, that

$$\begin{aligned}
&-\langle \mathbb{E}^k\left[\nabla j_u^k(u^k) - \nabla J_u(u^{*,N})\right], u^k - u^{*,N}\rangle = -\langle \nabla J_u^N(u^k) - \nabla J_u(u^{*,N}), u^k - u^{*,N}\rangle \\
&= -\langle \nabla J_u^N(u^k) - \nabla J_u(u^k), u^k - u^{*,N}\rangle - \langle \nabla J_u(u^k) - \nabla J_u(u^{*,N}), u^k - u^{*,N}\rangle \\
&\le \frac{2\|\nabla J_u^N(t,u^k) - \nabla J_u(u^k)\|_2^2}{\lambda} + \frac{\lambda}{2}\|u^k - u^{*,N}\|_2^2 - \lambda\|u^k - u^{*,N}\|_2^2 \\
&\le \frac{2}{\lambda}\frac{C}{N} - \frac{\lambda}{2}\|u^k - u^{*,N}\|_2^2.
\end{aligned} \tag{3.18}$$

Moreover, from Lemma 3.5, Lemma 3.6 and the convexity assumption (3.10), we have

$$
\begin{aligned}
&\mathbb{E}^k\big[\|\nabla j_u^k(u^k)-\nabla J_u^N(u^k)+\nabla J_u^N(u^k)-\nabla J_u(u^{*,N})\|_2^2\big]\\
&\le 2\mathbb{E}^k\big[\|\nabla J_u^N(u^k)-\nabla J_u(u^{*,N})\|_2^2\big]+CN\\
&\le 4\Big(\mathbb{E}^k\big[\|\nabla J_u^N(u^k)-\nabla J_u(u^k)\|_2^2\big]+\mathbb{E}^k\big[\|\nabla J_u(u^k)-\nabla J_u(u^{*,N})\|_2^2\big]\Big)+CN\\
&\le 4\Big(\frac{C}{N}+C_L\|u^k-u^{*,N}\|_2^2\Big)+CN,
\end{aligned}
\tag{3.19}
$$

where we use C to denote a generic constant independent of k, N, and controls.

Inserting Eqs. (3.18)-(3.19) in (3.17) and applying (3.14), we obtain

$$
\begin{aligned}
&\mathbb{E}^k\big[\|u^{k+1}-u^{*,N}\|_2^2\big]\\
&\le (1+\epsilon)\Big(\|(u^k-u^{*,N})\|_2^2+\frac{4}{\lambda}\eta_k\frac{C}{N}-\lambda\eta_k\|u^k-u^{*,N}\|_2^2\\
&\quad+4\eta_k^2\big(\frac{C}{N}+C_L\|u^k-u^{*,N}\|_2^2\big)+4\eta_k^2CN\Big)+(1+\frac{1}{\epsilon})\eta_k^2\frac{C}{N}\\
&=(1+\epsilon)\Big((1-c_k\eta_k)\|u^k-u^{*,N}\|_2^2+4\eta_k^2CN+(\frac{4}{\lambda}\eta_k+4\eta_k^2)\frac{C}{N}\Big)+(1+\frac{1}{\epsilon})\eta_k^2\frac{C}{N},
\end{aligned}
\tag{3.20}
$$

where $c_k:=\lambda-4C_L\eta_k$.

Let $\tilde{\eta}_k=\frac{1}{k+M}$. We can find θ and M such that

$$
2c:=\lambda\theta-4C_L\frac{\theta^2}{1+M}>2,
$$

and we have that, when k is large enough, $2c\tilde{\eta}_k\ge c_k\eta_k$ for $\eta_k=\frac{\theta}{k+M}$. Choosing $\epsilon=c\tilde{\eta}_k$ in (3.20), we have

$$
\begin{aligned}
&\mathbb{E}^k\big[\|u^{k+1}-u^{*,N}\|_2^2\big]\\
&\le (1+c\tilde{\eta}_k)\Big((1-2c\tilde{\eta}_k)\|u_t^k-u^{*,N}\|_2^2+C\big(\tilde{\eta}_k^2N+\frac{\tilde{\eta}_k}{N}\big)\Big)+(1+\frac{1}{c\tilde{\eta}_k})\tilde{\eta}_k^2\frac{C}{N}\\
&\le (1-c\tilde{\eta}_k)\|u_t^k-u^{*,N}\|_2^2+C\tilde{\eta}_k^2N+C\frac{\tilde{\eta}_k}{N}.
\end{aligned}
$$

Next, we take expectation $\mathbb{E}[\cdot]$ to both sides of the above estimate and apply it recursively from $k=0$ to $k=K$ to get

$$
\begin{aligned}
\mathbb{E}\big[\|u^{K+1}-u^{*,N}\|_2^2\big]\le&\prod_{k=0}^{K}(1-c\tilde{\eta}_k)\mathbb{E}\big[\|u^0-u^{*,N}\|_2^2\big]+\left(\sum_{m=1}^{K}\tilde{\eta}_{m-1}\prod_{k=m}^{K}(1-c\tilde{\eta}_k)\right)\frac{C}{N}\\
&+\frac{C\tilde{\eta}_k}{N}+\left(\sum_{m=1}^{K}\tilde{\eta}_{m-1}^2\prod_{k=m}^{K}(1-c\tilde{\eta}_k)\right)CN\\
\le&(K+M)^{-c}\|u^0-u^{*,N}\|_2^2+CN\left((K+M)^{-1}-\frac{(1+M)^{c-1}}{(K+M)^c}\right)+\frac{C}{N}.
\end{aligned}
$$

Since $c>1$ and $\prod_{k=m}^{K}(1-c\tilde{\eta}_k)\sim O\big((K/m)^{-c}\big)$, the above estimate gives us

$$
\mathbb{E}\big[\|u^{K+1}-u^{*,N}\|_2^2\big]\le\frac{CN}{K+M}+\frac{C}{N}\le C\left(\frac{N}{K}+\frac{1}{N}\right).
$$

□

Note that the estimate (3.15) provides the convergence between the estimated control u^{K+1} and the optimal control found in the subspace $\mathcal{U}_N$. The next theorem, which is the main result of this section, gives the convergence between u^{K+1} and the exact optimal control $u^*\in\mathcal{U}$.

THEOREM 3.9. *Assume that all the assumptions hold in Theorem 3.8 and assume the optimal control u^* is bounded. Then, for large enough K, we have the following convergence result:*

$$\mathbb{E}\big[\|u^{K+1}-u^*\|_2^2\big] \le C\left(\frac{N}{K}+\frac{1}{N}\right). \tag{3.21}$$

Proof. From the estimate (3.15) obtained in Theorem 3.8 and the fact (3.13), we have

$$\begin{aligned}\mathbb{E}\big[\|u^{K+1}-u^*\|_2^2\big] &\le 2\mathbb{E}\big[\|u^{K+1}-u^{*,N}\|_2^2\big]+2\mathbb{E}\big[\|u^{*,N}-u^*\|_2^2\big]\\ &\le C\left(\frac{N}{K}+\frac{1}{N}\right)+\frac{C}{N},\end{aligned}$$

as desired. □

REMARK 3.10. *The error estimate* (3.21) *reveals the interplay between the number of the sample-wise SGD iterations and the depth of the corresponding stochastic neural network and the error of approximating the SDE and the backward SDE. In particular, by choosing $K=N^2$, we recover the half order convergence ($O(\frac{1}{\sqrt{N}})$) the numerical algorithms* (2.9) *and* (2.10) *for the SDE and the corresponding backward SDE. On the other hand, by choosing $N=\sqrt{K}$, we obtain $\frac{1}{4}$ order convergence in K, which is half of that for conventional SGD iterations. Such order reduction is expected for stochastic computing involving SDEs.*

