SIAM Review
Vol. 67, No. 3, pp. 607–623

Diffusion Models for Generative Artificial Intelligence: An Introduction for Applied Mathematicians*

Catherine Higham†
Desmond J. Higham‡
Peter Grindrod§

Abstract. Generative artificial intelligence (GAI) refers to algorithms that create synthetic but realistic output. Diffusion models currently offer state-of-the-art performance in GAI for images. They also form a key component in more general tools, including text-to-image generators and large language models. Diffusion models work by adding noise to the available training data and then learning how to reverse the process. The reverse operation may then be applied to new random data in order to produce new outputs. We provide a brief introduction to diffusion models for applied mathematicians and statisticians. Our key aims are to (a) present illustrative computational examples, (b) give a careful derivation of the underlying mathematical formulas involved, and (c) draw a connection with partial differential equation (PDE) diffusion models. We provide code for the computational experiments. We hope that this topic will be of interest to advanced undergraduate and postgraduate students. Portions of the material may also provide useful motivational examples for those who teach courses in stochastic processes, inference, machine learning, PDEs, or scientific computing.

Key words. denoising, deep learning, Markov process, computer vision

MSC codes. 68T05, 68T45, 60J60

DOI. 10.1137/23M1626232

1. Motivation.

Generative artificial intelligence (GAI) models are designed to create new outputs that are similar to the examples on which they were trained. Over the past decade or so, advancements in GAI have included the development of variational autoencoders [9, 23], generative adversarial networks [19], and transformers [40]. In this work we focus on denoising diffusion probabilistic models [15]; for simplicity we use the term "diffusion models." They currently represent the state of the art in image generation [7] and form a key part of more sophisticated tools such as

*Received by the editors December 22, 2023; accepted for publication (in revised form) September 17, 2024; published electronically August 7, 2025.
https://doi.org/10.1137/23M1626232
Funding: The work of the first author was supported by EPSRC grant EP/T00097X/1. The work of the second author was supported by EPSRC grant EP/V046527/1. The work of the third author was supported by EPSRC grant EP/R018472/1.

†School of Computing Science, University of Glasgow, Glasgow, G12 8RZ, United Kingdom (Catherine.Higham@glasgow.ac.uk).
‡School of Mathematics, University of Edinburgh, Edinburgh, EH9 3FD, United Kingdom (d.j.higham@ed.ac.uk).
§Mathematical Institute, University of Oxford, Oxford, United Kingdom (grindrod@maths.ox.ac.uk).

DALL-E 2 and 3 [32]. We refer to [4, 6, 8, 30], and the references therein, for details of the historical developments that have led to the current state of the art in GAI.

The somewhat counterintuitive but deceptively powerful idea behind diffusion models is to destroy the training data by adding noise. While doing so, the model learns how to reverse the process. In this way, the final model is able to start with new, easily generated, random samples and denoise them, thereby generating new, synthetic examples. The task of building and applying a simple yet impressive model can be described very succinctly—see Algorithms 5.1 and 5.2 in section 5. However, deriving the expressions that go into these algorithms is not so straightforward, and we believe that there is a niche for a careful and accessible mathematically-oriented treatment.

Our intended readership is advanced undergraduate and postgraduate students in mathematics or related disciplines. The material should be suitable for independent study, and there are many directions in which it can be followed up—the literature is rapidly expanding, and new extensions and connections are being discovered at pace. We also hope that portions of this material will provide higher education professionals with topical and engaging examples that can be slipped into courses on stochastics, numerics, partial differential equations (PDEs), or data science.

We aim to keep the prerequisites (given below) to a minimum:

- For sections 2–5, ideas from statistics: mean, variance, Gaussian distribution, Markov chains, conditional probability.
- For sections 4–5, ideas from deep learning: the stochastic gradient method, artificial neural networks.
- For section 6, ideas from PDEs: multivariate calculus, the divergence theorem, spectral analysis.

We focus here on the task of image generation. We describe a bare bones form of a diffusion model, explain carefully how the key mathematical expressions arise, and illustrate the concept via computational examples. The key reference for this article is [15], which built on [36] and is currently receiving more than 100 citations per day. We also found [26] to be a very useful resource.

In section 2 we present some pictures that give a feel for the idea of diffusion models in GAI. We then provide details of the relevant forward and backward processes in sections 3 and 4, respectively, which leads to the algorithms presented in section 5. We continue in section 6 with more speculative material that suggests a connection between stable diffusion models and deterministic PDEs, providing a link to more traditional applied mathematics.

We emphasize that this is a very active and fast-moving research topic with connections to many related areas. In section 7 we provide some links to the relevant literature. That section also highlights wider issues around performance evaluation, computational expense, copyright, privacy, ethics, bias, explainability, and robustness.

2. Illustration. A diffusion model [15] aims to generate realistic-looking images and works by

(i) taking an existing image and iteratively adding noise until the original information is lost;

(ii) learning how to reconstruct the original image by iteratively removing the noise.

After training, we can then use the reverse diffusion process to generate a realistic image from a new, random, starting point—we remove the noise and see what emerges.

One way to conceptualize this method is to imagine an (unknown) probability distribution over the collection of all natural images. We hope to sample from this distribution; more likely images should be chosen with higher probability. We don't have access to this magic probability distribution, but instead we have training data; that is, examples of natural images. We also have a pseudorandom number generator that allows us to sample from a standard Gaussian distribution. In item (i) above, we are doing the easy part, mapping from the image distribution to the Gaussian distribution. In item (ii) we learn the inverse operator, mapping from the Gaussian distribution to the image distribution. This allows us to convert Gaussian samples into images.

We illustrate the idea using a diffusion model trained on images from the widely studied MNIST data set [24]. Here, each image represents a handwritten digit from the set $\{0, 1, 2, \ldots, 9\}$. These low-resolution images are black and white with 28×28 pixels, resized by the model to 32×32. Figure 1 shows a representative collection of 64 images.

Figures 2–4 were produced with a diffusion model based on a Mathworks tutorial at https://uk.mathworks.com/help/deeplearning/ug/generate-images-using-diffusion.html. Figure 2 illustrates the forward process that is used in the training phase. At time $t = 0$ we have an MNIST image. At each integer time $t = 0, 1, 2, \ldots, 499$, Gaussian noise is added. At $t = 500$ there is no visible evidence of the original image.

Figure 3 shows the effect of the backward process that is available after training. The top left panel displays nine randomly chosen final time $t = 500$ images—pure noise matrices consisting of independent Gaussian samples. We show the effect of applying the backward, denoising process as time is reversed. At $t = 0$ the model has produced new, synthetic examples that, in at least eight of the nine cases, correspond to handwritten digits. We emphasize that labels were not used in the training process. In this simple, unconditional model there is no way to control which (if any) of the $t = 0$ images will resemble any particular category of digit.

In Figure 4 we show the results from a larger experiment. Here we used the trained diffusion model to generate images from 500 independent time $t = 500$ choices. For this figure, we separated the images into categories using an independent convolutional neural network classifier that was trained separately on real MNIST data. Since we

Fig. 1 *Representative set of* 64 *images from the MNIST data set* [24].

