# NeuralSDE

2025년 7월 30일 수요일 오전 11:27

NeuralSDE

# Neural SDEs as Infinite-Dimensional GANs

**Patrick Kidger** [1 2] **James Foster** [1 2] **Xuechen Li** [3] **Harald Oberhauser** [1 2] **Terry Lyons** [1 2]


## Abstract

Stochastic differential equations (SDEs) are a staple of mathematical modelling of temporal dynamics. However, a fundamental limitation has been that such models have typically been relatively inflexible, which recent work introducing Neural SDEs has sought to solve. Here, we show that the current classical approach to fitting SDEs may be approached as a special case of (Wasserstein) GANs, and in doing so the neural and classical regimes may be brought together. The input noise is Brownian motion, the output samples are time-evolving paths produced by a numerical solver, and by parameterising a discriminator as a Neural Controlled Differential Equation (CDE), we obtain Neural SDEs as (in modern machine learning parlance) continuous-time generative time series models. Unlike previous work on this problem, this is a direct extension of the classical approach without reference to either prespecified statistics or density functions. Arbitrary drift and diffusions are admissible, so as the Wasserstein loss has a unique global minima, in the infinite data limit any SDE may be learnt. Example code has been made available as part of the `torchsde` repository.


## 1. Introduction

### 1.1. Neural differential equations

Since their introduction, neural ordinary differential equations (Chen et al., 2018) have prompted the creation of a variety of similarly-inspired models, for example based around controlled differential equations (Kidger et al., 2020; Morrill et al., 2020), Lagrangians (Cranmer et al., 2020), higher-order ODEs (Massaroli et al., 2020; Norcliffe et al., 2020), and equilibrium points (Bai et al., 2019).

In particular, several authors have introduced *neural stochastic differential equations* (neural SDEs), such as Tzen & Raginsky (2019a); Li et al. (2020); Hodgkinson et al. (2020), among others. This is our focus here.

Neural differential equations parameterise the vector field(s) of a differential equation by neural networks. They are an elegant concept, bringing together the two dominant modelling paradigms of neural networks and differential equations.

The main idea – fitting a parameterised differential equation to data, often via stochastic gradient descent – has been a cornerstone of mathematical modelling for a long time (Giles & Glasserman, 2006). The key benefit of the neural network hybridisation is its availability of easily-trainable high-capacity function approximators.

### 1.2. Stochastic differential equations

Stochastic differential equations have seen widespread use for modelling real-world random phenomena, such as particle systems (Coffey et al., 2012; Pavliotis, 2014; Lelièvre & Stoltz, 2016), financial markets (Black & Scholes, 1973; Cox et al., 1985; Brigo & Mercurio, 2001), population dynamics (Aratό, 2003; Soboleva & Pleasants, 2003) and genetics (Huillet, 2007). They are a natural extension of ordinary differential equations (ODEs) for modelling systems that evolve in continuous time subject to uncertainty.

The dynamics of an SDE consist of a deterministic term and a stochastic term:

$$\mathrm{d}X_t = f(t, X_t)\,\mathrm{d}t + g(t, X_t) \circ \mathrm{d}W_t, \qquad (1)$$

where $X = \{X_t\}_{t\in[0,T]}$ is a continuous $\mathbb{R}^x$-valued stochastic process, $f : [0,T]\times\mathbb{R}^x \to \mathbb{R}^x$, $g : [0,T]\times\mathbb{R}^x \to \mathbb{R}^{x\times w}$ are functions and $W = \{W_t\}_{t\geq 0}$ is a $w$-dimensional Brownian motion. We refer the reader to Revuz & Yor (2013) for a rigorous account of stochastic integration.

The notation "$\circ$" in the noise refers to the SDE being understood using Stratonovich integration. The difference between Itô and Stratonovich will not be an important choice here; we happen to prefer the Stratonovich formulation as the dynamics of (1) may then be informally interpreted as

$$X_{t+\Delta t} \approx \mathrm{ODESolve}\left(X_t, f(\cdot)+g(\cdot)\frac{\Delta W}{\Delta t}, [t, t+\Delta t]\right),$$

[1]Mathematical Institute, University of Oxford [2]The Alan Turing Institute, The British Library [3]Stanford. Correspondence to: Patrick Kidger <kidger@maths.ox.ac.uk>.

*Proceedings of the 38th International Conference on Machine Learning*, PMLR 139, 2021. 

**초록**

확률 미분 방정식(Stochastic Differential Equations, SDEs)은 시간적 동역학을 수학적으로 모델링하는 데 있어 핵심적인 도구이다. 그러나 이러한 모델은 전통적으로 유연성이 부족하다는 근본적인 한계를 지니고 있으며, 이를 해결하기 위해 최근 Neural SDEs가 도입되었다. 본 연구에서는 기존의 고전적인 SDE 추정 방식이 (Wasserstein) 생성적 적대 신경망(GANs)의 특수한 경우로 해석될 수 있음을 보인다. 이를 통해 신경망 기반 방법과 고전적 방식이 하나의 틀로 통합될 수 있다. 입력 노이즈는 브라운 운동(Brownian motion)이며, 출력 샘플은 수치 해석기를 통해 생성되는 시간에 따라 변화하는 경로이다. 판별기(discriminator)는 Neural Controlled Differential Equation(CDE)으로 파라미터화되며, 이를 통해 Neural SDEs는 (현대 머신러닝 용어로) 연속 시간 생성 시계열 모델로서 구현된다. 본 접근법은 이전 연구들과 달리 미리 지정된 통계량이나 밀도 함수 없이 고전적인 방법을 직접 확장한 것이다. 드리프트(drift)와 확산(diffusion)의 형태는 임의로 설정 가능하며, Wasserstein 손실 함수는 전역 최소값을 갖기 때문에, 무한한 데이터가 주어졌을 때 어떤 SDE도 학습될 수 있다. 예제 코드는 torchsde 저장소에 공개되어 있다.

where $\Delta W \sim \mathcal{N}(0, \Delta t I_w)$ denotes the increment of the Brownian motion over the small time interval $[t, t + \Delta t]$.

Historically, workflows for SDE modelling have two steps:

1. A domain expert will formulate an SDE model using their experience and knowledge. One frequent and straightforward technique is to add "$\sigma \circ dW_t$" to a pre-existing ODE model, where $\sigma$ is a fixed matrix.
2. Once an SDE model is chosen, the model parameters must be calibrated from real-world data. Since SDEs produce random sample paths, parameters are often chosen to capture some desired expected behaviours. That is, one trains the model to match target statistics:

$$\big\{\mathbb{E}\big[F_i(X)\big]\big\}_{1 \leq i \leq n}, \tag{2}$$

where the real-valued functions $\{F_i\}$ are prespecified. For example in mathematical finance, the statistics (2) represent option prices that correspond to the functions $F_i$, which are termed payoff functions; for the well-known and analytically tractable Black–Scholes model, these prices can then be computed explicitly for call and put options (Black & Scholes, 1973).

The aim of this paper (and neural SDEs more generally) is to strengthen the capabilities of SDE modelling by hybridising with deep learning.

### 1.3. Contributions

SDEs are a classical way to understand uncertainty over paths or over time series. Here, we show that the current classical approach to fitting SDEs may be generalised, and approached from the perspective of Wasserstein GANs. In particular this is done by putting together a neural SDE and a neural CDE (controlled differential equation) as a generator–discriminator pair.

Arbitrary drift and diffusions are admissible, which from the point of view of the classical SDE literature offers unprecedented modelling capacity. As the Wasserstein loss has a unique global minima, then in the infinite data limit arbitrary SDEs may be learnt.

Unlike much previous work on neural SDEs, this operates as a direct extension of the classical tried-and-tested approach. Moreover and to the best of our knowledge, this is the first approach to SDE modelling that involves neither prespecified statistics nor the use of density functions.

In modern machine learning parlance, neural SDEs become continuous-time generative models. We anticipate applications in the main settings for which SDEs are already used – now with enhanced modelling power. For example later we will consider an application to financial time series.

## 2. Related work

We begin by discussing previous formulations, and applications, of neural SDEs. Broadly speaking these may be categorised in two groups. The first use SDEs as a way to gradually insert noise into a system, so that the terminal state of the SDE is the quantity of interest. The second instead consider the full time-evolution of the SDE as the quantity of interest.

