UNDER REVIEW

RESCIENCE C

Replication / NeurIPS 2019 Reproducibility Challenge

[Re] Generative Modeling by Estimating Gradients of the Data Distribution

Antonio Matosevic[1], Eliisabet Hein[1], Francesco Nuzzo[1],

[1] KTH Royal Institute of Technology, Stockholm, Sweden

Edited by
Koustuv Sinha

Reviewed by
(Reviewer 1)
(Reviewer 2)

Received
15 February 2020

Published
–

DOI
–

A replication of https://dblp.org/rec/journals/corr/abs-1907-05600.bib.

Abstract

In this project we attempt to reproduce results from the paper *Generative Modeling by Estimating Gradients of the Data Distribution* by Song and Ermon [1]. The authors propose a novel generative framework based solely on gradients of data density estimated by a neural network. Once the model is trained, sampling can be performed with annealed Langevin dynamics. While we managed to reproduce the experiments qualitatively, we failed to achieve comparable results for Inception and FID scores for CIFAR-10. We further extended the original work in various directions (computing and analysing FID and IS also for CelebA, investigation of the sampling hyperparameters ϵ and T, linear instead of geometric annealing schedule for noise levels, and different network architecture).

1 Introduction

Recent popular deep generative models can roughly be categorised as either adversarial (GANs) or likelihood-based (VAEs, autoregressive, flow-based). While the former often suffer from unstable training, the latter require a particular architecture or a surrogate loss. Song and Ermon [1] introduce a novel method based on estimating the gradients of data log-density with respect to the input data (i.e. *score*).
Subsequently, during sampling, an initial random point is moved to a high-density region by using the estimated gradient. In this report, we investigate the reproducibility of [1]. For this purpose, we briefly introduce the method (Section 2), present the implementation details and comment on the reproducibility (Section 3), provide the results of toy (Section 4) and main (Section 5) experiments, as well as propose and test extensions to the original paper (Section 6).

2 Method

Score estimation One major component of the proposed approach is direct estimation of the score $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ using a neural network $s_{\boldsymbol{\theta}}(\mathbf{x})$, thus circumventing estimation of the data density $p(\mathbf{x})$. To this end, the theoretical objective $\frac{1}{2}\mathbb{E}_p\left[||s_{\boldsymbol{\theta}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log p(\mathbf{x})||_2^2\right]$ can be reformulated in different ways to get two different loss functions, sliced score matching (SSM) [2] and denoising score matching (DSM) [3]. SSM uses projections on random vectors $\mathbf{v} \sim p(\mathbf{v}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and is given as

$$\mathbb{E}_{p_{\mathbf{v}}}\mathbb{E}_{p(\mathbf{x})}\left[\mathbf{v}^{\mathrm{T}}\nabla_{\mathbf{x}} s_{\boldsymbol{\theta}}(\mathbf{x})\mathbf{v} + \frac{1}{2}||s_{\boldsymbol{\theta}}(\mathbf{x}))||_2^2\right]. \tag{1}$$

Correspondence should be addressed to Antonio Matosevic (matose@kth.se)
The authors have declared that no competing interests exists.
Code is available at https://github.com/Xemnas0/NCSN-TF2.0.
Open peer review is available at https://openreview.net/forum?id=B1lcYrBgLH.

While the SSM loss is exact, it is slow to compute, so often in practice DSM loss is used instead. DSM loss estimates the score of density for perturbed data $\tilde{\mathbf{x}}$ and is given as

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p(\mathbf{x})}\left[||s_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}}\log q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})||_2^2\right], \tag{2}$$

where $q_\sigma(\tilde{\mathbf{x}}|\mathbf{x}) = \mathcal{N}(\mathbf{x}, \sigma^2\mathbf{I})$ in [1]. Intuitively, a perturbed $\tilde{\mathbf{x}}$ should move us towards the original $\mathbf{x}$. In fact, it holds that $s^*_{\boldsymbol{\theta}}(\mathbf{x}) = \nabla_{\mathbf{x}}\log q_\sigma(\mathbf{x}) \approx \nabla_{\mathbf{x}}\log p(\mathbf{x})$ for σ small enough so that $q_\sigma(\mathbf{x}) \approx p(\mathbf{x})$.