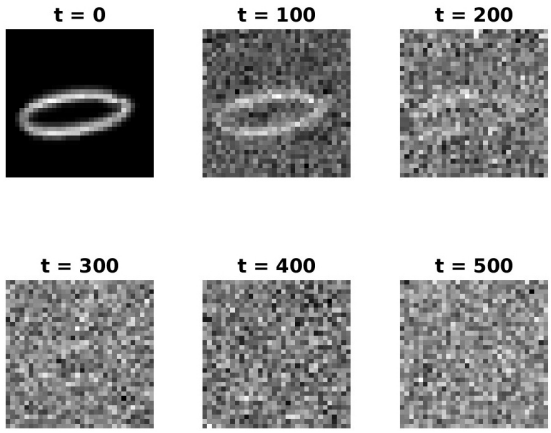

Fig. 2 *Result of forward map noising over time.*

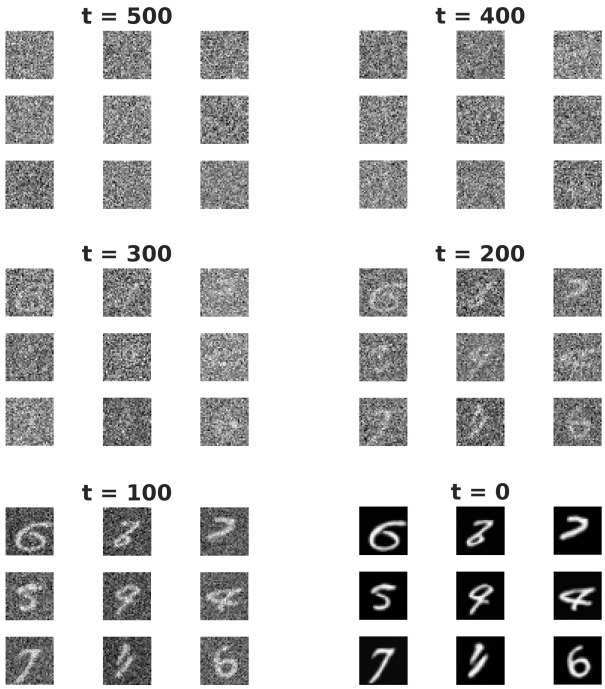

Fig. 3 *Result of backward map denoising over time for* 9 *different random choices at* $t = 500$.

have no control over how many of the 500 images will appear in each class, the number of synthetic outputs in each category varies considerably.

We finish with two experiments that illustrate that the backward, denoising process is both stochastic and unpredictable. In Figure 5 we show the images generated after applying nine independent denoising runs to the same Gaussian at $t = 500$. We see that the denoising process can produce considerably different results from a single source of randomness. In Figure 6 we perform a similar experiment where the

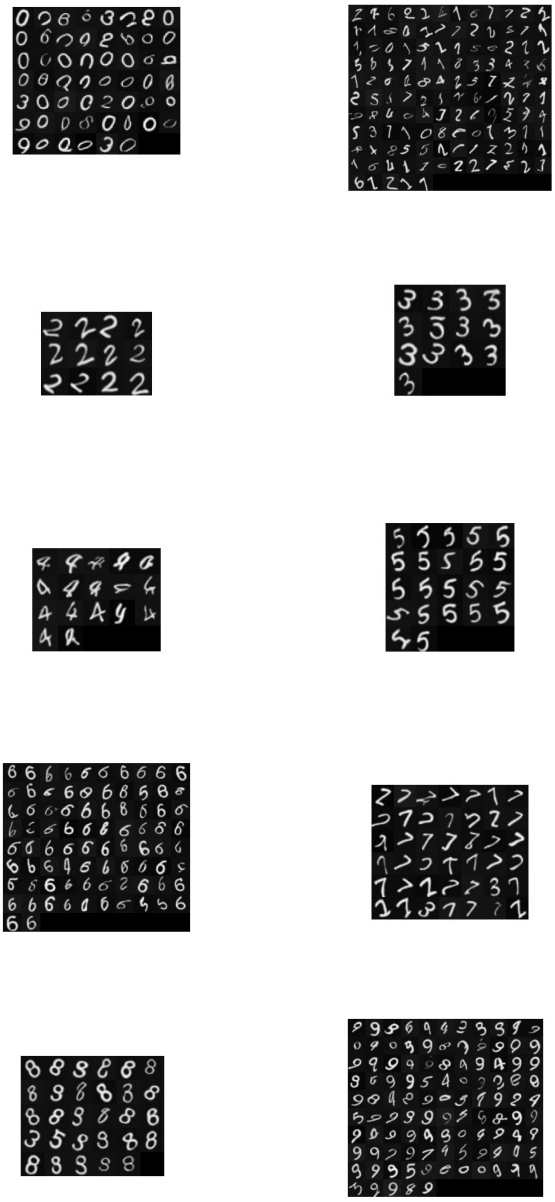

Fig. 4 500 *synthetic images generated by the diffusion model from different time* $t = 500$ *noise samples. These have been sorted into classes by a convolutional neural network classifier that was trained separately on real MNIST data.*

$t = 500$ data emerges from the training set. On the left we show a training image undergoing the forward, noising process up to time $t = 500$. On the right we show the results from nine independent denoising runs on this time $t = 500$ data. We see that none of the synthetically generated images resembles the original.

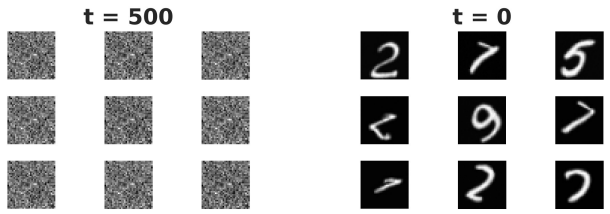

Fig. 5 *Nine images (right) created from the same Gaussian noise at $t = 500$ (left).*

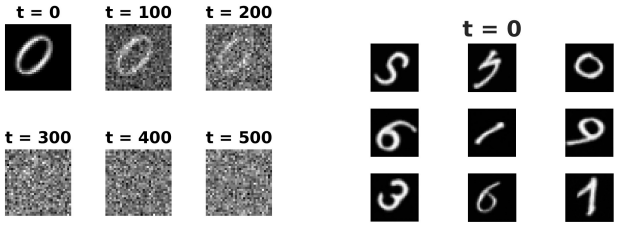

Fig. 6 *Nine images (right) created from the noisy version (left $t = 500$) of one original image (left $t = 0$).*

3. Forwards. We begin this section with some background on Gaussian random variables; see a standard text such as [3, 28] for more details. When dealing with Gaussians, we will always consider the multivariate, isotropic case. We denote the probability density at a point $\mathbf{x} \in \mathbb{R}^d$ by $\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \sigma^2 \mathbf{I})$, where

$$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \sigma^2 \mathbf{I}) := \frac{1}{(2\pi)^{d/2} \sigma^d} \exp\left(-\frac{1}{2\sigma^2} (\mathbf{x} - \boldsymbol{\mu})^T (\mathbf{x} - \boldsymbol{\mu}) \right). \tag{3.1}$$

Here, $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean, and we will refer to σ^2 as the variance, since the corresponding covariance matrix has the form $\sigma^2 \mathbf{I}$, with $\mathbf{I} \in \mathbb{R}^{d \times d}$ denoting the identity matrix. Such Gaussian random variables have the important property that their sums remain Gaussian, with means and variances combining additively: the sum of two independent Gaussians with means $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ and variances σ_1^2 and σ_2^2 is a Gaussian random variable with mean $\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2$ and variance $\sigma_1^2 + \sigma_2^2$. The term "standard Gaussian" refers to the case where the mean is $\mathbf{0} \in \mathbb{R}^d$ and the variance is 1. Multiplying a standard Gaussian by the scalar σ and shifting by $\boldsymbol{\mu} \in \mathbb{R}^d$ produces a Gaussian with mean $\boldsymbol{\mu}$ and variance σ^2. It follows that if $\mathbf{y}$ and $\mathbf{z}$ are independent standard Gaussians and a and b are scalars, then $a\mathbf{y} + b\mathbf{z}$ is Gaussian with mean zero and variance $a^2 + b^2$, so $a\mathbf{y} + b\mathbf{z}$ can be sampled as $\sqrt{a^2 + b^2}\, \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon}$ is a standard Gaussian.