Tzen & Raginsky (2019a;b) obtain Neural SDEs as a continuous limit of deep latent Gaussian models. They train by optimising a variational bound, using forward-mode autodifferentiation. They consider only theoretical applications, for modelling distributions as the terminal value of an SDE.

Li et al. (2020) give arguably the closest analogue to the neural ODEs of Chen et al. (2018). They introduce neural SDEs via a subtle argument involving two-sided filtrations and backward Stratonovich integrals, but in doing

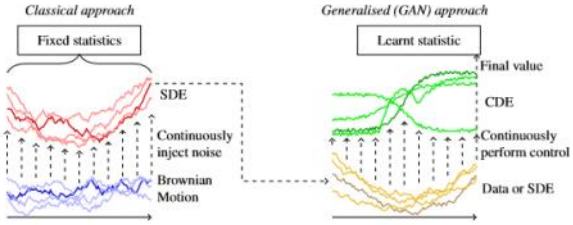


*Figure 1.* Pictorial summary of just the high level ideas: Brownian motion is continuously injected as noise into an SDE. The classical approach fits the SDE to prespecified statistics. Generalising to (Wasserstein) GANs, which instead introduce a learnt statistic (the discriminator), we may fit much more complicated models.

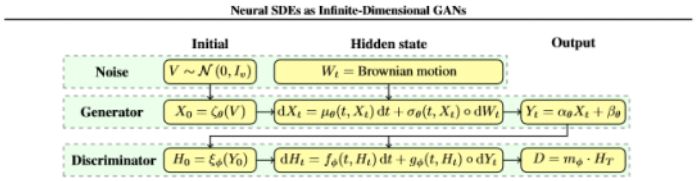


*Figure 2.* Summary of equations.

so are able to introduce a backward-in-time adjoint equation, using only efficient-to-compute vector-Jacobian products. In applications, they use neural SDEs in a latent variable modelling framework, using the stochasticity to model Bayesian uncertainty.

Hodgkinson et al. (2020) introduce neural SDEs as a limit of random ODEs. The limit is made meaningful via rough path theory. In applications, they use the limiting random ODEs, and treat stochasticity as a regulariser within a normalising flow. However, they remark that in this setting the optimal diffusion is zero. This is a recurring problem: Innes et al. (2019) also train neural SDEs for which the optimal diffusion is zero.

Rackauckas et al. (2020) treat neural SDEs in classical Feynman–Kac fashion, and like Hodgkinson et al. (2020); Tzen & Raginsky (2019a;b), optimise a loss on just the terminal value of the SDE.

Briol et al. (2020); Gierjatowicz et al. (2020); Cuchiero et al. (2020) instead consider the more general case of using a neural SDE to model a time-varying quantity, that is to say not just considering the terminal value of the SDE. Letting $\mu, \nu$ denote the learnt and true distributions on path space, they all train by minimising $\left|\int f \mathrm{d}\mu - \int f \mathrm{d}\nu\right|$ for functions of interest $f$ (such as derivative payoffs). This corresponds to training with a non-characteristic MMD (Gretton et al., 2013).

Several authors, such as Oganesyan et al. (2020); Hodgkinson et al. (2020); Liu et al. (2019), seek to use stochasticity as a way to enhance or regularise a neural ODE model.

Song et al. (2021), building on the discrete time counterparts Song & Ermon (2019); Ho et al. (2020), consider an SDE that is fixed (and prespecified) rather than learnt. However by approximating one of its terms with a neural network trained with score matching, then the SDE becomes a controlled way to inject noise so as to sample from complex high-dimensional distributions such as images.

Our approach is most similar to Li et al. (2020), in that we treat neural SDEs as learnt continuous-time model components of a differentiable computation graph. Like both Rackauckas et al. (2020) and Gierjatowicz et al. (2020) we emphasise the connection of our approach to standard mathematical formalisms. In terms of the two groups mentioned at the start of this section, we fall into the second: we use stochasticity to model distributions on path space. The resulting neural SDE is not an improvement to a similar neural ODE, but a standalone concept in its own right.

## 3. Method

### 3.1. SDEs as GANs

Consider some (Stratonovich) integral equation of the form

$$X_0 \sim \mu, \quad \mathrm{d}X_t = f(t, X_t)\,\mathrm{d}t + g(t, X_t) \circ \mathrm{d}W_t,$$

for initial probability distribution $\mu$, (Lipschitz continuous) functions $f$, $g$ and Brownian motion $W$. The strong solution to this SDE may be defined as the unique function $S$ such that $S(\mu, W) = X$ almost surely (Rogers & Williams, 2000, Chapter V, Definition 10.9).

Intuitively, this means that SDEs are maps from a noise distribution (Wiener measure, the distribution of Brownian motion) to some solution distribution, which is a probability distribution on path space.

We recommend any of Karatzas & Shreve (1991), Rogers & Williams (2000), or Revuz & Yor (2013) as an introduction to the theory of SDEs.

SDEs can be sampled from: this is what a numerical SDE solver does. However, evaluating its probability density is not possible; in fact it is not even defined in the usual sense.[1] As such, an SDE is typically fit to data by asking that the model statistics

$$\left\{\mathbb{E}_{X \sim \text{model}}\left[F_i(X)\right]\right\}_{1 \leq i \leq n},$$

match the data statistics

$$\left\{\mathbb{E}_{X \sim \text{data}}\left[F_i(X)\right]\right\}_{1 \leq i \leq n},$$

[1]Technically speaking, a probability density is the Radon–Nikodym derivative of the measure with respect to the Lebesgue measure. However, the Lebesgue measure only exists for finite dimensional spaces. In infinite dimensions, it is instead necessary to define densities with respect to for example Gaussian measures.

이 절의 시작에서 언급한 두 그룹 중에서, 우리는 두 번째 그룹에 해당한다: 우리는 경로 공간(path space) 위의 분포(distribution)를 모델링하기 위해 확률성(stochasticity)을 사용한다. 그 결과로 얻어지는 neural SDE(신경 확률 미분 방정식)는 유사한 neural ODE(신경 보통 미분 방정식)에 대한 개선이 아니라, 그 자체로 독립된 개념이다.

**3. 방법 (Method)**
**3.1. GAN으로서의 SDEs**
다음과 같은 (Stratonovich 해석의) 적분 방정식을 고려하자:

X_0 ∼ μ, dX_t = f(t, X_t)dt + g(t, X_t)∘dW_t,
여기서 초기 확률 분포는 μ, 함수 f, g는 (Lipschitz 연속인) 함수이며 W는 Brownian motion(브라운 운동)이다.
이 SDE에 대한 강해(strong) 해는 다음을 만족하는 유일한 함수 S로 정의될 수 있다: S(μ, W) = X (거의 확실하게),
(Rogers & Williams, 2000, Chapter V, Definition 10.9).
직관적으로, 이는 SDE가 잡음 분포(Wiener measure, 브라운 운동의 분포)로부터 어떤 해(solution) 분포로의 사상(mapping)임을 의미한다. 이 해 분포는 경로 공간 위의 확률 분포이다.
SDE 이론에 대한 입문서로는 Karatzas & Shreve (1991), Rogers & Williams (2000), 또는 Revuz & Yor (2013)를 추천한다.
SDE는 샘플링이 가능하다: 이것이 수치적 SDE 해석기(numerical SDE solver)가 수행하는 작업이다. 그러나 확률 밀도(probability density)를 평가하는 것은 불가능하며, 사실 통상적인 의미에서 정의조차 되지 않는다.[1] 따라서, SDE는 일반적으로 다음과 같이 데이터에 맞춰 학습된다:
모델의 통계값들이
E_{X ∼ model} F_i(X), 1 ≤ i ≤ n
데이터의 통계값들과
E_{X ∼ data} F_i(X), 1 ≤ i ≤ n
일치하도록 요구한다.

[1] 기술적으로 말하자면, 확률 밀도는 어떤 측도(measure)에 대한 Radon−Nikodym 도함수이다. 그러나 Lebesgue 측도는 유한 차원 공간에서만 존재한다. 무한 차원 공간에서는, 예를 들어 Gaussian 측도 같은 것에 대해 밀도를 정의해야 한다.

for some functions of interest $F_i$. Training may be done via stochastic gradient descent (Giles & Glasserman, 2006).