Sampling Using scores for generative modelling relies on sampling with Langevin dynamics. Namely, given a step size $\epsilon > 0$ and an initial value $\tilde{\mathbf{x}}_0$ we can, using only the score, compute $\tilde{\mathbf{x}}_t = \tilde{\mathbf{x}}_{t-1} + \frac{\epsilon}{2}\nabla_{\mathbf{x}}s^*_{\boldsymbol{\theta}}(\tilde{\mathbf{x}}_{t-1}) + \sqrt{\epsilon}\,\mathbf{z}_t$, where $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, for T steps. It has been shown that under some regularity conditions $\tilde{\mathbf{x}}_T$ is an exact sample from $p(\mathbf{x})$ as $T \to \infty$ and $\epsilon \to 0$ [4].
Unfortunately, direct implementation of the proposed framework usually suffers from two major issues, which the authors attribute to the manifold hypothesis and low-density regions of $p(\mathbf{x})$.

Manifold hypothesis For most natural images, the manifold hypothesis states that the intrinsic dimensionality of $\mathbf{x}$ has a support in $\mathbb{R}^M \subset \mathbb{R}^D$, where $M \ll D$. As a consequence, a score (gradient) taken in $\mathbb{R}^D$ is not going to be defined in $\mathbb{R}^M$. We will demonstrate the effects of this on training the score network by replicating a toy example from [1] (Section 4).

Low-density regions A negative effect of low-density regions concerns both score estimation and sampling. Firstly, the number of samples from low-density regions of $p(\mathbf{x})$ is not sufficient to accurately estimate the loss (i.e. scores) during training. Following [1], we visualise this by comparing analytic and estimated scores of a mixture of two Gaussians with near-zero density regions between the modes (Section 4). Regarding sampling, Langevin dynamics require an infeasible number of iterations in traversing these regions to obtain good mixing. We successfully reproduce an experiment showing this on the same GMM (Section 4).

Proposed solutions Firstly, to remedy the negative implications of the manifold hypothesis, Song and Ermon [1] suggest adding Gaussian noise to data. Since the noise is defined in $\mathbb{R}^D$, this results in gradients being defined everywhere. Secondly, the authors observe that adding large noise also helps to fill in the low-density regions, hence improving the score estimation. Thirdly, while large noise helps during the training, it is not favourable for sampling. To address this issue, they propose obtaining a sample by iteratively generating from distributions of perturbed data $\{q_{\sigma_i}\}_{i=1}^L$ parameterised by a decreasing geometric sequence $\{\sigma_i\}_{i=1}^L$ of noise levels. This way one utilises the benefits of large noise to avoid low-density regions, but also gradually transitions to small noise where the perturbed data distribution is indistinguishable from $p(\mathbf{x})$. In practice, this requires only a small modification to Langevin dynamics described in paragraph 2 by embedding it into iterations over noise levels[1] and multiplying ϵ by a factor of ${\sigma_i}^2/{\sigma_L}^2$. To employ this procedure, a single score network is trained with the total loss $\mathcal{L}(\boldsymbol{\theta}; \{\sigma_l\}_{l=1}^L) = \frac{1}{L}\sum_{l=1}^L \lambda(\sigma_l)\ell(\boldsymbol{\theta}; \sigma_i)$, where $\ell(\cdot)$ is a DSM loss[2], and $\lambda(\cdot)$ a parameter regulating that none of the individual loss terms dominates (σ_i^2 in [1]).

[1] Note that $\mathbf{x}_{\sigma_i} \sim q_{\sigma_i}$ is used to initialise sampling from $q_{\sigma_{i+1}}$. Intuitively, $\mathbf{x}_{\sigma_i}$ comes from a high-density region of q_{σ_i} and since q_{σ_i} and $q_{\sigma_{i+1}}$ are similar, $\mathbf{x}_{\sigma_i}$ is also in a high-density region of $q_{\sigma_{i+1}}$.

[2] Well-motivated since we are now estimating the score of perturbed data.

3 Implementation

All implementation was done in Python using TensorFlow 2.0, and the models were trained using P100 and V100 GPUs on the Google Cloud Platform. The original code by the authors was provided in PyTorch. To our knowledge, this is the first time this method has been ported to a new framework. Our code is available on GitHub at https://www.github.com/Xemnas0/NCSN-TF2.0.