3.3. Convergence analysis for non-convex cost functional. We now study the convergence of the same algorithm without the convexity assumption. In addition to the boundedness assumptions in Assumption 3.1, we assume that the running cost only depends on control u, and we let $R(u):=\int_0^T r(u_t)dt$, which is uniformly bounded from below by $C\|u\|_2$, i.e.

$$R^2(u) \ge C\|u\|_2^2, \tag{3.22}$$

where C is a constant that satisfies Lemma 3.6. Also, we assume that learning rates η_k satisfy the Robbins-Monro condition:

$$\sum_{k=1}^{\infty}\eta_k=\infty, \qquad \sum_{k=1}^{\infty}\eta_k^2<\infty. \tag{3.23}$$

To proceed, we denote J^N as the cost function corresponding to the fully calculated approximate gradient ∇J_u^N, i.e.,

$$\lim_{\delta\to 0}\frac{J^N(u+\delta v)-J^N(u)}{\delta}=\langle \nabla J_u^N(u),v\rangle, \qquad u,v\in\mathcal{U}_N,$$

Note that for any u_0, $u\in\mathcal{U}_N$,

$$\begin{aligned}J^N(u) &= J^N(u_0)+\int_0^1 \frac{d}{d\epsilon}\Big(J^N\big(u_0+\epsilon(u-u_0)\big)\Big)d\epsilon\\ &= J^N(u_0)+\int_0^1 \big\langle \nabla J_u^N\big(u_0+\epsilon(u-u_0)\big),u-u_0\big\rangle d\epsilon.\end{aligned} \tag{3.24}$$

Then, we can show that J^N is bounded from below based on the boundedness assumption for the running cost R, i.e.,

$$\begin{aligned}J^N(u^k) &= J(u^k)-J(u_0)+J^N(u_0)-\int_0^1\big\langle \nabla J_u\big(u_0+\epsilon(u^k-u_0)\big)-\nabla J_u^N\big(u_0+\epsilon(u^k-u_0)\big),(u^k-u_0)\big\rangle d\epsilon\\ &\ge J(u^k)-C_0-C\sup_{u^k\in\mathcal{U}_N}\|\nabla J_u(u^k)-\nabla J_u^N(u^k)\|_2\|u^k-u_0\|_2\\ &= \Phi(X_T^k)-C_0+R(u^k)-C\sup_{u^k\in\mathcal{U}_N}\|\nabla J_u(u^k)-\nabla J_u^N(u^k)\|_2\|u^k-u_0\|_2,\end{aligned}$$

where X_T^k is the solution of the state equation (2.2) driven by the control u^k. We choose $u_0=0$ in the above estimate. Then, it follows from Lemma 3.6 that

$$J^N(u^k)\geq \Phi(X_T^k)-C_0+\underbrace{R(u^k)-\sqrt{C/N}\|u^k\|_2}_{\geq 0}\geq \Phi(X_T^k)-C_0,$$

which indicates that J^N is bounded from below.

We aim to show that for a given depth N, $\lim_{k\to\infty}\|\nabla J_u(u^k)\|_2\to 0$ a.s.. To this end, we first state two propositions that can be derived following the same proofs as in Theorem 5.3.1 and Theorem 5.3.3 of [39].

PROPOSITION 3.11. *Assume the assumptions (a), (b), (d) in Assumption 3.1 hold. Let $X_n^{N,u}$ and $X_n^{N,v}$ be approximate solutions introduced in* (2.9) *driven by two controls $u,v\in\mathcal{U}_N$, and we denote $\Delta_N X_n^{u,v}:=X_n^{N,u}-X_n^{N,v}$. Then, we have*

$$\max_{0\leq n\leq N}|\Delta_N X_n^{u,v}|^2\leq C|u-v|^2.$$

PROPOSITION 3.12. *Assume that Assumption 3.1 hold and let $\Delta_N Y_n^{u,v}:=Y_n^{N,u}-Y_n^{N,v}$ and $\Delta_N Z_n^{u,v}:=Z_n^{N,u}-Z_n^{N,v}$ for numerical solutions Y^N and Z^N with controls $u,v\in\mathcal{U}_N$. We have the following estimate*

$$\sum_{0\leq n\leq N}\mathbb{E}[|\Delta_N Y^{u,v}|^2]+h\sum_{n=0}^{N-1}\mathbb{E}[|\Delta_N Z_n^{u,v}|^2]\leq C|u-v|^2.$$

With the above propositions, we can derive the following lemma:

LEMMA 3.13. *Let $w,v\in\mathcal{U}_N$. Under Assumption 3.1, there exists a positive constant C such that*

$$\|\nabla J_u^N(w)-\nabla J_u^N(v)\|_2^2\leq C\|w-v\|_2^2.$$

Proof. Since $w,v\in\mathcal{U}_N$, we can write $w=(w_0,w_1,\cdots,w_n,\cdots,w_N)$ and $v=(v_0,v_1,\cdots,v_n,\cdots,v_N)$. Then, it follows from Proposition 3.11, Proposition 3.12 and Assumption 3.1 that

$$\begin{aligned}
&\|\nabla J_u^N(u)-\nabla J_u^N(v)\|_2^2\\
&\leq h\sum_{n=0}^{N-1}\Big(\mathbb{E}\big[|f_u(X_n^{N,w},w_n)^\top Y_n^{N,w}-f_u(X_n^{N,v},v_n)^\top Y_n^{N,v}|+|g_u(X_n^{N,w},w_n)^\top Z_n^{N,w}-g_u(X_n^{N,v},v_n)^\top Z_n^{N,v}|\\
&\qquad\qquad+|r_u(w_n)^\top-r_u(v_n)^\top|\big]\Big)^2\\
&\leq Ch\sum_{n=0}^{N-1}\mathbb{E}\big[|Y_n^{N,w}-Y_n^{N,v}|^2+|Z_n^{N,w}-Z_n^{N,v}|^2+|X_n^{N,w}-X_n^{N,v}|^2+|w_n-v_n|^2\big]\\
&\leq Ch\sum_{n=0}^{N-1}|w_n-v_n|^2.
\end{aligned}$$

□

The following lemma mimics Lemma 4.4 of [3].