For completeness we now additionally introduce the relevant ideas for GANs. Consider some noise distribution $\mu$ on a space $\mathcal{X}$, and a target probability distribution $\nu$ on a space $\mathcal{Y}$. A generative model for $\nu$ is a learnt function $G_\theta \colon \mathcal{X} \to \mathcal{Y}$ trained so that the (pushforward) distribution $G_\theta(\mu)$ approximates $\nu$. Sampling from a trained model is typically straightforward, by sampling $\omega \sim \mu$ and then evaluating $G_\theta(\omega)$.

Many training methods rely on obtaining a probability density for $G_\theta(\mu)$; for example this is used in normalising flows (Rezende & Mohamed, 2015). However this is not in general computable, perhaps due to the complicated internal structure of $G_\theta$. Instead, GANs examine the statistics of samples from $G_\theta(\mu)$, and seek to match the statistics of the model to the statistics of the data. Most typically this is a learnt scalar statistic, called the discriminator. An optimally-trained generator is one for which

$$\mathbb{E}_{X \sim \text{model}}\big[F(X)\big] = \mathbb{E}_{X \sim \text{data}}\big[F(X)\big]$$

for all statistics $F$, so that there is no possible statistic (or 'witness function' in the language of integral probability metrics (Bińkowski et al., 2018)) that the discriminator may learn to represent, so as to distinguish real from fake.

There are some variations on this theme; GMMNs instead use fixed vector-valued statistics (Li et al., 2015), and MMD-GANs use learnt vector-valued statistics (Li et al., 2017).

In both cases – SDEs and GANs – the model generates samples by transforming random noise. In neither case are densities available. However sampling is available, so that model fitting may be performed by matching statistics. With this connection in hand, we now seek to combine these two approaches.

### 3.2. Generator

Let $Y_{\text{true}}$ be a random variable on $y$-dimensional path space. Loosely speaking, path space is the space of continuous functions $f \colon [0, T] \to \mathbb{R}^y$ for some fixed time horizon $T > 0$. For example, this may correspond to the (interpolated) evolution of stock prices over time. $Y_{\text{true}}$ is what we wish to model.

Let $W \colon [0, T] \to \mathbb{R}^w$ be a $w$-dimensional Brownian motion, and $V \sim \mathcal{N}(0, I_v)$ be drawn from a $v$-dimensional standard multivariate normal. The values $w, v$ are hyperparameters describing the size of the noise.

Let

$$\begin{aligned} \zeta_\theta &\colon \mathbb{R}^v \to \mathbb{R}^x, \\ \mu_\theta &\colon [0, T] \times \mathbb{R}^x \to \mathbb{R}^x, \\ \sigma_\theta &\colon [0, T] \times \mathbb{R}^x \to \mathbb{R}^{x \times w}, \\ \alpha_\theta &\in \mathbb{R}^{y \times x} \\ \beta_\theta &\in \mathbb{R}^y \end{aligned}$$

where $\zeta_\theta$, $\mu_\theta$ and $\sigma_\theta$ are (Lipschitz) neural networks. Collectively they are parameterised by $\theta$. The dimension $x$ is a hyperparameter describing the size of the hidden state.

We define neural SDEs of the form

$$\begin{aligned} X_0 &= \zeta_\theta(V), \\ \mathrm{d}X_t &= \mu_\theta(t, X_t)\,\mathrm{d}t + \sigma_\theta(t, X_t) \circ \mathrm{d}W_t, \\ Y_t &= \alpha_\theta X_t + \beta_\theta, \end{aligned} \tag{3}$$

for $t \in [0, T]$, with $X \colon [0, T] \to \mathbb{R}^x$ the (strong) solution to the SDE, such that in some sense $Y \overset{\mathrm{d}}{\approx} Y_{\text{true}}$. That is to say, the model $Y$ should have approximately the same distribution as the target $Y_{\text{true}}$ (for some notion of approximate). The solution $X$ is guaranteed to exist given mild conditions (such as Lipschitz $\mu_\theta, \sigma_\theta$).

**Architecture** Equation (3) has a certain minimum amount of structure. First, the solution $X$ represents hidden state. If it were the output, then future evolution would satisfy a Markov property which need not be true in general. This is the reason for the additional readout operation to $Y$. Practically speaking $Y$ may be concatenated alongside $X$ during an SDE solve.

Second, there must be an additional source of noise for the initial condition, passed through a nonlinear $\zeta_\theta$, as $Y_0 = \alpha_\theta \zeta_\theta(V) + \beta_\theta$ does not depend on the Brownian noise $W$.

$\zeta_\theta, \mu_\theta$, and $\sigma_\theta$ may be taken to be any standard network architecture, such as a simple feedforward network. (The choice does not affect the GAN construction.)

**Sampling** Given a trained model, we sample from it by sampling some initial noise $V$ and some Brownian motion $W$, and then solving equation (3) with standard numerical SDE solvers. In our experiments we use the midpoint method, which converges to the Stratonovich solution. (The Euler–Maruyama method converges to the Itô solution).

**Comparison to the Fokker–Planck equation** The distribution of an SDE, as learnt by a neural SDE, contains more information than the distribution obtained by learning a corresponding Fokker–Planck equation. The solution to a Fokker–Planck equation gives the (time evolution of the)

어떤 관심 함수 F_i들에 대해, 학습은 stochastic gradient descent(확률적 경사 하강법)을 통해 수행될 수 있다 (Giles & Glasserman, 2006).
완전성을 위해, 이제 GAN(생성적 적대 신경망)에 관련된 개념들도 추가로 소개한다.
어떤 공간 X 위의 잡음 분포 μ와, 어떤 공간 Y 위의 목표 확률 분포 ν가 있다고 하자. ν에 대한 생성 모델은 학습된 함수 G_θ : X → Y 로, (푸시포워드) 분포 G_θ(μ)가 ν를 근사하도록 학습된다.
학습된 모델로부터 샘플링하는 것은 일반적으로 간단한데, ω ~ μ에서 샘플링한 후 G_θ(ω)를 계산하면 된다.
많은 학습 기법들은 G_θ(μ)에 대한 확률 밀도를 얻는 것에 의존한다. 예를 들어 이는 normalising flows(정규화 흐름) (Rezende & Mohamed, 2015)에서 사용된다. 그러나 일반적으로 이는 계산이 불가능하다. 이는 G_θ의 내부 구조가 복잡하기 때문일 수 있다. 대신, GAN은 G_θ(μ)로부터 생성된 샘플의 통계량(statistics)을 조사하고, 모델의 통계량이 데이터의 통계량과 일치하도록 한다.
대부분의 경우, 이는 discriminator(판별자)라고 불리는 학습된 스칼라 통계량이다. 최적으로 학습된 generator(생성자)는 다음을 만족하는 모델이다:

E_{X ~ model} F(X) = E_{X ~ data} F(X)

모든 통계량 F에 대해 위와 같다면, 판별자는 실제와 가짜를 구별할 수 있는 어떤 통계량(또는 적분 확률 거리(integral probability metrics)의 언어로는 '증인 함수(witness function)')도 학습할 수 없다.
이 개념에는 몇 가지 변형이 존재한다. 예를 들어 GMMNs(Generative Moment Matching Networks)은 고정된 벡터값 통계량을 사용하고 (Li et al., 2015), MMD-GANs는 학습된 벡터값 통계량을 사용한다 (Li et al., 2017).
SDE와 GAN 두 경우 모두에서, 모델은 무작위 잡음을 변형하여 샘플을 생성한다. 두 경우 모두 확률 밀도는 사용 불가능하다. 그러나 샘플링은 가능하므로, 모델 적합(model fitting)은 통계량 일치를 통해 수행될 수 있다.
이러한 연결을 바탕으로, 우리는 이제 이 두 접근 방식을 결합하고자 한다.

### 3.2. Generator(생성자)

Y_true를 y차원 경로 공간(path space) 위의 확률 변수라고 하자. 느슨하게 말하면, 경로 공간은 어떤 고정된 시간 구간 T > 0에 대해 f: [0, T] → ℝ^y 인 연속 함수들의 공간이다. 예를 들어, 이는 시간에 따른 주가의 (보간된) 변화 과정을 나타낼 수 있다.
Y_true는 우리가 모델링하고자 하는 대상이다.

