# CCIMI Lent Term project

Literature review and further thoughts:
Score-Based Generative Modelling through
Stochastic Differential Equations (Song et al. 2020)


| | |
|---|---|
| PhD Students: | Parley Ruogu Yang & Georgios Batzolis |
| Postdoc: | Dr Christian Etmann |
| Big boss: | Prof Carola-Bibiane Schönlieb |


| | |
|---|---|
| This version: | 26 Apr 2021 |
| Presentation date (to CMI) : | 30 Mar 2021 |
| Presentation material (to CMI) : | click here (Parley's website) |
| Presentation date (to GAM) : | 14 Apr 2021 |

# 1 Introduction and key literature review (PRY)

## 1.1 Motivation

We start with a general introduction to the generative modellings.

Consider a set-up where we have $x_1, ..., x_n$ drawn from $p_0(\mathbb{R}^d)$ with $d \gg n$ and $p_0$ unknown. The ultimate aim that we embrace is to know more about $p_0$, and eventually being able to draw samples from it.

## 1.2 Reverse-time SDE by Anderson (1982)

Consider a Forward Ito SDE

$$dx = f(x,t)dt + g(t)dw \tag{1}$$

where $x : [0,T] \to \mathbb{R}^d$, a stochastic process; $w : [0,T] \to \mathbb{R}$, a standard Brownian motion, $f : \mathbb{R}^d \times [0,T] \to \mathbb{R}$, a vector-drift; and $g : [0,T] \to \mathbb{R}$, a scalar diffusion.

Now we construct a reverse-time Brownian motion $\overline{w} : [0,T] \to \mathbb{R}$ — a stochastic process satisfying the definition of Brownian motion in a reverse-manner — for a decreasing family $\{A(t)\}_{t\in[0,T]}$ of sub-$\sigma$-algebras of $A$, $\forall s \geq 0$,

$$\mathbb{E}[\overline{w}(t)|A(t+s)] = \overline{w}(t+s)$$

$$\mathbb{E}[(\overline{w}(t) - \overline{w}(t+s))^2|A(t+s)] = s$$

An important consequence is, for a reverse time model for $x$, we have

$$dx = (f(x,t) - g(t)^2 \nabla_x \log(p_t(x)))dt + g(t)d\overline{w} \tag{2}$$

where we note $p_t(x)$ as the probability distribution of $x_t$

## 1.3 A general algorithm given by Song et al. (2020)

A pair of algorithms for forward from 0 to $T$ and then backward $T$ to 0 can be written as follows:

**Algorithm 1:** Forward noise-making

**Input:** Data $\{x_i(0)\}_{i\in[n]}$ , specification of $f, g, T$

**Output:** Noises $\{x_i(T)\}_{i\in[n]}$ and eventual distribution $\hat{p_T}$

1. For $i \in [n]$, repeat:
   (a) Noise-adding as per Equation 1
2. Collect eventual distribution $\hat{p_T}$ from $\{x_i(T)\}_{i\in[n]}$

As a consequence, we aim for $\hat{p_T} \approx q$ where $q$ is an easy-to-draw distribution. A typical example, which is also theoretically valid, is simply a standard Gaussian distribution $N(0, I)$.

**Algorithm 2:** Backward image-generation

**Input:** Number of desired samples $m$, information from algorithm 1, $q$ and $\nabla_x \log(p_t(x))$

**Output:** Image samples $\{y_i(0)\}_{i\in[m]}$ and distribution $\hat{p_0}$

1. For $i \in [m]$, repeat:
   (a) Draw $y_i(T) \sim q$.
   (b) Time-reverse as per Equation 2 to obtain $y_i(0)$
2. Collect eventual distribution $\hat{p_0}$ from $\{y_i(0)\}_{i\in[m]}$

## 1.4 Some further theoretical appreciation

First, as an appreciation from the theory, we understand from Anderson (1982) that if we have any regular (say compact support and differentiable under Euclidean topology) distribution $p_0$ and we apply Equation 1 then Equation 2, we get the exactly the same result as if we were drawing from distribution $p_0$. More precisely, say if we draw directly from $p_0$ and obtain an empirical distribution $p^E_{0,n}$, then LLN applies, i.e. $\mu(p^E_{0,n}) \xrightarrow[n\to\infty]{a.s.} \mu(p_0)$ ; now because of Anderson (1982), if we were to proceed with algorithm 1 and algorithm 2 and obtain $\hat{p}_{0,n,m}$, then we must have $\mu(\hat{p}_{0,n,m}) \xrightarrow[n,m\to\infty]{a.s.} \mu(p_0)$. The same argument applies for CLT.