LEMMA 3.14. *Let $\{u^k\}_k$ be the sequence of approximate controls obtained by* (2.15). *Under Assumption 3.1 and the bounded from below assumption* (3.22), *we have the following estimate*

$$\mathbb{E}^k[J^N(u^{k+1})]\leq J^N(u^k)-\eta_k\|\nabla J_u^N(u^k)\|_2^2+CN\eta_k^2. \tag{3.25}$$

Proof. From the definition of J^N and (3.24), we have that

$$J^N(u^{k+1})-J^N(u^k)-\langle\nabla J_u^N(u^k),u^{k+1}-u^k\rangle=\int_0^1\big\langle\nabla J_u^N\big(u^k+\epsilon(u^{k+1}-u^k)\big)-\nabla J_u^N(u^k),\ u^{k+1}-u^k\big\rangle d\epsilon.$$

Applying Lemma 3.13 to the right-hand side of the above equation, we get

$$J^N(u^{k+1}) - J^N(u^k) - \langle \nabla J_u^N(u^k), u^{k+1} - u^k \rangle \leq C\|u^{k+1} - u^k\|_2^2.$$

Hence we can rewrite the above inequality to get the following estimate

$$J^N(u^{k+1}) \leq J^N(u^k) - \eta_k \langle \nabla J_u^N(u^k), \nabla j_u^k(u^k) \rangle + C(\eta_k)^2 \|\nabla j_u^k(u^k)\|_2^2. \tag{3.26}$$

Next, we take the conditional expectation $\mathbb{E}^k[\cdot]$ on both sides of (3.26). Similar to the argument in proving Proposition 3.2, we have

$$\mathbb{E}^k[\langle \nabla J_u^N(u^k), \nabla j_u^k(u^k) \rangle] = \|\nabla J_u^N(u^k)\|_2^2,$$

which, together with the estimate (3.8), gives us

$$\mathbb{E}^k[J^N(u^{k+1})] \leq J^N(u^k) - \eta_k \|\nabla J_u^N(u^k)\|_2^2 + CN\eta_k^2$$

as desired. □

The following lemma gives the final preparation for the main theorem of this subsection.

LEMMA 3.15. *Under Assumption 3.1 and the bounded from below assumption* (3.22)*, suppose that*

$$\sum_{k=0}^{\infty} \eta_k \mathbb{E}[\|\nabla J_u^N(u^k)\|_2^2] < \infty. \tag{3.27}$$

Then we have

$$\lim_{k\to\infty} \|\nabla J_u^N(u^k)\|_2^2 = 0, \qquad a.s..$$

Proof. We first show that

$$\liminf_{k\to\infty} \|\nabla J_u^N(u^k)\|_2^2 = 0, \qquad a.s.. \tag{3.28}$$

If (3.28) is not true, there exists constants $K > 0$ and $a > 0$ such that for all $k > K$, $\|\nabla J_u^N(u^k)\|_2^2 > a$. As a result, we have from our assumption in (3.23) that

$$\sum_{k=K+1}^{\infty} \eta_k \|\nabla J_u^N(u^k)\|_2^2 > a \sum_{k=K+1}^{\infty} \eta_k = \infty,$$

which contradicts the assumption (3.27) in this lemma.

On the other hand, suppose

$$\limsup_{k\to\infty} \|\nabla J_u^N(u^k)\|_2^2 > 0, \qquad a.s.. \tag{3.29}$$

Then, we can find two sequences of stopping times $\{m_k\}_k$ and $\{n_k\}_k$ defined inductively as follows: given $\epsilon > 0$, let

$$m_0 := \inf\{k : \|\nabla J_u^N(u^k)\|_2^2 > 2\epsilon\},$$

$$n_k := \inf\{k > m_k : \|\nabla J_u^N(u^k)\|_2^2 < \epsilon\},$$

$$m_{k+1} := \inf\{k > n_k : \|\nabla J_u^N(u^k)\|_2^2 > 2\epsilon\}.$$

Hence, we have

$$\infty > \sum_{k=0}^{\infty} \eta_k \|\nabla J_u^N(u^k)\|_2^2 \geq \sum_{k=0}^{\infty} \sum_{i=m_k}^{n_k-1} \eta_i \|\nabla J_u^N(u^i)\|_2^2 \geq \epsilon \sum_{k=0}^{\infty} \sum_{i=m_k}^{n_k-1} \eta_i.$$

Therefore, $\lim_{k\to\infty}\sum_{i=m_k}^{n_k-1}\eta_i=0$, *a.s.*. By (3.8) in the proof of Lemma 3.5, we have

$$\mathbb{E}^k[\|u^{k+1}-u^k\|_2^2]=\eta_k^2\mathbb{E}^k[\|\nabla j_u^k(u^k)\|_2^2]\le CN\eta_k^2.$$

By the triangle inequality, the above estimate gives

$$\|u^{n_k}-u^{m_k}\|_2\le\sqrt{CN}\sum_{i=m_k}^{n_k-1}\eta_i\longrightarrow 0,\ \text{ as } k\to\infty.$$

Then, by Lemma 3.13 we obtain

$$\lim_{k\to\infty}\|\nabla J_u^N(u^{n_k})-\nabla J_u^N(u^{m_k})\|_2^2\longrightarrow 0,\quad a.s.. \tag{3.30}$$

By definition of stopping times $\{m_k\}_k$ and $\{n_k\}_k$, we have

$$\begin{aligned}\epsilon&<\|\nabla J_u^N(u^{m_k})\|_2^2-\|\nabla J_u^N(u^{n_k})\|_2^2=\|\nabla J_u^N(u^{m_k})-\nabla J_u^N(u^{n_k})+\nabla J_u^N(u^{n_k})\|_2^2-\|\nabla J_u^N(u^{n_k})\|_2^2\\&\le(1+2)\|\nabla J_u^N(u^{m_k})-\nabla J_u^N(u^{n_k})\|_2^2+(1+\frac{1}{2})\|\nabla J_u^N(u^{n_k})\|_2^2-\|\nabla J_u^N(u^{n_k})\|_2^2\\&\le 3\|\nabla J_u^N(u^{m_k})-\nabla J_u^N(u^{n_k})\|_2^2+\frac{1}{2}\|\nabla J_u^N(u^{n_k})\|_2^2\\&<3\|\nabla J_u^N(u^{m_k})-\nabla J_u^N(u^{n_k})\|_2^2+\frac{1}{2}\epsilon.\end{aligned}$$

However, letting $k\to\infty$ in the above inequality will result a contradiction due to (3.30).

Therefore, we have $\limsup_{k\to\infty}\|\nabla J_u^N(u^k)\|_2^2=0$, *a.s.*. Together with (3.28), we obtain the desired convergence result in the lemma. □

We are ready to prove the main convergence theorem that shows our sample-wise backpropagation method convergences in training a N-layer SNN.

THEOREM 3.16. *Under Assumption 3.1 and the bounded from below assumption* (3.22)*, we have the following result for a given integer* $N\in\mathbb{N}$*:*

$$\lim_{k\to\infty}\|\nabla J_u^N(u^k)\|_2^2=0,\qquad a.s..$$

Proof. Let $\beta_k=\eta_k\|\nabla J_u^N(u^k)\|_2^2$. We proceed to prove $\sum_{k=0}^{\infty}\mathbb{E}[\beta_k]<\infty$, which is the condition (3.27) in Lemma 3.15 that will give the desired result of this theorem.