W: [0, T] → ℝ^w 를 w차원 브라운 운동(Brownian motion)이라 하고, V ~ N(0, I_v)는 v차원 표준 다변량 정규분포에서 샘플링된다고 하자. w와 v는 노이즈의 크기를 나타내는 하이퍼파라미터이다.

다음과 같이 정의하자:

ζ_θ : ℝ^v → ℝ^x,

μ_θ : [0, T] × ℝ^x → ℝ^x,

σ_θ : [0, T] × ℝ^x → ℝ^{x×w},

α_θ ∈ ℝ^{y×x},

β_θ ∈ ℝ^y

여기서 ζ_θ, μ_θ, σ_θ는 (Lipschitz 연속인) 신경망이며, 전체적으로 θ에 의해 매개변수화된다. 차원 x는 hidden state(숨겨진 상태)의 크기를 나타내는 하이퍼파라미터이다.
다음과 같은 형태의 neural SDE(신경 확률 미분 방정식)를 정의한다:

X_0 = ζ_θ(V),

dX_t = μ_θ(t, X_t)dt + σ_θ(t, X_t)∘dW_t,
Y_t = α_θ X_t + β_θ,
(3)

여기서 t ∈ [0, T], X: [0, T] → ℝ^x는 이 SDE의 (강한) 해이며, 어떤 의미에서 Y ≈_d Y_true, 즉 모델 Y는 목표 Y_true와 거의 동일한 분포를 가져야 한다 (어떤 근사 개념 하에서). μ_θ, σ_θ가 Lipschitz 조건을 만족하는 등의 완화된 조건 하에서 해 X의 존재는 보장된다.

**아키텍처**
식 (3)은 특정한 최소한의 구조를 가진다.
첫째, 해 X는 숨겨진 상태를 나타낸다. 만약 X가 출력이었다면, 이후의 변화는 마르코프 성질(Markov property)을 만족했을 것이며, 이는 일반적으로 성립하지 않을 수 있다. 따라서 Y로의 추가적인 readout 연산이 필요한 이유다. 실제 구현에서 SDE를 푸는 동안 Y는 X와 함께 연결(concatenate)될 수 있다.
둘째, 초기 조건을 위한 추가적인 잡음원이 필요하다. 이는 비선형 함수 ζ_θ를 통해 전달되며, Y_0 = α_θ ζ_θ(V) + β_θ는 브라운 운동 W에 의존하지 않는다.
ζ_θ, μ_θ, σ_θ는 단순한 feedforward network 등 어떤 표준 신경망 구조도 사용할 수 있다 (이 선택은 GAN 구성에 영향을 주지 않는다).

**샘플링**
학습된 모델이 주어졌을 때, 초기 잡음 V와 브라운 운동 W를 샘플링하고, 표준 수치적 SDE 해석기를 사용해 식 (3)을 푼다. 실험에서는 Stratonovich 해에 수렴하는 midpoint method(중점법)를 사용한다. (참고로 Euler–Maruyama 방법은 Itô 해에 수렴한다.)

**Fokker–Planck 방정식과의 비교**
SDE의 분포는, neural SDE를 통해 학습된 경우, 대응하는 Fokker–Planck 방정식을 통해 얻어진 분포보다 더 많은 정보를 포함한다.
Fokker–Planck 방정식의 해는 고정된 시간에서의 해의 확률 밀도 함수(및 그 시간적 변화)를 제공한다.

probability density of a solution *at fixed times*. It does not encode information about the time evolution of individual sample paths. This is exemplified by stationary processes, whose sample paths may be nonconstant but whose distribution does not change over time.

**Stratonovich versus Itô** The choice of Stratonovich solutions over Itô solutions is not mandatory. As the vector fields are learnt then in general either choice is equally admissible.

### 3.3. Discriminator

Each sample from the generator is a path $Y\colon [0,T] \to \mathbb{R}^y$; these are infinite dimensional and the discriminator must accept such paths as inputs. There is a natural choice: parameterise the discriminator as another neural SDE.

Let

$$\begin{aligned} \xi_\phi &\colon \mathbb{R}^y \to \mathbb{R}^h, \\ f_\phi &\colon [0,T] \times \mathbb{R}^h \to \mathbb{R}^h, \\ g_\phi &\colon [0,T] \times \mathbb{R}^h \to \mathbb{R}^{h \times y}, \\ m_\phi &\in \mathbb{R}^h \end{aligned} \tag{4}$$

where $\xi_\phi$, $f_\phi$ and $g_\phi$ are (Lipschitz) neural networks. Collectively they are parameterised by $\phi$. The dimension $h$ is a hyperparameter describing the size of the hidden state.

Recalling that $Y$ is the generated sample, we take the discriminator to be an SDE of the form

$$\begin{aligned} H_0 &= \xi_\phi(Y_0), \\ \mathrm{d}H_t &= f_\phi(t, H_t)\,\mathrm{d}t + g_\phi(t, H_t) \circ \mathrm{d}Y_t, \\ D &= m_\phi \cdot H_T, \end{aligned} \tag{5}$$

for $t \in [0,T]$, with $H\colon [0,T] \to \mathbb{R}^h$ the (strong) solution to this SDE, which exists given mild conditions (such as Lipschitz $f_\phi$, $g_\phi$). The value $D \in \mathbb{R}$, which is a function of the terminal hidden state $H_T$, is the discriminator's score for real versus fake.

**Neural CDEs** The discriminator follows the formulation of a neural CDE (Kidger et al., 2020) with respect to the control $Y$. Neural CDEs are the continuous-time analogue to RNNs, just as neural ODEs are the continuous-time analogue to residual networks (Chen et al., 2018). This is what motivates equation (5) as a probably sensible choice of discriminator. Moreover, it means that the discriminator enjoys properties such as universal approximation.

**Architecture** There is a required minimum amount of structure. There must be a learnt initial condition, and the output should be a function of $H_T$ and not a univariate $H_T$ itself. See Kidger et al. (2020), who emphasise these points in the context of CDEs specifically.

**Single SDE solve** In practice, both generator and discriminator may be concatenated together into a single SDE solve. The state is the combined $[X, H]$, the initial condition is the combined

$$[\zeta_\theta(V),\ \xi_\phi(\alpha_\theta \zeta_\theta(V) + \beta_\theta)],$$

the drift is the combined

$$[\mu_\theta(t, X_t),\ f_\phi(t, H_t) + g_\phi(t, H_t)\alpha_\theta \mu_\theta(t, X_t)],$$

and the diffusion is the combined

$$[\sigma_\theta(t, X_t),\ g_\phi(t, H_t)\alpha_\theta \sigma_\theta(t, X_t)].$$

$H_T$ is extracted from the final hidden state, and $m_\phi$ applied, to produce the discriminator's score for that sample.

**Dense data regime** We still need to apply the discriminator to the training data.

First suppose that we observe samples from $Y_{\text{true}}$ as an irregularly sampled time series $\mathbf{z} = ((t_0, z_0), \ldots, (t_n, z_n))$, potentially with missing data, but which is (informally speaking) densely sampled. Without loss of generality let $t_0 = 0$ and $t_n = T$.

Then it is enough to interpolate $\widehat{z}\colon [0,T] \to \mathbb{R}^y$ such that $\widehat{z}(t_i) = z_i$, and compute

$$\begin{aligned} H_0 &= \xi_\phi(\widehat{z}(t_0)), \\ \mathrm{d}H_t &= f_\phi(t, H_t)\,\mathrm{d}t + g_\phi(t, H_t) \circ \mathrm{d}\widehat{z}_t, \\ D &= m_\phi \cdot H_T, \end{aligned} \tag{6}$$

where $g_\phi(t, H_t) \circ \mathrm{d}\widehat{z}_t$ is defined as a Riemann–Stieltjes integral, stochastic integral, or rough integral, depending on the regularity of $\widehat{z}$.

In doing so the interpolation produces a distribution on path space; the one that is desired to be modelled. For example linear interpolation (Levin et al., 2013), splines (Kidger et al., 2020), Gaussian processes (Li & Marlin, 2016; Futoma et al., 2019) and so on are all acceptable.