We consider images that can be described by d real numbers, typically pixel values, and we collect these into a vector in $\mathbb{R}^d$. In practice, pixel values might be constrained—for example, only integers between 0 and 255 might be allowed—but we ignore this issue here for simplicity.

Given an image $\mathbf{x}_0 \in \mathbb{R}^d$, the *forward process* iteratively adds noise to create a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ according to the rule

$$\mathbf{x}_t = \sqrt{1 - \beta_t}\, \mathbf{x}_{t-1} + \sqrt{\beta_t}\, \boldsymbol{\epsilon}_t. \tag{3.2}$$

Here, each $\boldsymbol{\epsilon}_t$ is an independent standard Gaussian and the scalar parameter β_t is between zero and one. The sequence $\beta_1, \beta_2, \ldots, \beta_T$, known as the *variance schedule,* is predetermined. For example, in [15], linearly increasing values from $\beta_1 = 10^{-4}$ to $\beta_T = 0.02$ are used. Since β_t here is increasing, more noise is added as the forward process evolves. It is useful to think of t as a time-like variable. At time zero we have an image and at time T we effectively have pure Gaussian noise.

The process (3.2) defines a discrete time Markov process, and the associated transition density may be written as

$$q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t \mathbf{I}). \tag{3.3}$$

This quantifies the probability of observing $\mathbf{x}_t$ at time t, given $\mathbf{x}_{t-1}$ at time $t-1$.

Updating over one time step in the forward process (3.2) is straightforward; just scale the current value and add Gaussian noise. For later use, it is helpful to know that stepping from time zero to a general time t is possible with a single leap. To see this, we introduce $\alpha_t = 1 - \beta_t$ so that

$$\mathbf{x}_t = \sqrt{\alpha_t}\,\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}_t. \tag{3.4}$$

Then, applying (3.4) again, we have

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t}(\sqrt{\alpha_{t-1}}\,\mathbf{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\,\boldsymbol{\epsilon}_{t-1}) + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}_t \\ &= \sqrt{\alpha_t\alpha_{t-1}}\,\mathbf{x}_{t-2} + \sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\,\boldsymbol{\epsilon}_{t-1} + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}_t. \end{aligned} \tag{3.5}$$

Using the properties of Gaussians mentioned at the start of this section, we see that $\sqrt{\alpha_t}\sqrt{1-\alpha_{t-1}}\,\boldsymbol{\epsilon}_{t-1} + \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}_t$ can be combined into a single Gaussian. In this way, (3.5) may be written

$$\mathbf{x}_t = \sqrt{\alpha_t\alpha_{t-1}}\,\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\,\boldsymbol{\epsilon}_{t,t-2},$$

where $\boldsymbol{\epsilon}_{t,t-2}$ is a standard Gaussian.

Proceeding inductively, suppose that for some k between $t-2$ and 1,

$$\mathbf{x}_t = \sqrt{\alpha_t\alpha_{t-1}\ldots\alpha_{k+1}}\,\mathbf{x}_k + \sqrt{1-\alpha_t\alpha_{t-1}\ldots\alpha_{k+1}}\,\epsilon_{t,k}, \tag{3.6}$$

where $\epsilon_{t,k}$ is a standard Gaussian. Then, replacing $\mathbf{x}_k$ using (3.4),

$$\mathbf{x}_t = \sqrt{\alpha_t\alpha_{t-1}\ldots\alpha_{k+1}}\left(\sqrt{\alpha_k}\,\mathbf{x}_{k-1} + \sqrt{1-\alpha_k}\,\boldsymbol{\epsilon}_k\right) + \sqrt{1-\alpha_t\alpha_{t-1}\ldots\alpha_{k+1}}\,\epsilon_{t,k}.$$

Again replacing the sum of two independent Gaussians by a single, appropriate Gaussian, we have

$$\begin{aligned} \mathbf{x}_t &= \sqrt{\alpha_t\alpha_{t-1}\ldots\alpha_k}\,\mathbf{x}_{k-1} + \sqrt{\alpha_t\alpha_{t-1}\ldots\alpha_{k+1}(1-\alpha_k) + 1 - \alpha_t\alpha_{t-1}\ldots\alpha_{k+1}}\,\epsilon_{t,k-1} \\ &= \sqrt{\alpha_t\alpha_{t-1}\ldots\alpha_k}\,\mathbf{x}_{k-1} + \sqrt{1-\alpha_t\alpha_{t-1}\ldots\alpha_k}\,\epsilon_{t,k-1}, \end{aligned}$$

where $\epsilon_{t,k-1}$ is a standard Gaussian. Hence, the form (3.6) is valid all the way down to $k = 0$. So, letting

$$\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_i, \tag{3.7}$$

we may write

$$\mathbf{x}_t = \sqrt{\overline{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\,\bar{\boldsymbol{\epsilon}}_t, \tag{3.8}$$

where $\bar{\boldsymbol{\epsilon}}_t$ is a standard Gaussian. We may therefore step directly from time 0 to any later time t using a single Gaussian. This proves convenient for the analysis in section 4 and also for the training algorithm discussed in section 5.

In terms of a transition density, (3.8) shows that

$$q(\mathbf{x}_t \,|\, \mathbf{x}_0) := \mathcal{N}(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\,\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I}). \tag{3.9}$$

4. Backwards. We now consider the reverse process. We are interested in the probability of $\mathbf{x}_{t-1}$ given $\mathbf{x}_t$ and $\mathbf{x}_0$; that is, $q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, \mathbf{x}_0)$. To proceed we will make use of a result in conditional probability theory known as the product rule [3, 28], which for our purposes may be written

$$P(A,B,C) = P(A \,|\, B,C)\, P(B,C) = P(A \,|\, B,C)\, P(B \,|\, C)\, P(C).$$

By symmetry, we also have

$$P(A,B,C) = P(B,A,C) = P(B \,|\, A,C)\, P(A,C) = P(B \,|\, A,C)\, P(A \,|\, C)\, P(C).$$

Hence,

$$P(A \,|\, B,C) = \frac{P(B \,|\, A,C)\, P(A \,|\, C)}{P(B \,|\, C)}.$$

We will use this in the form

$$q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{x}_0)\, q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_0)}{q(\mathbf{x}_t \,|\, \mathbf{x}_0)}. \tag{4.1}$$

Now we focus on the quantities appearing on the right-hand side of (4.1).