It is important to remark here, however, that in practice we are mostly unlikely to have access to $\nabla_x \log(p_t(x))$, which creates another layer of complication. Song et al. (2020) and other papers suggest to use Neural Network (NN) architecture, and aim for a good approximation for $\nabla_x \log(p_t(x))$ by a NN-approximated score function $s_\theta(x, t)$. But the lack of theory on the theoretical development of NN makes us hard to obtain theoretical guarantee on the statistical side, e.g. the first paragraph.

## 1.5 Further materials relevant to ODE and SDE

Nonetheless, we have some exciting results from SDE theory on the transition kernel $p(x(t)|x(s))$. This gives fruitful information about approximating $\nabla_x \log(p_t(x))$, and is apparently crucial for NN approximation.

Below are extracts relevant to our problem from Särkkä and Solin (2019).

We first consider a linear ODE of the form

$$\partial_t x(t) = F(t)x(t) + L(t)w(t) \tag{3}$$

with $x(t_0)$ given, then consider a transition matrix $\Psi(t,s)$ with several technical assumptions (Särkkä and Solin 2019, p. 11), we may obtain a general solution

$$x(t) = \Psi(t,t_0)x(t_0) + \int_{t_0}^{t} \Psi(t,\tau)L(\tau)w(\tau)d\tau$$

Now, if we consider the following linear SDE

$$dx = F(t)x + u(t)dt + L(t)dw \tag{4}$$

Because of the heterogeneity term $L(t)dw$, we can easily conclude $p(x(t)|x(s))$ to be Gaussian, and that

$$m(t|s) = \Psi(t,s)x(s) + \int_{s}^{t} \Psi(t,\tau)u(\tau)d\tau \tag{5}$$

$$P(t|s) = \int_{s}^{t} \Psi(t,\tau)L(\tau)QL^T(\tau)\Psi^T(t,\tau)d\tau \tag{6}$$

where $Q$ is the diffusion matrix for Brownian motion $w$.

## 1.6 Potential applications to time series and financial data

The following literature can be relevant: Kong et al. (2020) and Li et al. (2020).

# 2 Computing implementation (GB)

As this is an entirely new approach to generative modelling, we decided to re-implement the paper so that we can understand better not only the key elements of the framework, but also the important details which can be overlooked if one only studies the paper.

## 2.1 Implementation of the basic form of the framework

The result from Anderson states that if we can access the score of the time dependent distribution of $x(t)$, i.e. $\nabla_{x(t)} P_t(x(t))$, we can derive the reverse diffusion process shown in 2, which runs backwards in time and transforms $P_T$ to $P_0$ (the target distribution). If we can sample from $P_T$, then we can perturb those samples using the reverse SDE and get samples from $P_0$. Therefore, there are two important questions that need to be addressed:

- **How do we approximate the score of the time dependent distribution?** The authors of the paper claim that the score of the distribution can be approximated by a time-dependent score-based model $s_\theta(x(t), t)$ which is usually a neural network whose parameters $\theta$ are optimised by minimising the following score-matching objective:

$$\theta^* = \arg\min_\theta \mathbb{E}_t \left[ \lambda(t) \mathbb{E}_{x(0)} \mathbb{E}_{x(t)|x(0)} \left[ \left\| s_\theta(x(t), t) - \nabla_{x(t)} \log p_{0t}(x(t)|x(0)) \right\|_2^2 \right] \right] \tag{7}$$

  where $\lambda : [0, T] \to \mathbb{R}_{>0}$ is a positive weighting function, $t$ is uniformly sampled over $[0, T]$, $x(0) \sim p_0(x)$ and $x(t) \sim p_{0t}(x(t)|x(0))$. They claim that with sufficient data and model capacity, score matching ensures that $s_{\theta^*}(x(t), t)$ equals $\nabla_{x(t)} P_t(x(t))$ for almost all $x(t)$ and $t$.