In each case the relatively dense sampling of the data makes the choice of interpolation largely unimportant. We use linear interpolation for three of our four experiments (stocks, air quality, weights) later.

**Sparse data regime** The previous option becomes a little less convincing when $\mathbf{z}$ is potentially sparsely observed. In this case, we instead first sample the generator at whatever time points are desired, and then interpolate both the training data and the generated data – solving equation (6) in both cases.

In this case, the choice of interpolation is simply part of the discriminator, and the interpolation is simply a way to

이러한 해는 개별 샘플 경로의 시간적 진화에 대한 정보를 인코딩하지 않는다. 이는 정상 과정(stationary processes)의 예시로 명확히 드러난다. 이러한 과정의 샘플 경로는 일정하지 않을 수 있지만, 그 분포는 시간에 따라 변하지 않는다.

**Stratonovich vs. Itô**
Stratonovich 해를 Itô 해보다 선택하는 것은 필수가 아니다. 벡터장(vector field)이 학습되기 때문에 일반적으로 어느 쪽 선택도 동일하게 유효하다.

### 3.3. Discriminator(판별자)

생성자(generator)로부터의 각 샘플은 Y: [0, T] → 𝕣^y 형태의 경로(path)이며, 이는 무한 차원이다. 따라서 판별자는 이러한 경로를 입력으로 받아야 한다. 이에 대해 자연스러운 선택은 판별자 역시 또 하나의 neural SDE(신경 확률 미분 방정식)로 매개변수화하는 것이다.
다음과 같이 정의하자:

ξ_φ : 𝕣^y → 𝕣^h,

f_φ : [0, T] × 𝕣^h → 𝕣^h,

g_φ : [0, T] × 𝕣^h → 𝕣^{h×y},

m_φ ∈ 𝕣^h
(4)
여기서 ξ_φ, f_φ, g_φ는 (Lipschitz 연속인) 신경망이며, 전체적으로 φ로 매개변수화된다. h는 hidden state의 크기를 나타내는 하이퍼파라미터이다.
생성된 샘플 Y를 기억하자. 우리는 판별자를 다음과 같은 형태의 SDE로 정의한다:
H_0 = ξ_φ(Y_0),

dH_t = f_φ(t, H_t)dt + g_φ(t, H_t) ◦ dY_t,
D = m_φ · H_T
(5)

여기서 t ∈ [0, T], H: [0, T] → 𝕣^h는 이 SDE의 (강한) 해이며, Lipschitz 조건(f_φ, g_φ)에 따라 해의 존재는 보장된다. D ∈ 𝕣는 종단 상태 H_T의 함수이며, 진짜와 가짜를 구별하는 판별자의 점수이다.

**Neural CDEs (신경 제어 미분방정식)**
판별자는 제어 신호(control) Y에 대한 neural CDE(Kidger et al., 2020)의 형식을 따른다. neural CDE는 RNN의 연속시간 아날로그이며, neural ODE가 residual network의 연속시간 아날로그인 것과 동일한 방식이다 (Chen et al., 2018). 이러한 이유로 식 (5)는 판별자로서 합리적인 선택으로 동기부여된다. 또한, 이는 판별자가 보편 근사(universal approximation)와 같은 성질을 가진다는 것을 의미한다.

**아키텍처**
일정한 최소 구조가 요구된다. 학습된 초기 조건이 존재해야 하며, 출력은 H_T 그 자체가 아닌, H_T의 함수여야 한다. 이러한 점은 CDE의 문맥에서 Kidger et al. (2020)이 명확히 강조하였다.

**단일 SDE 해 (Single SDE solve)**
실제에서는, 생성자(generator)와 판별자(discriminator)를 하나의 SDE 해석 과정으로 결합할 수 있다.
이때 상태(state)는 [X, H]로 구성된 결합 벡터이며, 초기 조건은
[ζ_θ(V), ξ_φ(α_θ ζ_θ(V) + β_θ)]
이고, drift 항은 다음과 같은 결합 형태이다:
[μ_θ(t, X_t), f_φ(t, H_t) + g_φ(t, H_t) α_θ μ_θ(t, X_t)]
diffusion 항은 다음과 같다:
[σ_θ(t, X_t), g_φ(t, H_t) α_θ σ_θ(t, X_t)]
최종 상태로부터 H_T를 추출하고, m_φ를 적용하여 해당 샘플에 대한 판별자의 점수(discriminator's score)를 생성한다.

**조밀한 데이터 환경 (Dense data regime)**
학습 데이터에 대해 판별자를 적용해야 한다.

먼저, Y_true로부터의 샘플을 z = ((t₀, z₀), …, (tₙ, zₙ)) 형태의 불규칙하게 샘플링된 시계열로 관측한다고 가정하자. 일부 누락된 데이터가 있을 수 있지만, (비공식적으로) 조밀하게 샘플링되어 있다고 하자. 일반성을 잃지 않고 t₀ = 0, tₙ = T라 하자.
z(t_i) = z_i라 할 때, 다음과 같은 보간 함수 ẑ: [0, T] → 𝕣^y 를 구성하는 것으로 충분하다. 그리고 다음과 같이 계산한다:

H₀ = ξ_φ(ẑ(t₀)),

dH_t = f_φ(t, H_t)dt + g_φ(t, H_t) ◦ dẑ_t,
D = m_φ · H_T
(6)

여기서 g_φ(t, H_t) ◦ dẑ_t는 ẑ의 regularity(정칙성)에 따라, 리만–스틸체스 적분(Riemann–Stieltjes integral), 확률 적분(stochastic integral), 또는 rough integral(거친 경로 적분)로 정의된다.
이렇게 하면, 보간(interpolation)은 경로 공간(path space) 상의 분포를 생성하게 되며, 이는 우리가 모델링하고자 하는 대상이다.
예를 들어 선형 보간(linear interpolation) (Levin et al., 2013), 스플라인(spline) (Kidger et al., 2020), 가우시안 프로세스(Gaussian processes) (Li & Marlin, 2016; Futoma et al., 2019) 등이 모두 사용 가능하다.
각 경우에, 데이터가 상대적으로 조밀하게 샘플링되어 있기 때문에, 보간 방식의 선택은 거의 중요하지 않다. 우리는 이후의 네 가지 실험 중 세 가지(주가, 대기질, 체중)에 대해 선형 보간을 사용한다.

**희소한 데이터 환경 (Sparse data regime)**
데이터 z가 드물게 관측(sparsely observed)된 경우에는, 위와 같은 접근 방식은 설득력이 약해진다.
이 경우, 먼저 생성자(generator)를 원하는 시간 지점에서 샘플링한 후, 학습 데이터와 생성 데이터를 모두 보간하여 식 (6)을 적용한다.
이 상황에서는 보간(interpolation)은 판별자의 일부로 간주되며, 보간은 단지 이산 데이터를 연속 공간으로 매핑하는 수단일 뿐이다.

embed discrete data into continuous space. We use this approach for the time-dependent Ornstein–Uhlenbeck experiment later.

**Training loss** The training losses used are the usual one for Wasserstein GANs (Goodfellow et al., 2014; Arjovsky et al., 2017). Let $Y_\theta \colon (V, W) \mapsto Y$ represent the overall action of the generator, and let $D_\phi \colon Y \mapsto D$ represent the overall action of the discriminator. Then the generator is optimised with respect to

$$\min_\theta \left[\mathbb{E}_{V,W} D_\phi(Y_\theta(V, W))\right], \tag{7}$$

and the discriminator is optimised with respect to

$$\max_\phi \left[\mathbb{E}_{V,W} D_\phi(Y_\theta(V, W)) - \mathbb{E}_{\pi} D_\phi(\hat{z})\right]. \tag{8}$$

Training is performed via stochastic gradient descent techniques as usual.

**Lipschitz regularisation** Wasserstein GANs need a Lipschitz discriminator, for which a variety of methods have been proposed. We use gradient penalty (Gulrajani et al., 2017), finding that neither weight clipping nor spectral normalisation worked (Arjovsky et al., 2017; Miyato et al., 2018).