By the Markovian nature of the forward process, from (3.3),

$$q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}, \mathbf{x}_0) = q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\,\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}). \tag{4.2}$$

Making use of (3.9) for $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$, we then see that

$$q(\mathbf{x}_{t-1} \,|\, \mathbf{x}_t, \mathbf{x}_0) = \frac{\mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\,\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I})\,\mathcal{N}(\mathbf{x}_{t-1}; \sqrt{\overline{\alpha}_{t-1}}\,\mathbf{x}_0, (1-\overline{\alpha}_{t-1})\mathbf{I})}{\mathcal{N}(\mathbf{x}_t; \sqrt{\overline{\alpha}_t}\,\mathbf{x}_0, (1-\overline{\alpha}_t)\mathbf{I})}. \tag{4.3}$$

From the definition (3.1), and ignoring the normalizing constants, we see that this expression has the form

$$\exp\left(-\frac{1}{2}\frac{(\mathbf{x}_t - \sqrt{\alpha_t}\,\mathbf{x}_{t-1})^T(\mathbf{x}_t - \sqrt{\alpha_t}\,\mathbf{x}_{t-1})}{1-\alpha_t} - \frac{1}{2}\frac{(\mathbf{x}_{t-1} - \sqrt{\overline{\alpha}_{t-1}}\,\mathbf{x}_0)^T(\mathbf{x}_{t-1} - \sqrt{\overline{\alpha}_{t-1}}\,\mathbf{x}_0)}{1-\overline{\alpha}_{t-1}}\right.$$
$$\left. + \frac{1}{2}\frac{(\mathbf{x}_t - \sqrt{\overline{\alpha}_t}\mathbf{x}_0)^T(\mathbf{x}_t - \sqrt{\overline{\alpha}_t}\,\mathbf{x}_0)}{1-\overline{\alpha}_t}\right). \tag{4.4}$$

We will show that this expression matches

$$\mathcal{N}\left(\mathbf{x}_{t-1}; \mu_q(\mathbf{x}_t, \mathbf{x}_0), \sigma_q^2(t)\mathbf{I}\right) \tag{4.5}$$

for appropriate $\mu_q(\mathbf{x}_t, \mathbf{x}_0)$ and $\sigma_q^2(t)$. From (3.1), we can find $\sigma_q^2(t)$ by considering the coefficient of $-\mathbf{x}_{t-1}^T\mathbf{x}_{t-1}$ in the exponent of (4.4). This coefficient is given by

$$\frac{1}{2}\frac{\alpha_t}{1-\alpha_t} + \frac{1}{2}\frac{1}{1-\overline{\alpha}_{t-1}} = \frac{1}{2}\frac{\alpha_t(1-\overline{\alpha}_{t-1}) + 1 - \alpha_t}{(1-\alpha_t)(1-\overline{\alpha}_{t-1})} = \frac{1}{2}\left(\frac{1-\overline{\alpha}_t}{(1-\alpha_t)(1-\overline{\alpha}_{t-1})}\right),$$

where we used $\alpha_t\,\overline{\alpha}_{t-1} = \overline{\alpha}_t$ from (3.7). Hence,

$$\sigma_q^2(t) = \frac{(1-\alpha_t)(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}. \tag{4.6}$$

Using the functional form (3.1) again, we can find $\mu_q(\mathbf{x}_t, \mathbf{x}_0)$ by considering the vector, say, $\mathbf{v}$, such that $\mathbf{x}_{t-1}^T\mathbf{v}$ is the cross-product in the exponent of (4.4). We see that

$$\frac{\mu_q(\mathbf{x}_t, \mathbf{x}_0)}{\sigma_q^2(t)} = \mathbf{v} = \frac{\sqrt{\alpha_t}\,\mathbf{x}_t}{1-\alpha_t} + \frac{\sqrt{\overline{\alpha}_{t-1}}\,\mathbf{x}_0}{1-\overline{\alpha}_{t-1}}.$$

Hence, using (4.6),

$$\mu_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}\,(1-\overline{\alpha}_{t-1})\mathbf{x}_t + \sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)\mathbf{x}_0}{1-\overline{\alpha}_t}. \tag{4.7}$$

We wish to compute a sample from the distribution in (4.5). This will allow us to perform the required transition along the backwards process. Our approach is to estimate the mean in (4.5) and then shift with an appropriate Gaussian in order to match the required variance.

If we know $\mathbf{x}_t$ and $\bar{\boldsymbol{\epsilon}}_t$ in (3.8), then we may write

$$\mathbf{x}_0 = \frac{\mathbf{x}_t - \sqrt{1-\overline{\alpha}_t}\,\bar{\boldsymbol{\epsilon}}_t}{\sqrt{\overline{\alpha}_t}}.$$

Substituting this expression for $\mathbf{x}_0$ into (4.7), we see that the mean of $\mathbf{x}_{t-1}$, given $\mathbf{x}_t$ and $\mathbf{x}_0$, takes the form

$$\mu_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t}\mathbf{x}_t + \frac{\sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)}{(1-\overline{\alpha}_t)\sqrt{\overline{\alpha}_t}}\mathbf{x}_t - \frac{\sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)\sqrt{1-\overline{\alpha}_t}}{(1-\overline{\alpha}_t)\sqrt{\overline{\alpha}_t}}\,\bar{\boldsymbol{\epsilon}}_t. \tag{4.8}$$

Noting from (3.7) that $\overline{\alpha}_{t-1}/\overline{\alpha}_t = 1/\alpha_t$ and $\alpha_t \times \overline{\alpha}_{t-1} = \overline{\alpha}_t$, we find that in (4.8) the coefficient of $\mathbf{x}_t$ simplifies as follows:

$$\frac{\sqrt{\alpha_t}(1-\overline{\alpha}_{t-1})}{1-\overline{\alpha}_t} + \frac{\sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)}{(1-\overline{\alpha}_t)\sqrt{\overline{\alpha}_t}} = \frac{1}{\sqrt{\alpha_t}(1-\overline{\alpha}_t)}\left(\alpha_t(1-\overline{\alpha}_{t-1}) + 1 - \alpha_t\right) = \frac{1}{\sqrt{\alpha_t}}.$$

Similarly, the coefficient of $\bar{\boldsymbol{\epsilon}}_t$ in (4.8) simplifies to

$$-\frac{\sqrt{\overline{\alpha}_{t-1}}(1-\alpha_t)\sqrt{1-\overline{\alpha}_t}}{(1-\overline{\alpha}_t)\sqrt{\overline{\alpha}_t}} = -\frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\overline{\alpha}_t}}.$$

Hence, (4.8) may be written

$$\mu_q(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\bar{\boldsymbol{\epsilon}}_t\right). \tag{4.9}$$

The missing ingredient here is $\bar{\boldsymbol{\epsilon}}_t$—the noise that drove the transition from $\mathbf{x}_0$ to $\mathbf{x}_t$. To deal with this we will train a neural network to predict $\bar{\boldsymbol{\epsilon}}_t$. After training, the network will be a black box which takes as input

- a value of t and a noisy image $\mathbf{x}_t$

and returns

- a prediction of $\bar{\boldsymbol{\epsilon}}_t$.

We will denote the prediction by the function $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$, where θ represents the parameters in the neural network—these will be learned during the training phase. In each training step, we select an image $\mathbf{x}_0$ from the training set, take a Gaussian $\bar{\boldsymbol{\epsilon}}_t$, and form a sample of $\mathbf{x}_t$ using (3.8). The job of the network is to make the output $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ as close as possible to $\bar{\boldsymbol{\epsilon}}_t$. This leads to the pseudocode forming Algorithm 5.1 in section 5.