3.1 Data pipeline

Since the datasets used (MNIST, CIFAR-10 and CelebA) are standard, we obtained them through the TensorFlow datasets API. We used the standard training split for each dataset (60,000 samples for MNIST and CIFAR-10, and 162,770 samples for CelebA). We applied preprocessing to the data following [1]. For CelebA, we extracted a 140×140 centre crop from the image and then resized to 32×32. Since the original paper does not specify which interpolation method to use for resizing, we used the default in TensorFlow (bilinear). For all the datasets, we scaled the values from [0, 255] to [0, 1] by dividing each pixel value by 255. Finally, for CIFAR-10 and CelebA, we also flipped each image along the vertical axis with 50% probability each time a batch was loaded. The data was shuffled at the beginning of every epoch with a buffer size of 10,000 and then split into batches. After consulting with the published code, we noted that instead of using the default, the authors create a randomly sampled training split for MNIST and CelebA, and shuffle the whole dataset each epoch, but since our buffer is reasonably large, we do not consider these differences significant.

3.2 Network architecture

An important aspect of the method is choice of the score network. The authors used three architectures (MLP and ResNet for toy experiments and RefineNet for the main experiment). As in [1], all models were trained with Adam optimizer with learning rate 0.001. For all experiments, batch size was 128. For ResNet and RefineNet, we used 64 filters for MNIST and 128 for CIFAR-10 and CelebA.

MLP For the results in Figure 2c, following [1], we use a three-layer MLP with 128 units per layer with softplus activation. While this was not specified by the authors, we do not apply activations to the outputs in any architecture, to keep the scores unnormalized and unbounded.

ResNet For the results in Figure 1, we use a ResNet encoder-decoder network as in [1]. The encoder consists of 5 pre-activation residual blocks with 32, 64, 64, 128, 128 filters in each layer respectively, mirrored in the decoder. Downsampling/upsampling is performed at the end of the 2nd and 4th residual blocks in the encoder and decoder respectively. Activation function is ELU. For the details that were not specified in the paper, we made the following assumptions:

- Convolutions (encoder) or transposed convolutions (decoder) are of size 3×3.
- Resizing is performed with (transposed) convolutions of stride 2.
- Normalisation is done over batches and only within the residual blocks.
- For changing the number of filters within a skip connection, we use 1×1 convolutions.

RefineNet The architecture in [1] follows a 4-cascades RefineNet [5] adjusted to account for different noise levels. In short, the RefineNet is a variant of U-Net with residual blocks, where the upwards cascade consists of RefineBlocks, each of which in turn consists of three components: *residual convolutional units* (RCU), *multi-resolution fusion* (MRF) and *chained residual pooling* (CRP). Since the architecture is complex, for brevity we refer to [5] for the exact details and to [1] for the modifications made to it. From a reproducibility standpoint, we found the wording in the architecture description ambiguous; we contacted the authors for clarifications on certain details and were referred to the official code[3]. Since the aim of this challenge was to test the reproducibility of the paper, we tried to rely only on that as much as possible, and checked the published code only when we could not make a justified assumption based on standard practice. Specifically, the aspects that were not clear from the paper, and the assumptions we made accordingly, were as follows:

- **Number of ResNet blocks**: In [5], the downward cascade (inputs to the RefineBlocks) are obtained from a ResNet pre-trained on ImageNet. In [1] an unspecified number of pre-activation ResNet blocks per cascade was trained from scratch instead. Referring to the code, we found 2 such blocks were used. In correspondence the authors said there was no specific reason for this number and that it should be robust to different choices, so we used only 1 due to computational limitations.

- **Dilated convolution**: The authors replace regular convolutions in the ResNet blocks with dilated convolutions. While the paper claims to increase the dilation by a factor of two in each cascade (which we interpreted as dilation rates 1, 2, 4, 8 corresponding to the four cascades), this would result in extremely large dilations relative to the size of the image in the lowest cascade. In the code, the authors use dilation rates 1, 1, 2, 4, while we decided to apply 1, 2, 2, 4 as a compromise between what is claimed in the paper – that dilated convolutions are not used only in the first cascade – and the aforementioned issue.