- **How do we sample from $P_T$?**. We need to choose the time-dependent drift and diffusion coefficients of the forward SDE appropriately, so that the final distribution $P_T$ is very close to a known distribution such as $\mathcal{N}(0, 1)$. The authors explore three different choices:

  1. Variance exploding SDE: $dx = \sqrt{\frac{d[\sigma(t)^2]}{dt}} \mathrm{dw}$, where $\sigma(t) = \sigma_{min}(\frac{\sigma_{max}}{\sigma_{min}})^t$, $t \in (0, 1]$. The perturbation kernel for this SDE is:

$$p_{0t}(x(t)|x(0)) = \mathcal{N}\left( x(t); x(0), \sigma_{min}^2 \left( \frac{\sigma_{max}}{\sigma_{min}} \right)^{2t} \mathbf{I} \right)$$

     If the input $x(0)$ is normalised to the range $[0, 1]$ and $\sigma_{max}$ is chosen to be at least equal to the largest euclidean distance of any two points in the

dataset, it turns out that the marginal distribution $p(x(1))$ is very close to $N(0, \sigma_{max}^2 \mathbf{I})$.

2. Variance Preserving SDE: $dx = -\frac{1}{2}\beta(t)xdt + \sqrt{\beta(t)}dw$, where $\beta(t) = \beta_{min} + t(\beta_{max} - \beta_{min})$. In their experiments, $\beta_{min} = 0.1$ and $\beta_{max} = 10$ are used. The perturbation kernel for this SDE is:

$$p_{0t}(x(t)|x(0)) = \mathcal{N}(x(t); \mu(t), cov(t))$$

where

$$\mu(t) = \exp\left( -\frac{1}{4}t^2(\beta_{max} - \beta_{min}) - \frac{1}{2}t\beta_{min}\right)x(0)$$

and

$$cov(t) = \mathbf{I} - \mathbf{I}\exp\left( -\frac{1}{2}t^2(\beta_{max} - \beta_{min}) - t\beta_{min}\right)$$

Therefore, the marginal $p(x(1))$ is very close to $N(0, 1)$ if $x(0)$ is reasonably scaled.

3. Sub-variance preserving SDE.

$$dx = -\frac{1}{2}\beta(t)xdt + \sqrt{\beta(t)\left(1 - \exp\left(2\int_0^t \beta(s)ds\right)\right)}dw$$

where $\beta(t)$ is chosen in exactly the same way as in the VP sde. The difference in the perturbation kernel is in the time dependent covariance which is strictly smaller than the covariance in the VP sde. This is why they named this SDE sub-variance preserving. The perturbation kernel for this SDE is:

$$p_{0t}(x(t)|x(0)) = \mathcal{N}(x(t); \mu(t), cov(t))$$

where

$$\mu(t) = \exp\left( -\frac{1}{4}t^2(\beta_{max} - \beta_{min}) - \frac{1}{2}t\beta_{min}\right)x(0)$$

and

$$cov(t) = \left(1 - \exp\left( -\frac{1}{2}t^2(\beta_{max} - \beta_{min}) - t\beta_{min}\right)\right)^2 \mathbf{I}$$

In this case, $p(x(1))$ is even closer to $N(0, 1)$ than it is in the VP sde if the same values of $\beta_{min}$ and $\beta_{max}$ are used.

The final practical issue that needs to be addressed is the numerical solving of the reverse-time SDE. One may use any general-purpose numerical solver for integrating the reverse time sde (e.g. Euler-Maruyama, stochastic Runge-Kutta) to generate samples. However, one may achieve even better performance by leveraging

the approximation of the score of the time dependent distribution, i.e. $s_\theta(x,t) \approx \nabla_x \log p_t(x)$ by employing score-based MCMC approaches such as Langevin MCMC. To do that they propose that every estimate of the numerical solver at each reverse time step (prediction) is followed by a step of a score-based MCMC algorithm which corrects the marginal distribution of the estimated sample, thus playing the role of the corrector. This hybrid sampling scheme is called predictor-corrector sampler and typically gives better performance than predictor only sampling schemes. However, the corrector step depends on a choice of hyper-parameters which should be tuned for each dataset using a grid search. The proposed predictor corrector samplers for the VE and VP sdes are shown in Algorithm 2 and Algorithm 3 respectively shown below in Figure 1.