Recalling the expression (4.5) for the required transition density, using the neural network prediction $\boldsymbol{\epsilon}_\theta$ in the expression (4.9) for the mean, and adjusting the variance using (4.6), we obtain $\mathbf{x}_{t-1}$ from

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}} \boldsymbol{\epsilon}_\theta \right) + \sigma_q(t)\, \mathbf{z}, \tag{4.10}$$

where $\mathbf{z}$ is a standard Gaussian. This allows us to run the denoising process from $t = T$ to $t = 0$, leading to the sampling procedure described in Algorithm 5.2 of section 5.

Having set up the required expressions, in the next section we outline the resulting training and sampling algorithms.

5. Algorithms. The training process is summarized in Algorithm 5.1, which first appeared in [15]. This pseudocode applies to the case of a basic stochastic gradient method [13]. In steps 1, 2, and 3 we choose an image, a time point, and a standard Gaussian. These quantities are used to create a noisy image for training purposes. In step 5 we apply a least-squares loss function to this single, randomly chosen training image in order to update the network parameters. The simple least-squares formulation can be justified from a likelihood perspective [4, 15, 16, 35].

For the MNIST training experiment in section 2, each image was resized from 28-by-28 pixels to 32-by-32 pixels and rescaled to take pixel values in the range $[-1, 1]$. Rather than basic stochastic gradient, a batch version was used—in step 2 we sampled 128 images and in step 4 we therefore required 128 independent standard Gaussian samples in order to create the 128 training images corresponding to the randomly chosen time t. The sum of the least-squares loss over these samples was then taken in step 5, and the Adam optimizer [22] was used with learning rate of 0.0005, a gradient decay factor of 0.9, and a squared gradient decay factor of 0.9999. The training proceeded over 50 epochs. The network architecture in step 5 combined residual and attention blocks in a U-Net [34] type structure, motivated by the choice in [15]. Overall, that network has 12.9 million parameters across 205 layers.

Algorithm 5.2 from [15] summarizes the sampling process. This is a straightforward implementation of (4.10). Here we define $\sigma_q(1) = 0$, so that only the mean estimate based on (4.9) is used at $t = 1$.

6. PDEs. For many applied mathematicians, diffusion is synonymous with certain parabolic PDEs. Here we aim to draw a PDE connection with the process described in section 3. From a PDE perspective, the forward problem combines a smoothing

Algorithm 5.1. Training with the forward process [15].

1: **repeat**
2: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$ ▷ choose an image from training set
3: $\quad t \sim \text{Uniform}(\{1, 2, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \text{N}(\mathbf{0}, \mathbf{I})$ ▷ standard Gaussian sample
5: $\quad$ Take gradient step w.r.t. θ on $\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\overline{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\overline{\alpha}_t}\, \boldsymbol{\epsilon}, t)\|_2^2$
6: **until** converged

Algorithm 5.2. Sampling with the backward process [15].

$\mathbf{x}_T \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$ ▷ standard Gaussian sample
for $t = T, T-1, \ldots, 1$ **do**
  $\mathbf{z} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$ ▷ standard Gaussian sample
  $\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta \right) + \sigma_q(t)\, \mathbf{z}$
end for
return $\mathbf{x}_0$

dispersive process, a rescaling process, and a stochastic perturbative (noise) process. This results in a (semigroup) flow over a suitable function space. The notion of continuously renormalizing a diffusion process takes us outside the realm of standard textbook analysis and opens up issues of independent interest. Depending on our choice of basic PDE, there are several ways to ensure that the norm of some derivative of the solution remains unchanged over time. Here we illustrate this general idea by continuously rescaling to preserve the norm of the gradient of the solution, that is, the total variation over the domain.

We consider a real-valued field $u(x,t)$, where $x \in \Omega$, a bounded domain in $\mathbb{R}^d$ with a piecewise smooth boundary, $\partial\Omega$, and time $t \geq 0$, subject to diffusion and rescaling,

$$u_t = \Delta u + r(t)u, \quad x \in \Omega, \quad \nabla u \cdot \mathbf{n} = 0, \quad x \in \partial\Omega, \tag{6.1}$$

with a given initial condition $u(x,0) = u_0(x)$. Here $r(t) > 0$ is a *shadow* time-dependent variable (akin to a Lagrange multiplier), which continuously rescales u so that the L^2 norm of the gradient of u is preserved. More explicitly, $r(t)$ must be such that a functional $J[u]$ is conserved, where

$$J[u] := \int_\Omega \nabla u \cdot \nabla u \, dx \equiv R > 0 \quad \text{(constant)}, \quad t \geq 0. \tag{6.2}$$

For this particular choice of $J[u]$ we are conserving the L^2 norm of the gradient of the solution, yet everything in what follows may be generalized to allow J to depend smoothly on $(x, u, \nabla u)$.

Taking the gradient in (6.1), then forming the scalar product with ∇u and integrating over Ω, we obtain

$$\frac{1}{2} \frac{d}{dt} \int_\Omega \nabla u \cdot \nabla u \, dx = \int_\Omega \nabla u \cdot \nabla(\Delta u) \, dx + r(t) \int_\Omega \nabla u \cdot \nabla u \, dx.$$

We have the identity $\nabla \cdot (\nabla u \Delta u) = (\Delta u)^2 + \nabla u \cdot \nabla(\Delta u)$. So, using the divergence theorem [27], in order to ensure that $J[u]$ is conserved we must set

$$R\, r(t) = \int_\Omega (\Delta u)^2 \, dx \geq 0.$$

Hence, we may write (6.1) and (6.2) as the nonlinear integro-differential equation

$$u_t = \Delta u + \frac{u}{R} \int_\Omega (\Delta u)^2 \, dx, \quad x \in \Omega, \quad \nabla u \cdot \mathbf{n} = 0, \quad x \in \partial\Omega. \tag{6.3}$$

Now, for any $R > 0$, the constrained equation represented by (6.1) and (6.2) has infinitely many possible steady states, each of which is of the form $u = \mu_k \phi_k(x)$ and

$r = \lambda_k$, where $(\phi_k(x), \lambda_k)$ is the kth $(k = 0, 1, 2, \ldots)$ eigenfunction-eigenvalue pair for the negative Laplacian on Ω with no-flux boundary conditions, in ascending order of the nonnegative eigenvalues. However, μ_k here must satisfy $\mu_k^2 = R/\int_\Omega ||\nabla\phi_k||^2\, dx$, and hence $k \geq 1$, since the simplest eigenfunction satisfies $||\nabla\phi_0|| \equiv 0$. In fact the ϕ_0 component of u is of no interest here (it is equal to $\int_0^t r(s)\, ds \int_\Omega u_0(x)\phi_0(x), dx$) and may be set to zero.

Linear stability analysis shows that each steady state is stable with respect to perturbations in all higher eigenmodes, yet is unstable with respect to any perturbations in lower eigenmodes. Thus, over a long time, the solution profile must decay to the first eigenmode, $\mu_1\phi_1(x)$.

Now we add a stochastic noise component to the equation:

$$u_t = \Delta u + r(t)u + \eta\mathcal{W}(x,t), \quad x \in \Omega, \quad \nabla u \cdot \mathbf{n} = 0, \quad x \in \partial\Omega,\ t > 0, \tag{6.4}$$

$$u(x,0) = u_0(x). \tag{6.5}$$

Again $r(t)$ must be such that

$$J[u] := \int_\Omega ||\nabla u(x,t)||^2\, dx \equiv R \quad \text{(constant)}, \quad t \geq 0, \tag{6.6}$$

where $R = J[u_0]$. When discretized, the simplest noise term $\mathcal{W}$ takes i.i.d. random values uniformly distributed in [-1,1], while the amplitude, $\eta \geq 0$, may be varied.