Define

$$\lambda_k=J^N(u^k)+CN\sum_{i=k}^{\infty}\eta_i^2,$$

where C is a constant that satisfies (3.25) in Lemma 3.14. Then, we apply (3.25) to get

$$\mathbb{E}^k[\lambda_{k+1}]=\mathbb{E}^k[J^N(u^{k+1})]+CN\sum_{i=k+1}^{\infty}\eta_i^2\le J^N(u^k)+CN\sum_{i=k}^{\infty}\eta_i^2-\beta_k=\lambda_k-\beta_k<\lambda_k. \tag{3.31}$$

Since J^N is bounded from below, as discussed above, $\mathbb{E}[\lambda_k]$ is also bounded below. Moreover, we know from (3.31) that $\{\lambda_k\}_k$ is a super-martingale bounded from below. Therefore, by the martingale convergence theorem, we obtain

$$\lim_{k\to\infty}\mathbb{E}[\lambda_k]<\infty.$$

Then, we apply (3.31) to get

$$\begin{aligned}\sum_{k=0}^{\infty}\mathbb{E}[\beta_k]&\le\sum_{k=0}^{\infty}\mathbb{E}\left[\lambda_k-\mathbb{E}^k[\lambda_{k+1}]\right]\\&=\sum_{k=0}^{\infty}\left(\mathbb{E}[\lambda_k]-\mathbb{E}[\lambda_{k+1}]\right)<\infty.\end{aligned}$$

Then, the desired result of this theorem holds by using the conclusion in Lemma 3.15. □

4. SNN implementation and numerical experiments . In this section, we verify the convergence results obtained in Section 3 and discuss the implementation issues of the sample-wise numerical algorithm for the SNN through three numerical experiments. In the first experiment, we solve a linear-quadratic stochastic optimal control problem that satisfies the convexity assumption, and the exact solution of the optimal control is known. We want to use this classic stochastic optimal control example to examine the convergence rate obtained in Theorem 3.9. In the second experiment, we learn a noise perturbed 8D function by using the SNN to demonstrate the capability of our sample-wise backpropagation method in learning high dimensional functions. We also want to use this experiment to verify the boundedness/differentiability assumption of the activation function in the convergence analysis and confirm that the ReLU activation function is not suitable to implement the SNN with sample-wise backward propagation. In the last experiment, we implement the SNN by incorporating convolutional blocks [15,20,28] and solve the benchmark classification problem using the MNIST handwritten dataset and the Fashion-MNIST dataset. We want to use this experiment to show that our method works well for classic machine learning tasks and has a robustness advantage compared with the conventional deterministic convolution neural network. The CPU we use to run the numerical experiments in this section is an M1 Pro Chip with 3.2GHz and 16 GB memory.

4.1. Numerical verification of convergence rate. In this experiment, we consider the following state process on the temporal interval $[0,T]$:

$$dX_t = (u_t - a_t)dt + \sigma u_t dW_t, \tag{4.1}$$

where X and u are 8D state and control. The vector function a_t is defined by

$$a_t = \Big[\frac{-t^2}{2\beta_t}, \frac{-\sin t}{\beta_t}, \frac{-0.5\exp(1-t)}{\beta_t}, \frac{-t^3}{3\beta_t}, \frac{-\ln(1+t)}{\beta_t}, \frac{-\cos 2\pi t}{\beta_t}, \frac{-\tan t}{\beta_t}, \frac{1-t}{2\sigma^2(1-t)+2}\Big]^\top,$$

where $\beta_t = (1+\sigma^2)+\sigma^2(1-t)$. The cost functional is defined by

$$J(u) = \frac{1}{2}\int_0^1 \mathbb{E}[|X_t - X_t^*|^2]dt + \frac{1}{2}\int_0^1 |u_t|^2 dt + \frac{1}{2}|X_T|^2, \tag{4.2}$$

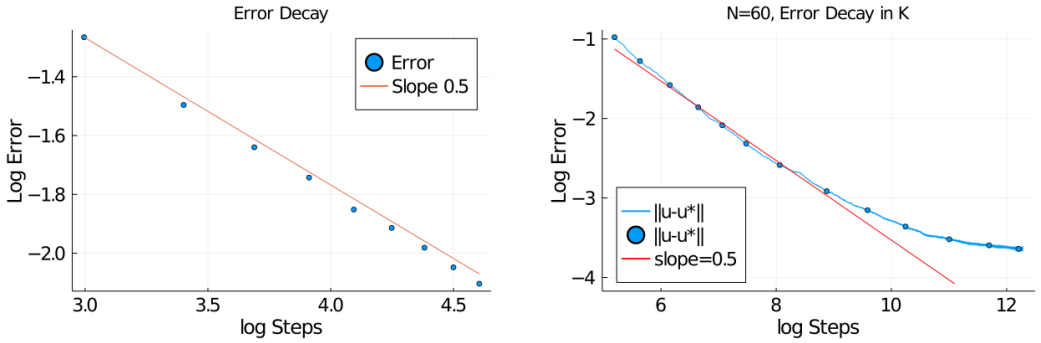
where $|\cdot|$ is the Euclidean norm in $\mathbb{R}^8$, and X_t^* is defined by

$$X_t^* := \Big[t + \alpha_t\frac{0.5 - x_T^{(1)}}{\sigma^2}, \cos t + \alpha_t\frac{\sin 1 - x_T^{(2)}}{\sigma^2}, -\frac{\exp(1-t)}{2} + \alpha_t\frac{0.5 - x_T^{(3)}}{\sigma^2}, t^2 + \alpha_t\frac{1/3 - x_T^{(4)}}{\sigma^2},$$
$$\frac{1}{1+t} + \alpha_t\frac{\ln 2 - x_T^{(5)}}{\sigma^2}, -2\pi\sin t + \alpha_t\frac{2\pi\cos 1 - x_T^{(6)}}{\sigma^2}, \sec^2 t + \alpha_t\frac{\tan 1 - x_T^{(7)}}{\sigma^2}, \frac{t}{\sigma^2} + 1 - \frac{1}{2\sigma^4}\ln\frac{1+\sigma^2}{1+\sigma^2(1-t)}\Big]^\top,$$

where $\alpha_t = \ln\dfrac{1+2\sigma^2}{\sigma^2(2-t)+1}$. For $D := \dfrac{\ln(1+\frac{\sigma^2}{1+\sigma^2})}{\sigma^2 + \ln(1+\frac{\sigma^2}{1+\sigma^2})}$, $x_T = [x_T^{(1)}, \cdots, x_T^{(7)}]$ in the above are defined as $x_T := \Big[D/2, D\cdot\sin 1, D/2, D, D\cdot\ln 2, D\cos 2\pi, D\cdot\tan 1\Big]^\top$. Since the stochastic optimal control problem (4.1)-(4.2) is a linear-quadratic problem, we can find the analytic expression for the optimal control as

$$u_t^* := \Big[\frac{-t^2/2 + T^2/2 - x_T^{(1)}}{\beta_t}, \frac{-\sin t + \sin 1 - x_T^{(2)}}{\beta_t}, \frac{-1/2\exp(T-t) + 1/2 - x_T^{(3)}}{\beta_t}, \frac{-t^3 + T^3 - 3x_T^{(4)}}{3\beta_t},$$
$$\frac{-\ln(1+t) + \ln(1+T) - x_T^{(5)}}{\beta_t}, \frac{-\cos 2\pi t + \cos 2\pi - x_T^{(6)}}{\beta_t}, \frac{-\tan t + \tan 1 - x_T^{(7)}}{\beta_t}, \frac{T-t}{\sigma^2(T-t)+1}\Big]^\top.$$

We choose $T=1$, $\sigma=0.5$, $X_0=0$, and $N=20,30,40,\cdots,100$ with iteration steps $K=0.2\times N^2$ for each N. In the convergence analysis (3.21) proved in Theorem 3.9, we can see that when choosing $K\sim O(N^2)$, the approximation error $\|u^{K+1}-u^*\|_2$ has half-order convergence rate. To verify this



(a) Convergence with respect to N ($K=0.2N^2$). (b) Convergence with respect to K.

Fig. 1: Example 1. Convergence results.

result, we solve the stochastic optimal control problem (4.1)-(4.2) 50 times and plot the root mean square errors (RMSEs) in Figure 1 (a). The blue dots give the RMSEs corresponding to N (presented by $\log N$ on the x-axis), and the red straight line shows the slope of half-order convergence. From this figure, we can see that our algorithm does provide a half-order convergence rate when the relation $K\sim O(N^2)$ is satisfied.