**Algorithm 2** PC sampling (VE SDE)

$\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \sigma_{\max}^2 \mathbf{I})$
**for** $i = N-1$ **to** $0$ **do**
  $\mathbf{x}'_i \leftarrow \mathbf{x}_{i+1} + (\sigma_{i+1}^2 - \sigma_i^2)\mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_{i+1}, \sigma_{i+1})$
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\sigma_{i+1}^2 - \sigma_i^2}\mathbf{z}$
  **for** $j = 1$ **to** $M$ **do**
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i, \sigma_i) + \sqrt{2\epsilon_i}\mathbf{z}$
**return** $\mathbf{x}_0$

**Algorithm 3** PC sampling (VP SDE)

$\mathbf{x}_N \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
**for** $i = N-1$ **to** $0$ **do**
  $\mathbf{x}'_i \leftarrow (2 - \sqrt{1-\beta_{i+1}})\mathbf{x}_{i+1} + \beta_{i+1}\mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_{i+1}, i+1)$
  $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{x}_i \leftarrow \mathbf{x}'_i + \sqrt{\beta_{i+1}}\mathbf{z}$ — Predictor
  **for** $j = 1$ **to** $M$ **do** — Corrector
    $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
    $\mathbf{x}_i \leftarrow \mathbf{x}_i + \epsilon_i \mathbf{s}_{\boldsymbol{\theta}*}(\mathbf{x}_i, i) + \sqrt{2\epsilon_i}\mathbf{z}$
**return** $\mathbf{x}_0$

Figure 1: PC sampling algorithms for VE and VP reverse SDEs.

Taking all this information into consideration we re-implemented the framework in pytorch and used the code to create a demo which shows visually how the framework works. In the demo, we show: 1.) how forward SDE gradually modifies $P_0$ to $P_T$, where $P_T$ is very close to a distribution which we know a priori $\hat{P}_T$ and 2.) how the reverse SDE gradually modifies $\hat{P}_T$ to a distribution $\hat{P}_0$ which is very close to $P_0$. We model a two dimensional $P_0$ distribution because it is easy to visualise the evolution of the entire distribution. In our experimental setting we decided to model a distribution which is a mixture of 4 Gaussians and is shown in Figure 2.

In Figure 3, we show the evolution of the distribution $P_0$ to $P_T$ as the forward SDE is solved from 0 to 1 on the first row, while we show the evolution of distribution $\hat{p}_T$ to $\hat{p}_0$ as the reverse SDE is solved from 1 to 0.001 on the second row. A subtle point is that we do not integrate the forward SDE from 0 to 1, but from 0.001 to 1, because the VE sde that we used is not defined at $T = 0$. The same holds for the reverse SDE. In contrast to our expectation, we did not get better results when we used the hybrid predictor corrector sampler. We attribute that to the fact that we did not tune the hyperparameters of the corrector step on our dataset, but we used the hyperparameter values of the paper. The authors tuned the hyperparameters on the CIFAR-10 dataset, which is very different from our synthetic 2D dataset.