Figure 7 shows some numerical solutions of (6.4)–(6.6) obtained using finite differences on a uniform grid ($\delta x = 0.01$ and $\delta t = 0.000015$), with i.i.d. random forcing, $\mathcal{W}$, at all grid points. The instability in the ϕ_0 component of the solution has been constrained to zero. Notice that as η increases, more of the total variation, $J[u]$, is made up from the noise response, and consequently less amplitude is available to the otherwise stable first eigenmode.

Suppose that for a set of K distinct initial values we calculate the solution at some fixed time $T > 0$. The set of pairs $H = \{(u(0,x), u(x,T))\}$ can be considered as containing elements in $X \times X$ for an appropriate function space, X, such as the Sobolev space $W^{1,2}(\Omega)$ (the space of square integrable functions on Ω having weak first derivatives that are also square integrable on Ω).

In general, we have a forward map: $\phi : X \to 2^X$ such that $\phi(u(x,0)) = u(x,T)$, which is one-to-many owing to the stochastic forcing. H merely contains some observations.

We wish to represent the *generative* backward map, $\psi : X \to 2^X$, such that

$$u(x,0) \in \psi(v) \text{ for all } v \in \phi(x,0).$$

This too is one-to-many owing to the nature of the forward problem.

In essence, the generative neural network constructs and learns this backward map from the (observed/calculated) elements of H. Alternative methods might also be successful.

For the numerical solution of the forward problem, as in Figure 7, we may solve on a fixed grid with, say, N grid points and use finite differences and time-stepping for the derivatives and quadrature required by (6.4)–(6.6). Hence, the elements of X are approximated by elements of $\mathbb{R}^N$. Alternatively, if solutions represent pixelated images, then there will be a natural N-point grid. This makes matters somewhat easier in practice.

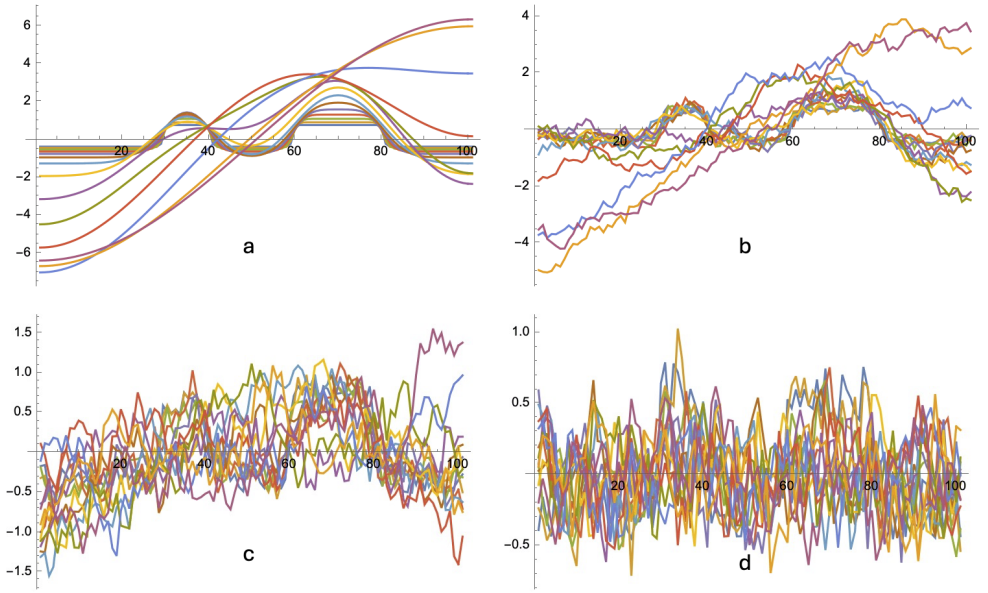

Fig. 7 *The case where $d = 1$, $\Omega = [0, 1]$, and $J[u] \equiv R = 2$, with noise amplitude η increasing. In each graph the numerical solution profiles of u in (6.4)–(6.6) obtained on a uniform grid with $\delta x = 0.01$ and $\delta t = 0.000015 = 0.15\delta x^2$ are shown for some successive time steps (2^k for $k = 1, 2 \ldots, 14$), with different colors, where the (instability in the) ϕ_0 component has been constrained to zero. We have increased the stochastic component:* (a) $\eta = 0$, (b) $\eta = 10{,}000$, (c) $\eta = 12{,}500$, (d) $\eta = 15{,}000$.

In this example, the parabolicity of the forward PDE dynamic evolution, (6.4) and (6.6), means that, formally, the backward PDE is ill-posed, there being no global solution guaranteed (with instabilities, discontinuities, and point masses possibly occurring as time moves backwards). But we do not attempt that: the generative algorithm (or any proposed alternative) for ψ implicitly assumes well-definedness (hence sufficient smoothness).

This framework and others in physics (see [43, 44]) suggest a number of ways in which GAI might exploit, or be explained by, PDE theory.

7. Furthermore. In this final section we touch upon some issues that may have occurred to the reader and we provide references where further information may be found.

How do we judge the performance of GAI? A generative model must balance the contradictory aims of producing outputs that are plausible (like the training data) and novel (not like the training data). Any attempt to quantify performance must involve somewhat arbitrary choices that allow this circle to be squared. A popular quantitative measure, which focuses on the plausibility aspect, is Fréchet inception distance [12]. This measure approximates and compares the probability distributions of the real and synthetic image spaces, under a Gaussian assumption. Some studies also make use of subjective human opinions, which raises new issues, including reproducibility and representativeness.

What are useful applications of diffusion models? Given that the internet already stores a bewildering array of real images, it is reasonable to ask whether the world needs synthetic examples, however realistic. Nevertheless, in some domains rep-

resentative artificial data is valuable. In medical imaging, for example, synthetically generated data may help address scarcity, class imbalance, and privacy concerns within an educational setting [21]. Perhaps the biggest attraction of diffusion models lies in their use within larger systems. A diffusion model for image generation may be viewed as a representation of the hidden, or latent, distribution of real-world images. By conditioning or guiding the image generation according to user-specified requirements, it is then possible to tailor the output to meet certain goals [2, 10, 17, 47]. For example, diffusion forms part of several systems with text-to-image capabilities, including Open AI's DALL-E 2 [32], stability.ai's Dreamstudio, which builds on [33], and Google's Imagen [35]. Inpainting and overwriting unwanted pixels is also possible [30, 46].

Stable diffusion may also be exploited within ChatGPT-style large language models; an example is stability.ai's StableLM-3B-4E1T [41].

How computationally expensive is it to train and employ a diffusion model? For the simple low-resolution examples in section 2, using a pretrained network to produce new images is feasible on a standard desktop machine. However, high-resolution image generation with a state-of-the-art pretrained diffusion model is a "resource intensive and slow task that prohibits interactive experience and results in huge computational costs on expensive GPUs" [1]. The size of many diffusion based models also raises storage issues: "generating high-resolution images with diffusion models is often infeasible on consumer-grade GPUs due to the excessive memory requirements" [30].