In Figure 1 (b), we further investigate the convergence with respect to K, and we let $N=60$ be a fixed partition number. The blue dots show the RMSEs corresponding to different K values (presented by $\log K$ on the x-axis), and the red straight line indicates the slope of half-order convergence. From this figure, we can see that the convergence rate of our algorithm is well-aligned with the half-order slope while the number of iterations K is relatively small. However, when K values become larger, the accuracy of our algorithm does not always improve accordingly. This verifies the existence of the $\frac{C}{N}$ term in the analysis (3.21), which becomes the bottleneck in convergence with large iteration number K. In other words, a fixed partition number N limits the algorithm's convergence no matter how many iteration steps we implement in the optimization procedure.

4.2. Implementation with combined-sigmoid activation functions. We aim to use the SNN model (2.1) to learn random noise perturbed functions by using discrete function values as data and demonstrate the performance of our method in both function approximation and uncertainty quantification. We let $h=1$ and choose the activation function in the SNN as a combination of sigmoid functions, which is defined as follows:

$$b(X)=\sum_{l=1}^{L} a_l\sigma(WX+V), \tag{4.3}$$

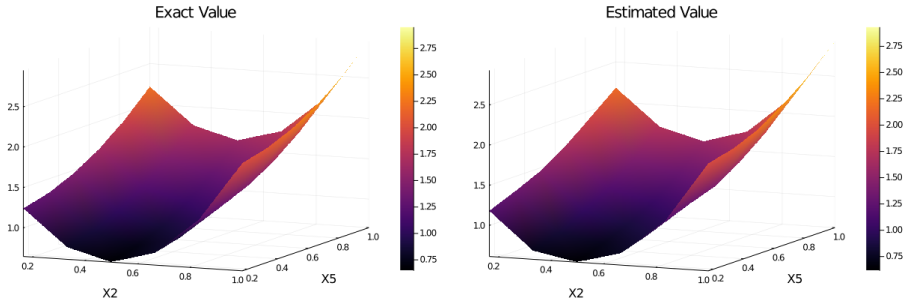
where $\sigma(x)=\frac{1}{1+e^{-x}}$, $\{a_l\}_{l=1}^{L}$ take values in $[-4.5,4.5]$, which is a set of weights of different sigmoid functions, W is the weight matrix for the state of neurons, and V is a bias vector.

Now we apply the SNN with the combined sigmoid activation function to approximate the following noise perturbed 8 dimensional function:

$$f(x_1,\ldots,x_8):=\exp(x_1)\cos(2\pi x_2)+8x_3(x_4-0.5)^2+x_5+\log(2+x_6)+x_7^2+2x_8+0.05\xi,\quad \xi\sim N(0,1).$$

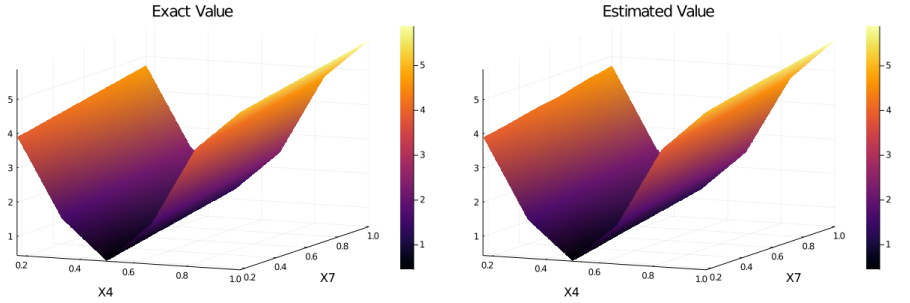
The training data are collected on 6^8 spatial mesh points in the hypercube $[0,1]^8$. Since this is a high dimensional function, we use an SNN with $N=15$ layer and put 40 neurons in each layer. The number of optimization iterations for training the SNN is 1.5×10^7, and the CPU time for this training procedure is 1508 seconds. It's worth mentioning that each data sample contains only a limited amount of information about the perturbation noise, and a significant amount of computational effort in this 8-dimensional approximation example would contribute to finding the confidence band of the function,

which is also the main challenge in uncertainty quantification for deep learning. Although by collecting data on some sophisticatedly selected high dimensional sample points, such like Latin Hypercube points, would make the data more effectively represent the target function. The main reason that we choose the uniform grid in this example is for the convenience of presentation. When plotting the predicted functions, which will be presented in Figure 3, it's easier to demonstrate the approximated function on fixed uniform mesh-grid points in one direction while fixing others. On the other hand, the challenge of high-dimensional approximation still remains in the current framework with limited data on uniform mesh-grid points.



(a) Exact surface: X_2-X_5

(b) Approximate surface: X_2-X_5



(c) Exact surface: X_4-X_7

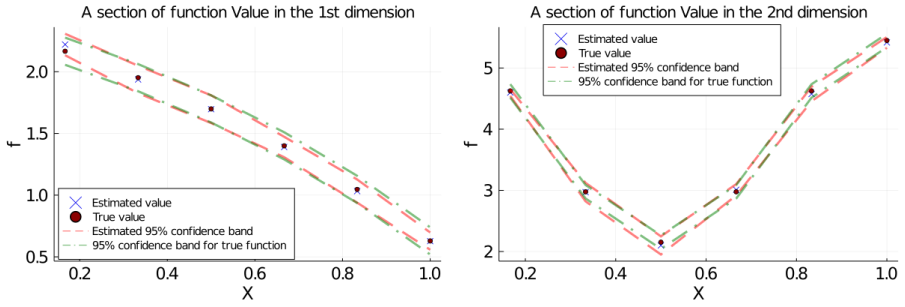
(d) Approximate surface: X_4-X_7

Fig. 2: Example 2. 8D Function approximation – surface views.

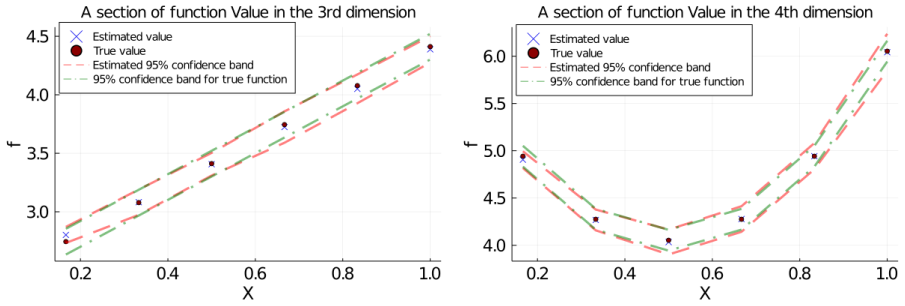
In Figure 2, we compare the real marginal surface of function f (presented in mean values) with our estimated surface using the trained SNN. In Figure 2 (a), we present the exact X_2-X_5 marginal surface of function f, and Figure 2 (b) shows the SNN learned X_2-X_5 marginal surface; In Figure 2 (c), we present the exact X_4-X_7 marginal surface of function f, and Figure 2 (d) shows the SNN learned X_4-X_7 marginal surface. From this figure, we can see that training the SNN 1.5×10^7 steps with our sample-wise backpropagation algorithm can give a very good approximation for the original function.