- **Downsampling**: It was not specified how and where downsampling was done in the downward cascade, so we referred to the code and found that it was performed by 2×2 average pooling with stride 2, and only after the first cascade. We would like to emphasize that the phrasing in the original paper is ambiguous in this regard, giving the impression that dilated convolutions are used to perform downsampling, when in reality this was not the case.

- **Filters**: Based on the authors' code, we used 3×3 convolutions at the beginning and end of the architecture to move between input channels and number of filters.

  Following the description in the original paper, we doubled the number of filters in all layers corresponding to the 2nd, 3rd and 4th cascades, and in the residual blocks where the number of filters is doubled, we added another 1×1 convolution to also double the filters in the skip connection, as is standard practice. However, we later found that this differs slightly from how the authors' implementation – they increase the number of filters in the final convolution of the 1st cascade, whereas we do it in the first convolution of the 2nd cascade, and they use 3×3 convolutions to increase the number of filters in the skip connections. We further noticed that the authors have 3×3 convolutions in the skip connections of the first residual block of each cascade even when the number of filters does not change.

- **Initialisation of normalisation parameters**: Since these values were not specified, we initialized the α and γ parameters for conditional instance normalisation to ones (instead on $\mathcal{N}(1, 0.02)$ as in the code), and β to zeros, as is common practice for batch normalisation.

[3] https://github.com/ermongroup/ncsn

3.3 Evaluation

Due to limited computational resources we save checkpoints during training every 10,000 iterations instead of every 5,000 as in [1]. We choose the best model for CIFAR-10 and CelebA by computing the FID on 1000 samples generated from each checkpoint. For MNIST we used the final model as in [1]. The reported Inception score and FID were computed on 50,000 samples from the best model.

4 Reproduction of toy experiments

The first experiment tests whether perturbing the data with small noise from a Gaussian distribution $\mathcal{N}(0, 0.0001)$ addresses the problem with the manifold hypothesis and facilitates learning. As in [1], we trained a ResNet encoder-decoder (Section 3.2) on CIFAR-10 with SSM loss for 50,000 iterations (Figure 1). While we do not get exactly the same magnitude or behaviour of the loss curves as in [1], we can confirm that the model only converges with perturbed data.

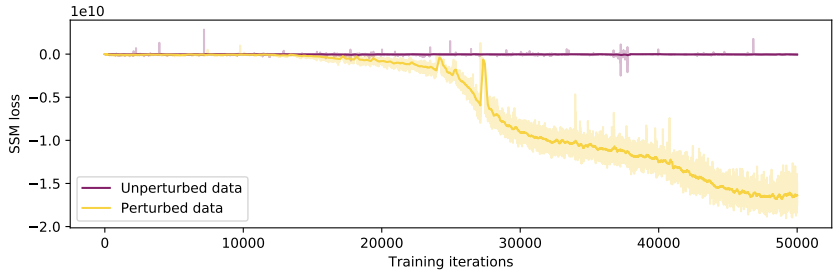
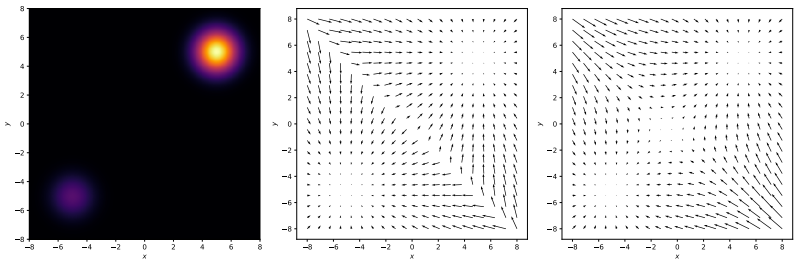


Figure 1. Loss curves for score networks trained on CIFAR-10 before and after data perturbation.

In the second experiment we compared analytically computed scores (Figure 2b) with those estimated by an MLP (Section 3.2) trained with SSM loss for 10,000 iterations (Figure 2c) for a GMM with $p(\mathbf{x}) = 0.2\,\mathcal{N}\left((-5,-5),\mathbf{I}\right) + 0.8\,\mathcal{N}\left((5,5),\mathbf{I}\right)$ with near-zero density between the components. The plots look similar to those from the original paper. We can see that scores estimation is accurate in high-density regions around the modes, but fails in low-density ones, as expected.