We attribute this to the observation that neural SDEs (as with RNNs) have a recurrent structure. If a single step has Lipschitz constant $\lambda$, then the Lipschitz constant of the overall neural SDE will be $\mathcal{O}(\lambda^T)$ in the time horizon $T$. Even small positive deviations from $\lambda = 1$ may produce large Lipschitz constants. In contrast gradient penalty regularises the Lipschitz constant of the entire discriminator.

Training with gradient penalty implies the need for a double backward. If using the continuous-time adjoint equations of (Li et al., 2020), then this implies the need for a double-adjoint. Mathematically this is fine: however for moderate step sizes this produces gradients that are sufficiently inaccurate as to prevent models from training. For this reason we instead backpropagate through the internal operations of the solver.

**Learning any SDE** The Wasserstein metric has a unique global minima at $Y = Y_{\text{true}}$. By universal approximation of Neural CDEs (with respect to either continuous inputs or interpolated sequences, corresponding to dense and sparse data regimes respectively) (Kidger et al., 2020), the discriminator is sufficiently powerful to approximate the Wasserstein metric over any compact set of inputs.

Meanwhile by the universal approximation theorem for neural networks (Pinkus, 1999; Kidger & Lyons, 2020) and convergence results for SDEs (Friz & Victoir, 2010, Theorem 10.29) it is immediate that any (Markov) SDE of the form

$$\mathrm{d}Y_t = \mu(t, Y_t)\,\mathrm{d}t + \sigma(t, Y_t) \circ \mathrm{d}W_t$$

may be represented by the generator. Beyond this, the use of hidden state $X$ means that non-Markov dependencies may also be modelled by the generator. (This time without theoretical guarantees, however – we found that proving a formal statement hit theoretical snags.)

## 4. Experiments

We perform experiments across four datasets; each one is selected to represent a different regime. First is a univariate synthetic example to readily compare model results to the data. Second is a large-scale (14.6 million samples) dataset of Google/Alphabet stocks. Third is a conditional generative problem for air quality data in Beijing. Fourth is a dataset of weight evolution under SGD.

In all cases see Appendix A for details of hyperparameters, learning rates, optimisers and so on.

### 4.1. Synthetic example: time-dependent Ornstein–Uhlenbeck process

We begin by considering neural SDEs only (our other experiments feature comparisons to other models), and attempt to mimic a time-dependent one-dimensional Ornstein–Uhlenbeck process. This is an SDE of the form

$$\mathrm{d}z_t = (\mu t - \theta z_t)\,\mathrm{d}t + \sigma \circ \mathrm{d}W_t.$$

We let $\mu = 0.02, \theta = 0.1, \sigma = 0.4$, and generate 8192 samples from $t = 0$ to $t = 63$, sampled at every integer.

**Marginal distributions** We plot marginal distributions at $t = 6, 19, 32, 44, 57$. (Corresponding to 10%, 30%, 50%, 70% and 90% of the way along.) See Figure 3. We can visually confirm that the model has accurately recovered the true marginal distributions.

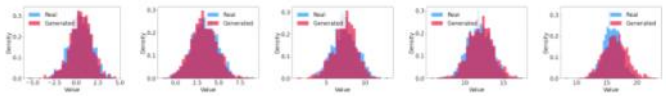

*Figure 3.* Left to right: marginal distributions at $t = 6, 19, 32, 44, 57$.

우리는 이후에 다룰 시간 종속적인 Ornstein–Uhlenbeck 실험에서 이 접근법을 사용한다.

**학습 손실(Training loss)**
학습에 사용된 손실 함수는 Wasserstein GANs에 일반적으로 사용되는 것이다 (Goodfellow et al., 2014; Arjovsky et al., 2017).

생성자의 전체 동작을 Y_θ : (V, W) ↦ Y로, 판별자의 전체 동작을 D_φ : Y ↦ D로 나타내자.
그러면 생성자는 다음과 같은 목적함수로 최적화된다:
min_θ E_{V, W} [D_φ (Y_θ (V, W))] (7)
그리고 판별자는 다음 목적함수로 최적화된다:

max_φ E_{V, W} [D_φ (Y_θ (V, W))] − E_{z} [D_φ (ẑ)] (8)

학습은 통상적인 확률적 경사 하강법(stochastic gradient descent) 기법을 통해 수행된다.

**Lipschitz 정규화(Lipschitz regularisation)**
Wasserstein GAN은 Lipschitz 연속인 판별자가 필요하며, 이를 위한 다양한 기법이 제안되어 왔다.
우리는 gradient penalty(그래디언트 패널티) 방법(Gulrajani et al., 2017)을 사용한다.
weight clipping이나 spectral normalization은 효과가 없음을 확인하였다 (Arjovsky et al., 2017; Miyato et al., 2018).
이유는 neural SDE가 (RNN처럼) 순환 구조(recurrent structure)를 가지기 때문이라고 판단된다.
한 단계의 Lipschitz 상수가 λ일 경우, 전체 neural SDE의 Lipschitz 상수는 시간 지평 T에 따라 O(λ^T)가 된다.
λ = 1에서 약간만 양의 편차가 생겨도, 전체 Lipschitz 상수는 매우 커질 수 있다.
반면, gradient penalty는 전체 판별자의 Lipschitz 상수를 정규화한다.
gradient penalty를 사용한 학습은 double backward(이차 도함수 계산)를 필요로 한다.
Li et al. (2020)의 연속시간 adjoint 방정식을 사용하는 경우, 이는 double-adjoint 계산이 필요함을 의미한다.
수학적으로는 문제가 없지만, 중간 크기의 스텝 사이즈에서는 그래디언트가 충분히 부정확해져서 학습을 방해할 수 있다.
이러한 이유로 우리는 solver의 내부 연산을 통해 직접 backpropagation을 수행한다2E

**임의의 SDE 학습하기(Learning any SDE)**
Wasserstein 거리(Wasserstein metric)는 Y = Y_true일 때 유일한 전역 최소값(global minimum)을 가진다.
Neural CDE는 (연속적인 입력 또는 보간된 시퀀스에 대해) 보편 근사성(universal approximation)을 가지므로 (Kidger et al., 2020),
판별자는 어떤 콤팩트한 입력 집합(compact set of inputs)에 대해서도 Wasserstein 거리를 근사할 수 있을 정도로 충분히 표현력이 높다.

한편, 신경망에 대한 보편 근사 정리(universal approximation theorem) (Pinkus, 1999; Kidger & Lyons, 2020)와 SDE에 대한 수렴 결과(Friz & Victoir, 2010, 정리 10.29)에 따르면, 다음과 같은 형태의 (마르코프적인) SDE:

dY_t = μ(t, Y_t)dt + σ(t, Y_t) ◦ dW_t

는 생성자(generator)를 통해 표현 가능하다는 것이 즉시 성립한다.
이보다 더 나아가, 숨겨진 상태(hidden state) X의 사용은 마르코프가 아닌(non−Markov) 의존성도 생성자가 모델링할 수 있게 한다.
(다만, 이 경우에는 이론적 보장은 존재하지 않는다. 우리는 이에 대해 형식적인 정리를 증명하려 했으나, 이론적 장애물에 부딪혔다.)

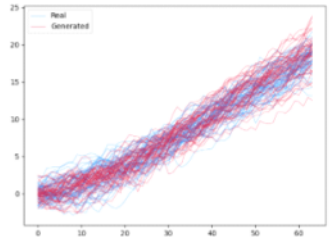


*Figure 4.* Sample paths from the time-dependent Ornstein–Uhlenbeck SDE, and from the neural SDE trained to match it.

**Sample paths** Next we plot 50 samples from the true distribution against 50 samples from the learnt distribution.

See Figure 4. Once again we see excellent agreement between the data and the model.

Overall we see that the neural SDEs are sufficient to recover classical non-neural SDEs: at least on this experiment, nothing has been lost in the generalisation.

### 4.2. Google/Alphabet stock prices

**Dataset** Next we consider a dataset consisting of Google/Alphabet stock prices, obtained from LOBSTER (Haase, 2013). The data consists of limit orders, in particular ask and bid prices.

A year of data corresponding to 2018–2019 is used, with an average of 605 054 observations per day. This is then downsampled and sliced into windows of length approximately one minute, for a total of approximately 14.6 million datapoints. We model the two-dimensional path consisting of the midpoint and the log-spread.