To show more details of the SNN's performance in function approximation, we present section views of each direction in Figure 3, where the exact function mean values are plotted by red dots, the SNN estimated function mean values are plotted by blue crosses, the true 95% uncertainty bands are presented by green dashed lines, and the SNN estimated 95% confidence bands are shown by red dashed lines. From this figure, we can see that the trained SNN can accurately approximate the function's mean values and the uncertainty caused by the random variable ξ.

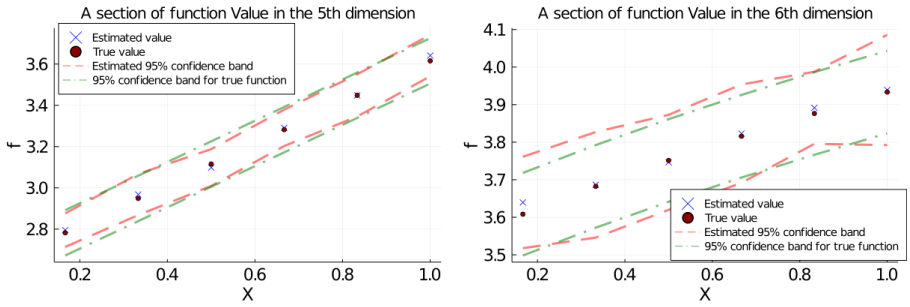
To validate the convergence of our algorithm, we train SNNs with $N=2,5,9,12,15$, and present the accuracy of function approximation and uncertainty quantification with different numbers of training steps in Figure 4 and Figure 5. We can see from those figures that the SNN can achieve higher accuracy in estimating both function values and uncertainty bands by using more layers. For a fixed SNN depth, we can see a clear convergence trend when implementing more iteration steps. However, an accuracy barrier appears in each experiment due to the fixed SNN depth, which typically results in limited



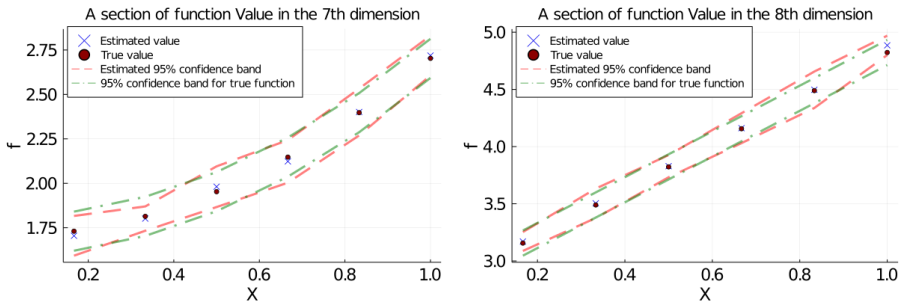
(a) X_1 section (b) X_2 section



(c) X_3 section (d) X_4 section



(e) X_5 section (f) X_6 section



(g) X_7 section (h) X_8 section

Fig. 3: Example 2. 8D Function approximation – section views

representation capability by using neural network models.

REMARK 4.1. *In the convergence analysis, we require boundedness for the activation function in the SNN (see Lemma 3.6). To verify this requirement, we repeat the above experiment by replacing the*

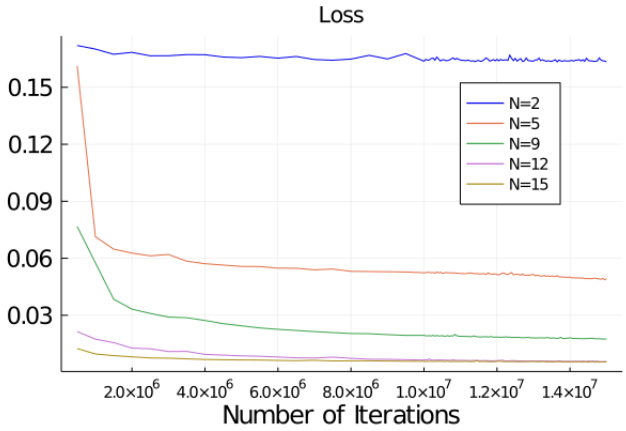


Fig. 4: Example 2. Convergence in function approximation.

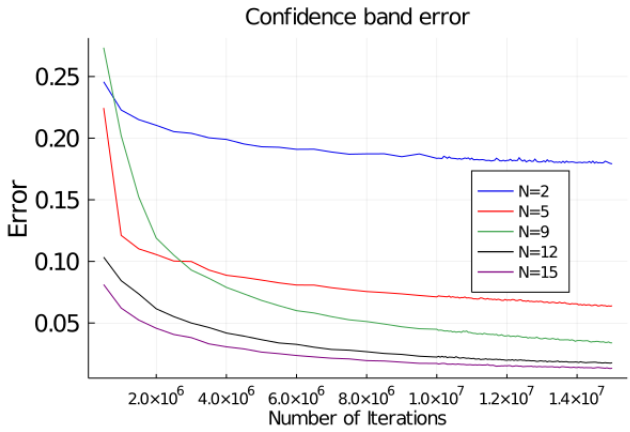


Fig. 5: Example 2. Convergence in uncertainty quantification.

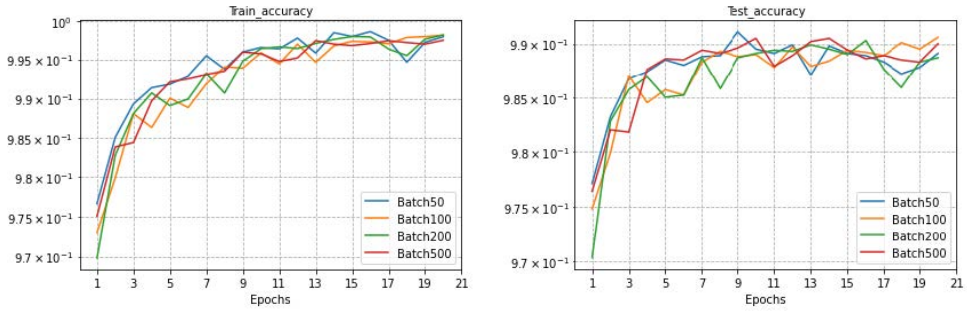
sigmoid function σ in (4.3) *with a ReLU function. It turns out that ReLU-activated SNN does not even produce convergent results, which confirms the necessity of boundedness for the activation function*[1].

4.3. Implementation with convolutional blocks. We implement the SNN with convolution blocks and examine its performance in solving benchmark machine learning problems and confirm the robustness advantage of SNN over deterministic deep neural networks. Specifically, we solve the image classification problem with the MNIST handwritten digit dataset and the Fashion-MNIST dataset. In each dataset, we have 60,000 training data samples and 10,000 testing data samples. Here, instead of using a single-realization sample to represent the state variable, we use a mini-batch of samples to approximate expectations in our SNN algorithm[2] .

In Figure 6, we present the training accuracy and testing accuracy of our SNN algorithm in solving the classification problem over the MINST handwritten dataset with different batch sizes, and in Figure 7, we present the training accuracy and testing accuracy of our SNN algorithm in solving the classification problem over the Fashion-MINST dataset with different batch sizes. We can see from those figures that our SNN algorithm can generate accurate classification results fairly fast (with various batch sizes) in both training and testing for bench-mark classification tasks.