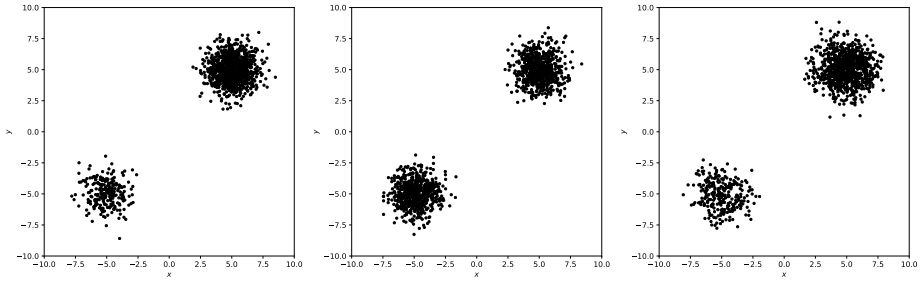


(a) Density of the data (b) True score (c) Estimated score

Figure 2. Effect of low-density regions on score estimation.

Finally, we attempted to reproduce Figure 3 from the original paper to show the effect of low-density regions on sampling using Langevin dynamics with and without annealing from the GMM from the previous experiment. Here we used the true scores computed analytically. Song and Ermon [1] claim that they used a geometric sequence of 10 noise levels σ between 10 to 0.1, $T = 100$ and $\epsilon = 0.1$. However, with this setup, we failed to

reproduce the plot (no samples were obtained because of numerical overflow).[4] When consulting with the authors' code, we found that unlike what was reported in [1], a geometric sequence between 20 and $e^{0.1}$ was used. With these values, we indeed obtained a very similar plot (Figure 3c) to what was reported in the paper.



(a) Exact samples (b) Samples without annealing (c) Samples with annealing

Figure 3. Effect of low-density regions on sampling with Langevin dynamics.

We reason that the reported hyperparameters fail due to step size $\epsilon(\sigma_1/\sigma_L)^2$, which becomes relatively large when ϵ and σ_1 are large or σ_L is small. The authors avoid the issue by increasing σ_L.[5] However, this goes against the idea of annealed Langevin dynamics, which is to converge to a distribution which is almost indistinguishable from the true one (i.e. having small σ_L is desirable). Therefore we decided to experiment with decreasing ϵ instead. We include the results in Figure 11 in the Appendix. Even with extensive fine-tuning of ϵ we did not manage to get visually satisfying results, so it is clear that the algorithm is very sensitive to the choice of this hyperparameter. We suggest a goodness-of-fit or a comparison of likelihoods as the future work to quantitatively assess the similarity of distributions.

5 Reproduction of main experiment

In this section, we present our reproduction of the main experiments from [1], which consist of training RefineNet models (Section 3.2) on MNIST, CIFAR-10 and CelebA datasets and performing sampling and inpainting with the best models. Following [1], the baseline was trained without annealing, conditioned on only one noise level ($\sigma = 0.01$), while the final model was conditioned on a geometric sequence $\{\sigma_i\}_{i=1}^{10}$ from 1 to 0.01. All models were trained for 200,000 iterations.

5.1 Image generation

One noise level The samples generated from the baseline model trained on MNIST are given in Figure 4. Following [1], we used $T = 1000$ and $\epsilon = 2 \cdot 10^{-5}$ for sampling. As we can see, the model fails to generate correct samples, which is in agreement with the original results, even for a simple dataset such as MNIST. Due to limited computational resources, we omit training the baseline for CIFAR-10 and CelebA, and focus instead on additional experiments. Overall, we consider this choice of baseline redundant, as the importance of multiple noise levels has already been showcased on the toy example (Section 4). In our opinion, a more informative baseline would be a simplified network architecture or simpler annealing schedule; we will investigate this in Section 6.

[4] We also confirmed that running the authors' code with these parameters gave the same results.

[5] This could be by mistake as the logarithm is applied to σ_1 but not to σ_L when constructing the sequence.