**Models** Here we compare against two recently-proposed and state-of-the-art competing neural differential equation models; specifically the Latent ODE model of Rubanova et al. (2019) and the continuous time flow process (CTFP) of Deng et al. (2020). The extended version of CTFPs, including latent variables, is used.

Between them these models cover several training regimes. Latent ODEs are trained as variational autoencoders; CTFPs are trained as normalising flows; neural SDEs are trained as GANs. (To the best of our knowledge neural SDEs as considered here are in fact the first model in their class, namely continuous-time GANs.)

**Performance metrics** We study three test metrics: classification, prediction, and MMD.

*Classification* is given by training an auxiliary model to distinguish real data from fake data. We use a neural CDE (Kidger et al., 2020) for the classifier. Larger losses, meaning inability to classify, indicate better performance of the generative model.

*Prediction* is a train-on-synthetic-test-on-real (TSTR) metric (Hyland et al., 2017). We train a sequence-to-sequence model to predict the latter part of a time series given the first part, using generated data. Testing is performed on real data. We use a neural CDE/ODE as an encoder/decoder pair. Smaller losses, meaning ability to predict, are better.

*Maximum mean discrepancy* is a distance between probability distributions with respect to a kernel or feature map. We use the depth-5 signature transform as the feature map (Király & Oberhauser, 2019; Toth & Oberhauser, 2020). Smaller values, meaning closer distributions, are better.

**Results** The results are shown in Table 1. We see that neural SDEs outperform both competitors in all metrics. Notably the Latent ODE fails completely on this dataset. We believe this reflects the fact the stochasticity inherent in the problem; this highlights the inadequacy of neural ODE-based modelling for such tasks, and the need for neural SDE-based modelling instead.

### 4.3. Air Quality in Beijing

Next we consider a dataset of the air quality in Beijing, from the UCI repository (Zhang et al., 2017; Dua & Graff, 2017). Each sample is a 6-dimensional time series of the $SO_2$, $NO_2$, CO, $O_3$, $PM_{2.5}$ and $PM_{10}$ concentrations, as they change over the course of a day.

We consider the same collection of models and performance statistics as before. We train this as a conditional

*Table 1.* Results for stocks dataset. Bold indicates best performance; mean $\pm$ standard deviation over three repeats.

| Metric | Neural SDE | CTFP | Latent ODE |
|---|---|---|---|
| Classification | **0.357 $\pm$ 0.045** | 0.165 $\pm$ 0.087 | 0.000239 $\pm$ 0.000086 |
| Prediction | **0.144 $\pm$ 0.045** | 0.725 $\pm$ 0.233 | 46.2 $\pm$ 12.3 |
| MMD | **1.92 $\pm$ 0.09** | 2.70 $\pm$ 0.47 | 60.4 $\pm$ 35.8 |

*Table 2.* Results for air quality dataset. Bold indicates best performance; mean ± standard deviation over three repeats.

| Metric | Neural SDE | CTFP | Latent ODE |
|---|---|---|---|
| Classification | 0.589 ± 0.051 | **0.764 ± 0.064** | 0.392 ± 0.011 |
| Prediction | **0.395 ± 0.056** | 0.810 ± 0.083 | 0.456 ± 0.095 |
| MMD | **0.000160 ± 0.000029** | 0.00198 ± 0.00001 | 0.000242 ± 0.000002 |

*Table 3.* Results for weights dataset. Bold indicates best performance; mean ± standard deviation over three repeats.

| Metric | Neural SDE | CTFP | Latent ODE |
|---|---|---|---|
| Classification | 0.507 ± 0.019 | **0.676 ± 0.014** | 0.0112 ± 0.0025 |
| Prediction | **0.00843 ± 0.00759** | 0.0808 ± 0.0514 | 0.127 ± 0.152 |
| MMD | **5.28 ± 1.27** | 12.0 ± 0.5 | 23.2 ± 11.8 |

generative problem, using class labels that correspond to 14 different locations the data was measured at. Class labels are additionally made available to the auxiliary models performing classification and prediction. See Table 2.

On this problem we observe that neural SDEs win on two out of the three metrics (prediction and MMD). CTFPs outperform neural SDEs on classification; however the CTFP severely underperforms on prediction. We believe this reflect the fact that CTFPs are strongly diffusive models; in contrast see how the drift-only Latent ODE performs relatively well on prediction. Once again this highlights the benefits of SDE-based modelling, with its combination of drift and diffusion terms.

### 4.4. Weights trained via SGD

Finally we consider a problem that is classically understood via (stochastic) differential equations: the weight updates when training a neural network via stochastic gradient descent with momentum. We train several small convolutional networks on MNIST (LeCun et al., 2010) for 100 epochs, and record their weights on every epoch. This produces a dataset of univariate time series; each time series corresponding to a particular scalar weight.

We repeat the comparisons of the previous section. Doing so we obtain the results shown in Table 3. Neural SDEs once again perform excellently. On this task we observe similar behaviour to the air quality dataset: the CTFP obtains a small edge on the classification metric, but the neural SDE outcompetes it by an order of magnitude on prediction, and by a factor of about two on the MMD. Latent ODEs perform relatively poorly in comparison to both.

### 4.5. Successfully training neural SDEs

As a result of our experiments, we empirically observed that successful training of neural SDEs was predicated on several factors.

**Final tanh nonlinearity** Using a final tanh nonlinearity (on both drift and diffusion, for both generator and discriminator) constrains the rate of change of hidden state. This avoids model blow-up as in Kidger et al. (2020).

**Stochastic weight averaging** Using the Cesàro mean of both the generator and discriminator weights, averaged over training, improves performance in the final model (Yazıcı et al., 2019). This averages out the oscillatory training behaviour for the min-max objective used in GAN training.

**Adadelta** We experimented with several different standard optimisers, in particular including SGD, Adadelta (Zeiler, 2012) and Adam (Kingma & Ba, 2015). Amongst all optimisers considered, Adadelta produced substantially better performance. We do not have an explanation for this.

**Weight decay** Nonzero weight decay also helped to damp the oscillatory behaviour resulting from the min-max objective used in GAN training.

## 5. Conclusion

By coupling together a neural SDE and a neural CDE as a generator/discriminator pair, we have shown that neural SDEs may be trained as continuous time GANs. Moreover we have shown that this approach extends the existing classical approach to SDE modelling – using prespecified payoff functions – so that it may be integrated into existing SDE modelling workflows. Overall, we have demonstrated the capability of neural SDEs as a means of modelling distributions over path space.

### Acknowledgements

PK was supported by the EPSRC grant EP/L015811/1. JF was supported by the EPSRC grant EP/N509711/1. PK, JF, HO, TL were supported by the Alan Turing Institute

**결론**

Neural SDE와 Neural CDE를 생성자/판별자 쌍으로 결합함으로써, 우리는 Neural SDE가 연속 시간 생성적 적대 신경망(GAN)으로 학습될 수 있음을 보였다. 더 나아가, 이 접근법이 기존의 고전적 SDE 모델링 방식—미리 정의된 수익 함수(payoff functions)를 사용하는 방식—을 확장하며, 기존 SDE 모델링 워크플로우에 통합될 수 있음을 입증하였다. 전반적으로, 우리는 Neural SDE가 경로 공간(path space) 위의 분포를 모델링하는 수단으로서의 가능성을 입증하였다.

under the EPSRC grant EP/N510129/1. PK thanks Penny Drinkwater for advice on Figure 2.

## References


Arató, M. A famous nonlinear stochastic equation (Lotka-Volterra model with diffusion). *Mathematical and Computer Modelling*, 38(7–9):709–726, 2003.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein Generative Adversarial Networks. volume 70 of *Proceedings of Machine Learning Research*, pp. 214–223, International Convention Centre, Sydney, Australia, 2017. PMLR.

Bai, S., Kolter, J., and Koltun, V. Deep equilibrium models. In *Advances in Neural Information Processing Systems 32*, pp. 690–701. Curran Associates, Inc., 2019.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD GANs. In *International Conference on Learning Representations*, 2018.

Black, F. and Scholes, M. The Pricing of Options and Corporate Liabilities. *Journal of Political Economy*, 81(3): 637–654, 1973.