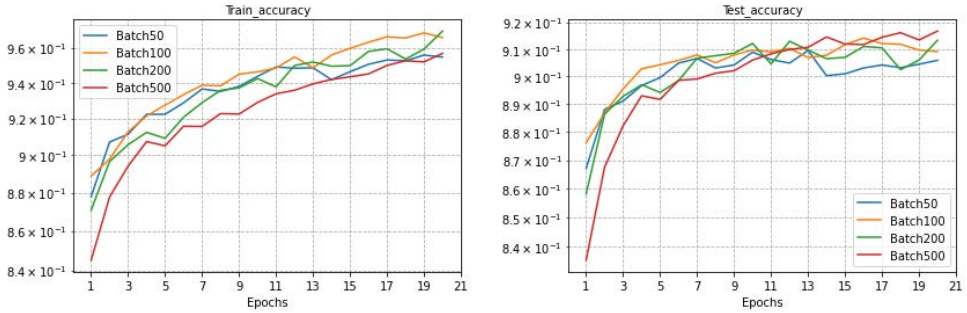
[1]The numerical implementation about the ReLU activation experiment can be found on github at *https://github.com/Huisun317/SNN*

[2]Detailed formulation of the SNN structure and our specific numerical implementation can be found on github at *https://github.com/Huisun317/SNN*



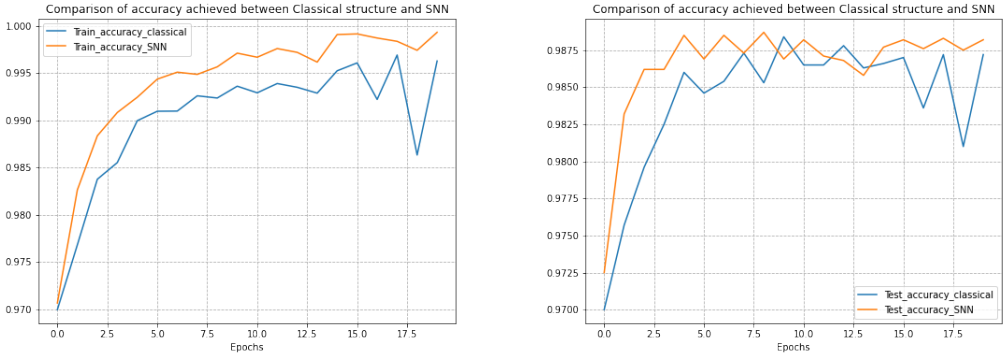
(a) Training accuracy (b) Testing accuracy

Fig. 6: Example 3. Performance of SNN: MINST handwritten dataset.



(a) Training accuracy (b) Testing accuracy

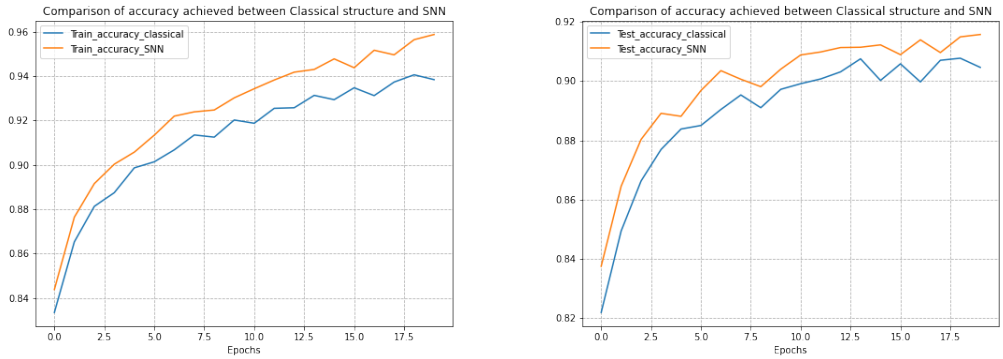
Fig. 7: Example 3. Performance of SNN: Fashion-MINST dataset.



(a) Training accuracy (b) Testing accuracy

Fig. 8: Example 3. Comparison: MINST handwritten dataset.

In addition, we compare the performance of our SNN algorithm with the classical (deterministic) conventional convolution neural network (CNN), where both methods utilize the same network architecture (except for the noise term introduced in the SNN method) with the batch size 200. The comparison of training and testing accuracy of each method is presented in Figure 8 and Figure 9. As we can see



(a) Training accuracy

(b) Testing accuracy

Fig. 9: Example 3. Comparison: Fashion-MINST dataset.

from those figures, the SNN constantly outperforms the CNN in terms of both training accuracy and testing accuracy with respect to the number of training epochs. However, we need to point out that the SNN method is more time consuming in each training epoch due to the complexity of the SNN formulation, and people can further improve the efficiency of the current SNN algorithm by applying accelerated SGD techniques.

To present the necessity of applying SNN as a probabilistic learning tool, we consider an adversarial game scenario in which some "attacker" attacks the data (through Fast Gradient Sign Attack (FGSM)) and try to fool the neural network (pre-trained on the clean dataset). At the same time, the model retrains itself along the way by using the attacked dataset to play an adversarial game against the attacker. We want to use this experiment to show that our SNN algorithm is more robust than the deterministic CNN in handling unexpected noises in the data.

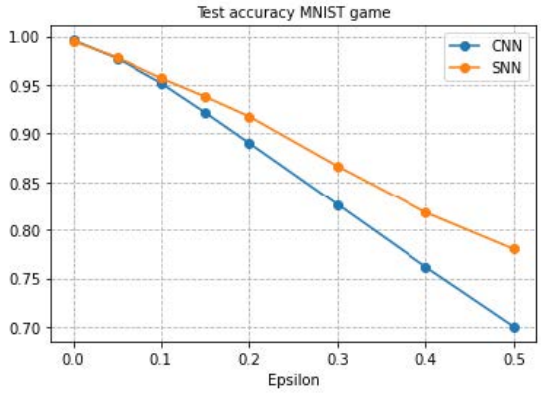


Fig. 10: Example 3. Attacked model accuracy with different levels of noise (MNIST).

In Figure 10, we compare the accuracy between our SNN with CNN with respect to the level of noises added to the MNIST handwritten dataset, where the same CNN structure is adopted to our SNN model in the drift term. We can see that without noise in the dataset, both SNN and CNN provide very accurate classification results. Then, SNN and CNN suffer lower accuracy due to the errors added to the dataset by the "attacker". However, our SNN algorithm started outperforming the CNN while more noises were added. In Table 1, we present the accuracy comparison between SNN and CNN under adversarial attacks. We can see from this table that with 50% of noise added to the data that causes

Table 1: Classification accuracy with different levels of noise: MNIST

ϵ	0.0	0.05	0.1	0.15	0.2	0.3	0.4	0.5
Attacked CNN	0.996	0.9773	0.952	0.9217	0.89	0.826	0.762	0.701
Attacked SNN	0.995	0.9782	0.957	0.938	0.917	0.866	0.818	0.770

Table 2: Classification accuracy with different levels of noise: Fashion-MNIST

ϵ	0.0	0.05	0.1	0.15	0.2	0.3	0.4	0.5
Attacked CNN	0.915	0.760	0.668	0.639	0.596	0.550	0.531	0.499
Attacked SNN	0.910	0.796	0.745	0.698	0.662	0.633	0.607	0.583

misclassification, the SNN could still reach 77% of accuracy, which is 7% higher than the conventional CNN method.