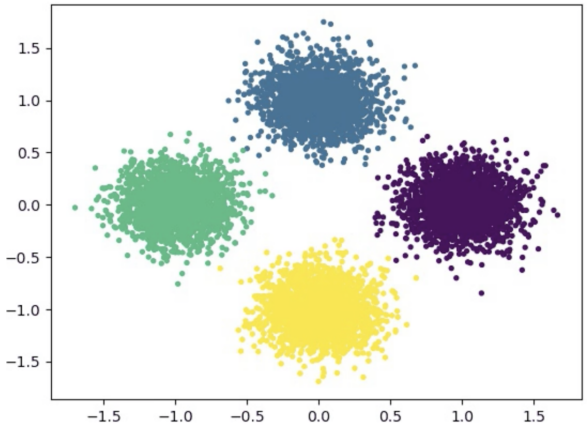


Figure 2: Target distribution $P_0$

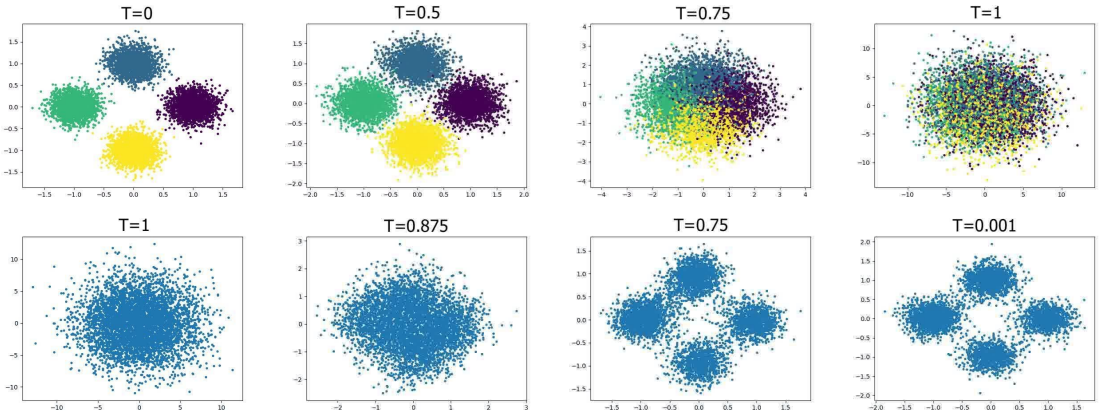



Figure 3: demo. First row: evolution from $P_0$ to $P_T$ by the forward SDE. Second row: evolution from $\hat{P}_T$ to $\hat{P}_0$ by the reverse SDE.

## 2.2 Extension for class conditional sample generation

The framework can be readily extended for class conditional sample generation. Our goal is to sample from $p_0(x(0)|y)$ where $y$ is the class condition. Using the result from Anderson, it turns out that we can sample from $p_0(x(0)|y)$ if we sample from $p_T(x(T)|y)$ and solve the following reverse-time SDE:

$$dx = [f(x,t) - g(t)^2 \nabla_{x(t)} p_t(x(t)|y)]dt + g(t)d\bar{w} \tag{8}$$

Noting that

$$\nabla_{x(t)} p_t(x(t)|y) = \nabla_{x(t)} p_t(x(t)) + \nabla_{x(t)} p_t(y|x(t))$$

equation (8) is transformed to:

$$dx = \big(f(x,t) - g(t)^2 [\nabla_{x(t)} p_t(x(t)) + \nabla_{x(t)} p_t(y|x(t))]\big)dt + g(t)d\bar{w} \tag{9}$$

The difference from the unconditional generation is that apart from approximating the score of the time dependent distribution, $\nabla_{x(t)} p_t(x(t))$, we additionally need to approximate the gradient of the forward process, $\nabla_{x(t)} p_t(y|x(t))$. The authors of the

paper claim that equation (9) can be used to solve a large family of inverse problems if an estimate of the gradient of the forward process can be obtained. In the case of class conditional generation where y represents a class, we can train an auxiliary time-dependent classifier $p_t(y|x(t))$ to obtain an estimate of the time dependent forward process. The training samples $(x(t), y)$ for the time-dependent classifier are obtained by sampling a pair $(x(0), y)$ from the dataset and then getting $x(t)$ by integrating the forward SDE until time $t$. The training loss for the auxiliary classifier is a mixture of cross-entropy losses over uniformly sampled time steps similarly to equation (7).

We trained the auxiliary classifier on the synthetic dataset, assigning all points of each part of the mixture to the same class. Figure 4 shows how the accuracy of the time-dependent classifier decreases from 1 to $0.25 + \epsilon$ as time flows from 0 to 1.