Brigo, D. and Mercurio, F. *Interest Rate Models: Theory and Practice*. Springer, Berlin, 2001.

Briol, F.-X., Barp, A., Duncan, A., and Girolami, M. Statistical Inference for Generative Models with Maximum Mean Discrepancy. *arXiv:1906.05944*, 2020.

Chen, R. T. Q. torchdiffeq, 2018. https://github.com/rtqichen/torchdiffeq.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31*, pp. 6571–6583. Curran Associates, Inc., 2018.

Coffey, W. T., Kalmykov, Y. P., and Waldron, J. T. *The Langevin Equation: With Applications to Stochastic Problems in Physics, Chemistry and Electrical Engineering*. World Scientifc, 2012.

Cox, J. C., Ingersoll, J. E., and Ross, S. A. A theory of term structure of interest rates. *Econometrica*, 53(2):385–407, 1985.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Cuchiero, C., Khosrawi, W., and Teichmann, J. A generative adversarial network approach to calibration of local stochastic volatility models. *Risks*, 8(4), 2020.

Deng, R., Chang, B., Brubaker, M., Mori, G., and Lehrmann, A. Modeling Continuous Stochastic Processes with Dynamic Normalizing Flows. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7805–7815. Curran Associates, Inc., 2020.

Dua, D. and Graff, C. UCI Machine Learning Repository, 2017. URL http://archive.ics.uci.edu/ml.

Friz, P. K. and Victoir, N. B. *Multidimensional stochastic processes as rough paths: theory and applications*. Cambridge University Press, 2010.

Futoma, J., Hariharan, S., and Heller, K. Learning to Detect Sepsis with a Multitask Gaussian Process RNN Classifier. *Proceedings of the 34th International Conference on Machine Learning*, pp. 1174–1182, 2019.

Gierjatowicz, P., Sabate-Vidales, M., Šiška, D., Szpruch, L., and Žurič, Ž. Robust Pricing and Hedging via Neural SDEs. *arXiv:2007.04154*, 2020.

Giles, M. and Glasserman, P. Smoking Adjoints. *Risk*, 2006.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, pp. 2672–2680. Curran Associates, Inc., 2014.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations*, 2019.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773, 2013.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems 30*, pp. 5767–5777. Curran Associates, Inc., 2017.

Haase, J. Limit order book system – the efficient reconstructor, 2013. URL https://lobsterdata.com/.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020.

Hodgkinson, L., van der Heide, C., Roosta, F., and Mahoney, M. Stochastic Normalizing Flows. *arXiv:2002.09547*, 2020.

Huillet, T. On Wright-Fisher diffusion and its relatives. *Journal of Statistical Mechanics: Theory and Experiment*, 11, 2007.

Hyland, S. L., Esteban, C., and Rätsch, G. Real-Valued (Medical) Time Series Generation with Recurrent Conditional GANs. *arXiv:1706.02633*, 2017.

Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., and Tebbutt, W. A Differentiable Programming System to Bridge Machine Learning and Scientific Computing. *arXiv:1907.07587*, 2019.

Karatzas, I. and Shreve, S. *Brownian Motion and Stochastic Calculus*. Graduate Texts in Mathematics. Springer New York, 1991.

Kidger, P. `torchcde`, 2020. https://github.com/patrick-kidger/torchcde.

Kidger, P. and Lyons, T. Universal Approximation with Deep Narrow Networks. *COLT 2020*, 2020.

Kidger, P. and Lyons, T. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. *International Conference on Learning Representations*, 2021. URL https://github.com/patrick-kidger/signatory.

Kidger, P., Morrill, J., Foster, J., and Lyons, T. Neural Controlled Differential Equations for Irregular Time Series. *arXiv:2005.08926*, 2020.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Király, F. and Oberhauser, H. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 2019.

LeCun, Y., Cortes, C., and Burges, C. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

Lelièvre, X. and Stoltz, G. Partial differential equations and stochastic methods in molecular dynamics. *Acta Numerica*, 25:681–880, 2016.

Levin, D., Lyons, T., and Ni, H. Learning from the past, predicting the statistics for the future, learning an evolving system. *arXiv 1309.0260*, 2013.

Li, C.-L., Chang, W.-C., Cheng, Y., Yang, Y., and Poczos, B. MMD GAN: Towards Deeper Understanding of Moment Matching Network. In *Advances in Neural Information Processing Systems*, volume 30, pp. 2203–2213. Curran Associates, Inc., 2017.

Li, S. C.-X. and Marlin, B. M. A scalable end-to-end Gaussian process adapter for irregularly sampled time series classification. In *Advances in Neural Information Processing Systems*, pp. 1804–1812. Curran Associates, Inc., 2016.

Li, X. `torchsde`, 2020. https://github.com/google-research/torchsde.

Li, X., Wong, T.-K. L., Chen, R. T. Q., and Duvenaud, D. Scalable Gradients and Variational Inference for Stochastic Differential Equations. *AISTATS*, 2020.

Li, Y., Swersky, K., and Zemel, R. Generative Moment Matching Networks. In *Proceedings of the 32nd International Conference on Machine Learning*. 2015.

Liu, X., Xiao, T., Si, S., Cao, Q., Kumar, S., and Hsieh, C.-J. Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise. *arXiv:1906.02355*, 2019.

Massaroli, S., Poli, M., Park, J., Yamashita, A., and Asama, H. Dissecting Neural ODEs. *arXiv:2002.08071*, 2020.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral Normalization for Generative Adversarial Networks. In *International Conference on Learning Representations*, 2018.

Morrill, J., Kidger, P., Salvi, C., Foster, J., and Lyons, T. Neural CDEs for Long Time-Series via the Log-ODE Method. *arXiv:2009.08295*, 2020.

Norcliffe, A., Bodnar, C., Day, B., Simidjievski, N., and Lió, P. On Second Order Behaviourin Augmented Neural ODEs. *arXiv:2006.07220*, 2020.

Oganesyan, V., Volokhova, A., and Vetrov, D. Stochasticity in Neural ODEs: An Empirical Study. *arXiv:2002.09779*, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

Pavliotis, G. A. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*. Springer, New York, 2014.

Pinkus, A. Approximation theory of the MLP model in neural networks. *Acta Numer.*, 8:143–195, 1999.

Rackauckas, C., Ma, Y., Martensen, J., Warner, C., Zubov, K., Supekar, R., Skinner, D., and Ramadhan, A. Universal Differential Equations for Scientific Machine Learning. *arXiv:2001.04385*, 2020.

Revuz, D. and Yor, M. *Continuous martingales and Brownian motion*, volume 293. Springer Science & Business Media, 2013.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1530–1538, Lille, France, 2015. PMLR.

Rogers, L. and Williams, D. *Diffusions, Markov Processes and Martingales: Volume 2, Itô Calculus*. Cambridge Mathematical Library. Cambridge University Press, 2000.

Rubanova, Y., Chen, R. T. Q., and Duvenaud, D. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *Advances in Neural Information Processing Systems 32*, pp. 5320–5330. Curran Associates, Inc., 2019.

Soboleva, T. K. and Pleasants, A. B. Population Growth as a Nonlinear Stochastic Process . *Mathematical and Computer Modelling*, 38(11–13):1437–1442, 2003.

Song, Y. and Ermon, S. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, volume 32, pp. 11918–11930. Curran Associates, Inc., 2019.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

Toth, C. and Oberhauser, H. Variational Gaussian Processes with Signature Covariances. *ICML 2020*, 2020.

Tzen, B. and Raginsky, M. Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit. *arXiv:1905.09883*, 2019a.

Tzen, B. and Raginsky, M. Theoretical guarantees for sampling and inference in generative models with latent diffusions. *COLT*, 2019b.

Yazıcı, Y., Foo, C.-S., Winkler, S., Yap, K.-H., Piliouras, G., and Chandrasekhar, V. The unusual effectiveness of averaging in GAN training. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=SJgw_sRqFQ.

Zeiler, M. D. ADADELTA: An Adaptive Learning Rate Method. *arXiv 1212.5701*, 2012.

Zhang, S., Guo, B., Dong, A., He, J., Xu, Z., and Chen, S. X. Cautionary Tales on Air-Quality Improvement in Beijing. *Proceedings of the Royal Society A*, 473(2205), 2017.