Training is greater challenge. For the examples in section 2 we trained the network for 500 epochs in under 35 minutes on a single NVIDIA GeForce RTX 3090 GPU. It is reported in [42] that training the model in [7] consumes 150–1000 days of NVIDIA V100 GPU time. StableLM-3B-4E1T [41] is a 3 billion parameter language model trained on 1 trillion tokens of diverse English and code data sets; a 7 billion parameter version was later released. Developing smaller-scale versions of such models, or applying the models to compressed latent spaces, is therefore an active area of research [33, 45].

In terms of power usage when a trained model is deployed, Luccioni, Jernite, and Strubell [25] estimated that "the most carbon-intensive image generation model (stable-diffusion-xl-base-1.0) generates 1,594 g of CO2 for 1,000 inferences, which is roughly the equivalent to 4.1 miles driven by an average gasoline-powered passenger vehicle."

Is it a coincidence that (3.4) and (4.10) look similar to a numerical discretization of a stochastic differential equation? It is natural to compare (3.4) and (4.10) with the Euler–Maruyama method [14], and indeed there are variations of the forward diffusion model that have a direct correspondence with stochastic differential equations [15, 26, 30, 38]. The reverse process may also be associated with backward stochastic differential equations [42].

What about the dark side: Ethics, privacy, bias, and related concerns? Carlini et al. [5] showed that diffusion models have a tendency to memorize and reproduce training images. For tests on Stable AI [33] and Imagen [35], they were able to "extract over a hundred near-identical replicas of training images that range from personally identifiable photos to trademarked logos." Somepalli et al. [37] also found examples where a diffusion model "blatantly copies" from training data. The resulting harms to professional artists are considered in [20]; these include "reputational damage, economic loss, plagiarism and copyright infringement." When we move into the realm of text-to-image algorithms there are many further issues to consider, including fairness, toxicity, and trust [11].

The figures in section 2 indicate that the output from a simple diffusion model is difficult to predict and hence to interpret. In particular, very different results can be generated from the same input. Explainable AI is a serious challenge in this setting.

On a more general note, any machine learning algorithm is likely to reflect the biases, omissions, and errors in the training set [29]. See [18] for a proposed framework for data transparency.

We also mention that discussions around ethics in this field often assume that AI is, or will become, all-powerful, thereby overlooking empirical observations that these systems may fail to operate as intended—the so-called *fallacy of AI functionality* [31]. So, as well as the important question of what tasks *should* AI be used for, we must also ask what tasks *can* AI reliably perform. This latter issue is ripe for mathematical and statistical contributions [39].

Using GAI to create content (text, images, music, videos, and so on) that is difficult or impossible to discriminate from human generated content may allow fakery and conspiracy theories to undermine societal safety and benefits. This begets novel risks that are already upon us, identified in part by the inaugural AI Safety Summit which met at Bletchley Park in November 2023.[1] Arguably, some of the decadal data science focus on ethics and privacy should have been redirected toward the societal risk of fake truths and the widespread inability to discriminate among content, and the introduction of bias. These risks now require in-depth consideration, as we seek to uncover and tackle the full range of possibilities. An understanding of the mathematical foundations of GAI methods will be key to ensuring transparency.

Data Statement. For the computational experiments in sections 3 and 4, we followed very closely the code provided by the Mathworks tutorial *Generate Images Using Diffusion.*[2] Illustrative MATLAB code is available at https://www.maths.ed.ac.uk/∼dhigham/algfiles.html.

REFERENCES

[1] S. Agarwal, S. Mitra, S. Chakraborty, S. Karanam, K. Mukherjee, and S. Saini, *Approximate caching for efficiently serving text-to-image diffusion models*, in Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation, Santa Clara, CA, USENIX Association, 2024, pp. 1173–1189. (Cited on p. 620)

[2] A. Bansal, H.-M. Chu, A. Schwarzschild, S. Sengupta, M. Goldblum, J. Geiping, and T. Goldstein, *Universal guidance for diffusion models*, in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2023, pp. 843–852. (Cited on p. 620)

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*, Inform. Sci. Statist., Springer, Berlin, 2007. (Cited on pp. 612, 614)

[4] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, *A survey of AI-generated content (AIGC)*, ACM Comput. Surv., 5 (2025), pp. 125:1–125:38. (Cited on pp. 608, 616)

[5] N. Carlini, J. Hayes, M. Nasr, M. Jagielski, V. Sehwag, F. Tramèr, B. Balle, D. Ippolito, and E. Wallace, *Extracting training data from diffusion models*, in Proceedings of the 32nd USENIX Conference on Security Symposium, SEC '23, USENIX Association, 2023, pp. 5253–5270. (Cited on p. 620)

[6] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, *Diffusion models in vision: A survey*, IEEE Trans. Pattern Anal. Mach. Intell., 45 (2023), pp. 10850–10869, https://doi.org/10.1109/TPAMI.2023.3261988. (Cited on p. 608)

[7] P. Dhariwal and A. Nichol, *Diffusion models beat GANs on image synthesis*, in Advances in Neural Information Processing Systems, Vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin,

[1] https://www.gov.uk/government/publications/ai-safety-summit-2023-chairs-statement-2-november/chairs-summary-of-the-ai-safety-summit-2023-bletchley-park

[2] https://uk.mathworks.com/help/deeplearning/ug/generate-images-using-diffusion.html

P. Liang, and J. W. Vaughan, eds., Curran Associates, 2021, pp. 8780–8794. (Cited on pp. 607, 620)

[8] S. Feuerriegel, J. Hartmann, C. Janiesch, and P. Zschech, *Generative AI*, Business Inform. Systems Engrg., 66 (2023), pp. 111–126. (Cited on p. 608)

[9] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber, and X. Alameda-Pineda, *Dynamical variational autoencoders: A comprehensive review*, Found. Trends Mach. Learn., 15 (2021), pp. 1–175, https://doi.org/10.1561/2200000089. (Cited on p. 607)

[10] R. Gozalo-Brizuela and E. C. Garrido-Merchán, *A survey of generative AI applications*, J. Comput. Sci., 20 (2024), pp. 801–818. (Cited on p. 620)

[11] S. Hao, P. Kumar, S. Laszlo, S. Poddar, B. Radharapu, and R. Shelby, *Safety and fairness for content moderation in generative models*, in CVPR Workshop on Ethical Considerations in Creative applications of Computer Vision, 2023. (Cited on p. 620)

[12] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, *GANs trained by a two time-scale update rule converge to a local Nash equilibrium*, in Advances in Neural Information Processing Systems, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, eds., Long Beach, CA, 2017, pp. 6626–6637. (Cited on p. 619)

[13] C. F. Higham and D. J. Higham, *Deep learning: An introduction for applied mathematicians*, SIAM Rev., 61 (2019), pp. 860–891, https://doi.org/10.1137/18M1165748. (Cited on p. 616)

[14] D. J. Higham and P. E. Kloeden, *An Introduction to the Numerical Simulation of Stochastic Differential Equations*, SIAM, 2021, https://doi.org/10.1137/1.9781611976434. (Cited on p. 620)

[15] J. Ho, A. Jain, and P. Abbeel, *Denoising diffusion probabilistic models*, in Proceedings of the 34th International Conference on Neural Information Processing Systems, Curran Associates, Red Hook, NY, 2020, pp. 6840–6851. (Cited on pp. 607, 608, 613, 616, 617, 620)