In Figure 11, we compare the accuracy between our SNN with CNN with respect to the level of noise added to the Fashion-MNIST dataset. We can see that without noise in the dataset, both SNN and CNN provide very accurate classification results. Then, SNN and CNN suffer lower accuracy due to the errors added to the dataset by the "attacker", and our SNN algorithm increasingly outperforms CNN while more and more noises are added. In Table 2, we present the accuracy comparison between

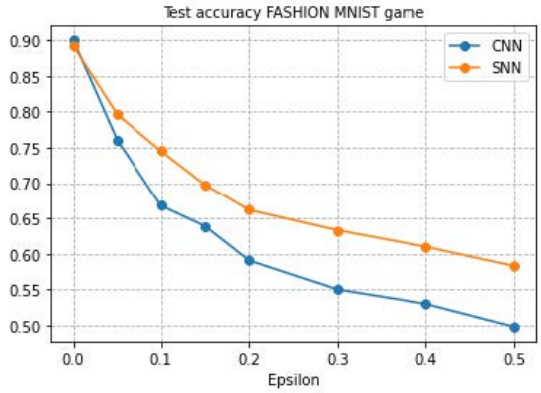


Fig. 11: Example 3. Classification accuracy with different levels of noise (Fashion-MNIST).

SNN and CNN under adversarial attacks. We can see from this table that with 50% of noise added to the data that causes misclassification, the SNN could still reach 58% of accuracy, which is 8% higher than the conventional CNN method.

5. Conclusions. In this paper, we carried out rigorous numerical analysis to prove the convergence of a novel sample-wise backpropagation algorithm for training a class of stochastic neural networks (SNNs). Under the convexity assumption, we derived a half-order convergence rate for our algorithm in the mean square sense. Without the convexity assumption, we proved the convergence of stochastic gradient descent iteration in the algorithm. Numerical experiments are conducted with appropriately designed neural network architecture under the SNN framework, and the results of the numerical experiments validated the convergence of the algorithm as well as the advantageous performance of the SNN algorithm for probabilistic machine learning.

REFERENCES

[1] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark, 2020.

[2] F. Bao, Y. Cao, R. Archibald, and H. Zhang. A backward sde method for uncertainty quantification in deep learning. *arXiv:2011.14145*, 2021.

[3] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM Rev.*, 60(2):223–311, 2018.

[4] A. Athalye C. G. Northcutt and J. Mueller. Pervasive label errors in test sets destabilize machine learning benchmarks. *arXiv:2103.14749*, 2021.

[5] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 6571–6583. Curran Associates, Inc., 2018.

[6] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 3140–3150. Curran Associates, Inc., 2019.

[7] Weinan E. A proposal on machine learning via dynamical systems. *Commun. Math. Stat.*, 5(1):1–11, 2017.

[8] N. El Karoui, S. Peng, and M. C. Quenez. Backward stochastic differential equations in finance. *Math. Finance*, 7(1):1–71, 1997.

[9] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR.

[10] Nicholas Geneva and Nicholas Zabaras. Quantifying model form uncertainty in Reynolds-averaged turbulence models with Bayesian deep neural networks. *J. Comput. Phys.*, 383:125–147, 2019.

[11] Nicholas Geneva and Nicholas Zabaras. Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks. *J. Comput. Phys.*, 403:109056, 32, 2020.

[12] R. Gerstberger and P. Rentrop. Feedforward neural nets as discretization schemes for ODEs and DAEs. volume 82, pages 117–128. 1997. 7th ICCAM 96 Congress (Leuven).

[13] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34(1):014004, 22, 2018.

[14] M. Miremadi J. Bughin K. George P. Willmott J. Manyika, M. Chui and M. Dewhurst. A future that works: Automation, employment, and productivity. *Technical report, McKinsey Global Institute*, 2017.

[15] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678, New York, NY, USA, 2014. ACM.

[16] Peter E. Kloeden and Eckhard Platen. *Numerical solution of stochastic differential equations*, volume 23 of *Applications of Mathematics (New York)*. Springer-Verlag, Berlin, 1992.

[17] Lingkai Kong, Jimeng Sun, and Chao Zhang. Sde-net: Equipping deep neural networks with uncertainty estimates, 2020.

[18] Yongchan Kwon, Joong-Ho Won, Beom Joon Kim, and Myunghee Cho Paik. Uncertainty quantification using Bayesian neural networks in classification: application to biomedical image segmentation. *Comput. Statist. Data Anal.*, 142:106816, 17, 2020.

[19] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 6402–6413. Curran Associates, Inc., 2017.

[20] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998.

[21] Xuechen Li, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 3870–3882, Online, 26–28 Aug 2020. PMLR.

[22] Xuanqing Liu, Tesi Xiao, Si Si, Qin Cao, Sanjiv Kumar, and Cho-Jui Hsieh. How does noise help robustness? explanation and exploration under the neural sde framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[23] Jin Ma and Jiongmin Yong. *Forward-backward stochastic differential equations and their applications*, volume 1702 of *Lecture Notes in Mathematics*. Springer-Verlag, Berlin, 1999.

[24] Patrick L. McDermott and Christopher K. Wikle. Bayesian recurrent neural network models for forecasting and quantifying uncertainty in spatial-temporal data. *Entropy*, 21(2):Paper No. 184, 25, 2019.

[25] A.Nemirovski, A.Juditsky, G.Lan, A.Shapiro. *Robust Stochastic Approximation Approach to Stochastic Programming*. SIAM J. OPTIM. Vol 19, No. 4, pp.1574-1609.

[26] Huyên Pham. *Continuous-time Stochastic Control and Optimization with Financial Applications* Stochastic Modelling and Applied Probability 61. Springer.

[27] Andrey V. Savchenko. Probabilistic neural network with complex exponential activation functions in image recognition. *IEEE Trans. Neural Netw. Learn. Syst.*, 31(2):651–660, 2020.

[28] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013.

[29] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, L Robert Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

[30] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. 550:354–, October 2017.

[31] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel S. Emer. Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE*, 105:2295–2329, 2017.

[32] G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[33] Belinda Tzen and Maxim Raginsky. Neural stochastic differential equations: Deep latent gaussian models in the diffusion limit. *CoRR*, abs/1905.09883, 2019.

[34] Ling Wu, Kepa Zulueta, Zoltan Major, Aitor Arriaga, and Ludovic Noels. Bayesian inference of non-linear multiscale model parameters accelerated by a deep neural network. *Comput. Methods Appl. Mech. Engrg.*, 360:112693, 17, 2020.

[35] Y. Bengio Y. LeCun and G. Hinton. Deep learning. *Nature*, 512(7553):436–444, 2015.

[36] Liu Yang, Xuhui Meng, and George Em Karniadakis. B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data. *J. Comput. Phys.*, 425:109913, 2021.

[37] Jiayu Yao, Weiwei Pan, Soumya Ghosh, and Finale Doshi-Velez. Quality of uncertainty quantification for bayesian neural network inference, 2019.

[38] Jiongmin Yong and Xun Yu Zhou. *Stochastic controls*, volume 43 of *Applications of Mathematics (New York)*. Springer-Verlag, New York, 1999. Hamiltonian systems and HJB equations.

[39] J. Zhang. *Backward Stochastic Differential Equations - From Linear to Fully Nonlinear Theory*. Probability Theory and Stochastic Modelling. Springer, New York, NY, 2017.

[40] Weidong Zhao, Lifeng Chen, and Shige Peng. A new kind of accurate numerical method for backward stochastic differential equations. *SIAM J. Sci. Comput.*, 28(4):1563–1581, 2006.