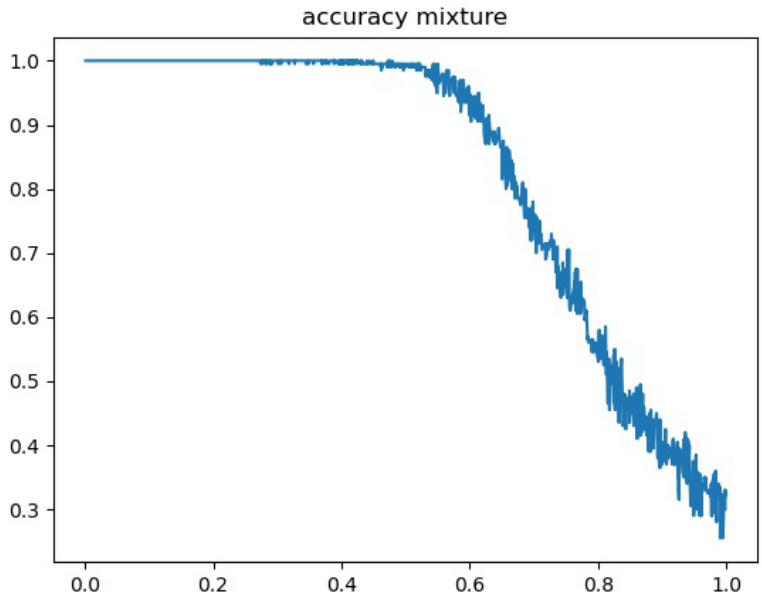


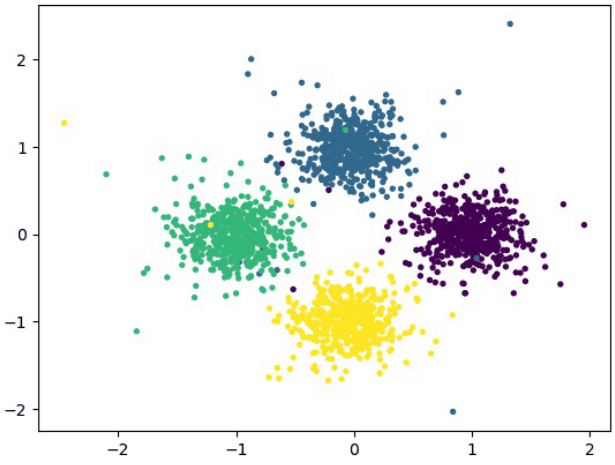

Figure 4: Auxiliary classifier accuracy as time $t$ increases from 0 to 1.

This behaviour is expected given that the forward sde converts the synthetic data distribution to a distribution which is very close to $\mathcal{N}(0, \sigma_{max}^2 \mathbf{I})$ as seen in the first row of figure 3. However, given that $\hat{P}_T$ is close to but not the same as $P_T$ when $T = 1$ and that the perturbation kernel for the VE SDE is

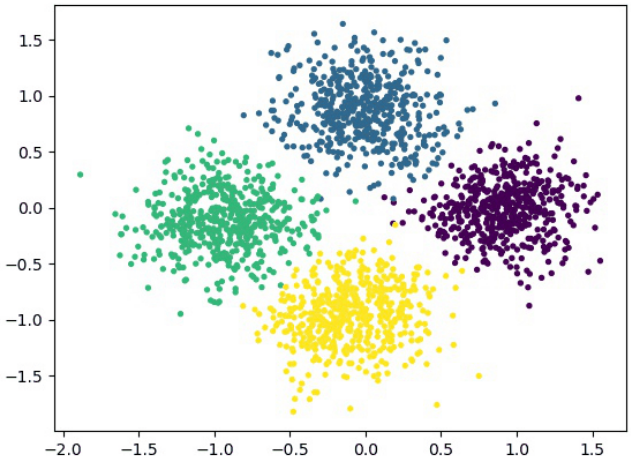
$$p_{0t}(x(t)|x(0)) = \mathcal{N}\left(x(t); x(0), \sigma_{min}^2 \left(\frac{\sigma_{max}}{\sigma_{min}}\right)^{2t} \mathbf{I}\right)$$

, we expect that the classifier will do slightly better than random guessing when $T = 1$ which is indeed verified by the accuracy mixture graph.

Finally, we used the auxiliary classifier and the score-based model to derive the reverse time SDE shown in equation (9) which allowed us to get samples conditioned




(a) predictor only




(b) predictor-corrector

Figure 5: Samples from the class conditional distribution with different samplers

on each class. We solved the reverse-time SDE using both the predictor only and predictor-corrector samplers and we present the results in Figures 5(a) and 5(b) respectively. The predictor corrector sampler clearly outperforms the predictor only sampler. In Figure 5(a), there are easily spotted outliers for each class, while this is not the case in Figure 5(b) which was obtained using the hybrid predictor-corrector sampling scheme.

# References


Anderson, Brian D.O. (1982). "Reverse-time diffusion equation models." In: *Stochastic Processes and their Applications* 12.3, pp. 313–326.

Kong, Zhifeng, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro (2020). "DiffWave: A Versatile Diffusion Model for Audio Synthesis." In: eprint: arXiv:2009.09761.

Li, Xuechen, Ting-Kam Leonard Wong, Ricky T. Q. Chen, and David Duvenaud (2020). "Scalable Gradients for Stochastic Differential Equations." In: eprint: arXiv:2001.01328.

Särkkä, Simo and Arno Solin (2019). *Applied Stochastic Differential Equations*. Cambridge University Press. DOI: 10.1017/9781108186735.

Song, Yang, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole (2020). "Score-Based Generative Modeling through Stochastic Differential Equations." In: eprint: arXiv:2011.13456.