[16] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans, *Cascaded diffusion models for high fidelity image generation*, J. Mach. Learn. Res., 23 (2022), pp. 1–33. (Cited on p. 616)

[17] J. Ho and T. Salimans, *Classifier-free diffusion guidance*, in NeurIPS. 2021 Workshop on Deep Generative Models and Downstream Applications, 2021. (Cited on p. 620)

[18] B. Hutchinson, A. Smart, A. Hanna, E. Denton, C. Greer, O. Kjartansson, P. Barnes, and M. Mitchell, *Towards accountability for machine learning datasets: Practices from software engineering and infrastructure*, in Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, FAccT '21, ACM, New York, 2021, pp. 560–575. (Cited on p. 621)

[19] G. Iglesias, E. Talavera, and A. Díaz-Álvarez, *A survey on GANs for computer vision: Recent research, analysis and taxonomy*, Comput. Sci. Rev., 48 (2023), 100553, https://doi.org/10.1016/j.cosrev.2023.100553. (Cited on p. 607)

[20] H. H. Jiang, L. Brown, J. Cheng, M. Khan, A. Gupta, D. Workman, A. Hanna, J. Flowers, and T. Gebru, *AI art and its impact on artists*, in Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society, AIES '23, ACM, New York, 2023, pp. 363–374. (Cited on p. 620)

[21] A. Kazerouni, E. K. Aghdam, M. Heidari, R. Azad, M. Fayyaz, I. Hacihaliloglu, and D. Merhof, *Diffusion models in medical imaging: A comprehensive survey*, Medical Image Anal., 88 (2023), 102846, https://doi.org/10.1016/j.media.2023.102846. (Cited on p. 620)

[22] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, in International Conference on Learning Representations (ICLR), San Diego, CA, 2015. (Cited on p. 616)

[23] D. P. Kingma and M. Welling, *Auto-encoding variational Bayes*, in 2nd International Conference on Learning Representations (Banff, AB, Canada, 2014), Conference Track Proceedings, 2014. (Cited on p. 607)

[24] Y. LeCun, C. Cortes, and C. J. C. Burges, *The MNIST Database of Handwritten Digits*, http://yann.lecun.com/exdb/mnist/. (Cited on p. 609)

[25] A. S. Luccioni, Y. Jernite, and E. Strubell, *Power hungry processing: Watts driving the cost of AI deployment?*, in FAccT '24: Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency, Association for Computing Machinery, Rio de Janeiro, Brazil, 2024, pp. 85–99. (Cited on p. 620)

[26] C. Luo, *Understanding Diffusion Models: A Unified Perspective*, arXiv:2208.11970, 2022. (Cited on pp. 608, 620)

[27] P. C. Matthews, *Vector Calculus*, Springer, Berlin, 1998. (Cited on p. 617)

[28] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*, MIT Press, Boston, 2022. (Cited on pp. 612, 614)

[29] A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna, *Data and its (dis)contents: A survey of dataset development and use in machine learning research*, Patterns, 2 (2021), 100336, https://doi.org/10.1016/j.patter.2021.100336. (Cited on p. 621)

[30] R. Po, W. Yifan, V. Golyanik, K. Aberman, J. T. Barron, A. H. Bermano, E. R. Chan, T. Dekel, A. Holynski, A. Kanazawa, C. K. Liu, L. Liu, B. Mildenhall, M. Niessner, B. Ommer, C. Theobalt, P. Wonka, and G. Wetzstein, *State of the art on diffusion models for visual computing*, Comput. Graph. Forum., 43 (2024). (Cited on pp. 608, 620)

[31] I. D. Raji, I. E. Kumar, A. Horowitz, and A. Selbst, *The fallacy of AI functionality*, in Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, FAccT '22, ACM, New York, 2022, pp. 959–972. (Cited on p. 621)

[32] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, *Hierarchical Text-Conditional Image Generation with CLIP Latents*, arXiv:2204.06125, 2022. (Cited on pp. 608, 620)

[33] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 10684–10695. (Cited on p. 620)

[34] O. Ronneberger, P. Fischer, and T. Brox, *U-Net: Convolutional networks for biomedical image segmentation*, in Medical Image Computing and Computer-Assisted Intervention – MICCAI. 2015, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds., Springer International, Cham, 2015, pp. 234–241. (Cited on p. 616)

[35] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, *Photorealistic text-to-image diffusion models with deep language understanding*, in Advances in Neural Information Processing Systems, Vol. 35, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds., Curran Associates, 2022, pp. 36479–36494. (Cited on pp. 616, 620)

[36] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics*, in Proceedings of the 32nd International Conference on Machine Learning, Proc. Mach. Learn. Res. 37, F. Bach and D. Blei, eds., PMLR, Lille, France, 2015, pp. 2256–2265. (Cited on p. 608)

[37] G. Somepalli, V. Singla, M. Goldblum, J. Geiping, and T. Goldstein, *Diffusion art or digital forgery? Investigating data replication in diffusion models*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 6048–6058. (Cited on p. 620)

[38] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, *Score-based generative modeling through stochastic differential equations*, in International Conference on Learning Representations, 2021. (Cited on p. 620)

[39] O. J. Sutton, Q. Zhou, I. Y. Tyukin, A. N. Gorban, A. Bastounis, and D. J. Higham, *How adversarial attacks can disrupt seemingly stable accurate classifiers*, Neural Netw., 180 (2024). (Cited on p. 621)

[40] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, *Efficient transformers: A survey*, ACM Comput. Surv., 55 (2022), 109. (Cited on p. 607)

[41] J. Tow, M. Bellagente, D. Mahan, and C. R. Ruiz, *StableLM-3B-4E1T*, Technical report, 2023. (Cited on p. 620)

[42] Z. Wang, *Score-Based Generative Modeling through Backward Stochastic Differential Equations: Inversion and Generation*, arXiv:2311.16863, 2023. (Cited on p. 620)

[43] Y. Xu, Z. Liu, M. Tegmark, T. Jaakkola, S. Koyejo, S. Mohamed, and A. Agarwal, *Poisson flow generative models*, in Advances in Neural Information Processing Systems, Vol. 35, D. Belgrave, K. Cho, and A. Oh, eds., Curran Associates, 2022, pp. 16782–16795, https://proceedings.neurips.cc/paper_files/paper/2022/file/6ad68a54eaa8f9bf6ac698b02ec05048-Paper-Conference.pdf. (Cited on p. 619)

[44] Y. Xu, Z. Liu, Y. Tian, S. Tong, M. Tegmark, and T. Jaakkola, *PFGM++: Unlocking the potential of physics-inspired generative models*, in Proceedings of the 40th International Conference on Machine Learning, Proc. Mach. Learn. Res. 202, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds., 2023, pp. 38566–38591. (Cited on p. 619)

[45] X. Yang, D. Zhou, J. Feng, and X. Wang, *Diffusion probabilistic model made slim*, in 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 22552–22562, https://doi.org/10.1109/CVPR52729.2023.02160. (Cited on p. 620)

[46] A. B. Yildirim, V. Baday, E. Erdem, A. Erdem, and A. Dundar, *Inst-Inpaint: Instructing to Remove Objects with Diffusion Models*, arXiv:2304.03246, 2023. (Cited on p. 620)

[47] L. Zhang, A. Rao, and M. Agrawala, *Adding conditional control to text-to-image diffusion models*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 3836–3847. (Cited on p. 620)