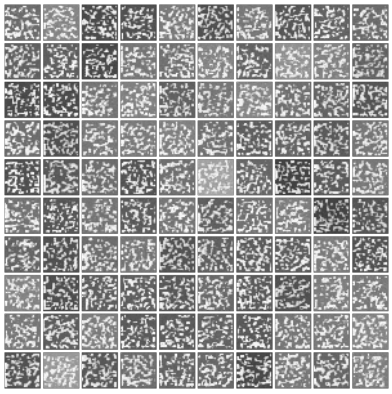


Figure 4. Samples from the baseline.

Multiple noise levels We now present our results from replicating the main experiments with models trained with a geometric sequence $\{\sigma_i\}_{i=1}^{10}$, to showcase that this approach yields reasonable samples of real data. The loss curves are given in Figure 12 in the Appendix. We chose the best model from checkpoints as explained in Section 3.3. The samples obtained from this model with $T = 100$ and $\epsilon = 2 \cdot 10^{-5}$ and intermediate samples from each q_{σ_i} are shown in Figure 3a. Additional samples are provided in the Appendix. The best model for CIFAR-10 was found at 120k iterations and for CelebA at 30k. The Inception score and FID computed from 50k samples from the final model for CIFAR-10 were $\mathbf{6.5 \pm 0.118}$ and $\mathbf{33.0}$, and for CelebA $\mathbf{3.4 \pm 0.0395}$ and $\mathbf{81.5}$.

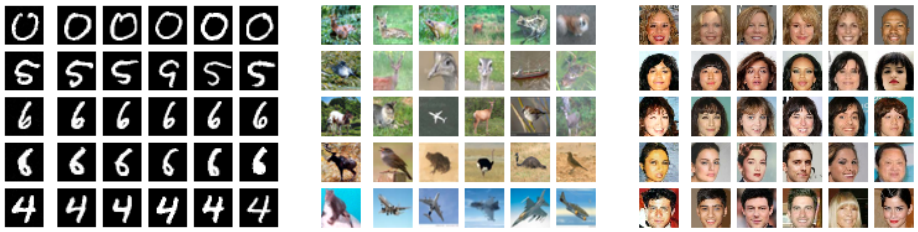
(a) MNIST samples (b) CIFAR-10 samples (c) CelebA samples

(d) MNIST intermediate (e) CIFAR-10 intermediate (f) CelebA intermediate

Figure 5. Uncurated samples generated from trained models for MNIST, CIFAR-10 and CelebA datasets with annealed Langevin dynamics (a-c) and intermediate steps during sampling process (d-f).

As we can see, while the samples are qualitatively good, our Inception and FID scores for CIFAR-10 are noticeably worse than those reported in the paper (8.87±.12 and 25.32, respectively). A possible reason could be the high variability in the FID score when choosing the best model as described in Section 3.3; we show that the FID score fluctuates for checkpoints for both CIFAR-10 and CelebA in Figure 13 in the Appendix. We believe using more samples at each checkpoint (the standard for FID is a minimum of

2048, while here we only used 1000 as per [1]) would yield a more reliable metric, but this comes at the cost of more expensive computations. There could also be some differences in the architecture that were missed in the implementation.[6] As an extension to the original paper, we also computed FID and IS for CelebA. While cropping and resizing means that results are not directly comparable to results from literature, we can see that our achieved results are significantly worse – the current reported state of the art FID score for 64×64 CelebA is 4.00 [6], while we obtained 81.5. Perhaps an interesting observation is that there are more female than male faces generated, which also reflects the ratio of these in the training data. In the interest of reproducability, we also report the time cost of training and sampling for each dataset in Table 1 in the Appendix.
Following [1], we find k nearest neighbours for a set of generated samples from the training data with respect to l_2 distance. Results are shown in Figure 6, and more extensive examples in Figure 17 in the Appendix. As we can see, the network has not memorised exact training images, but still preserves some high-level features from the training set such as colour, shape, style, orientation etc.[7]



(a) MNIST (b) CIFAR-10 (c) CelebA

Figure 6. $k = 5$ nearest neighbours from the training data for 5 samples based on l_2-distance.

5.2 Inpainting

We performed inpainting as in [1], occluding the right half of the image and completing the image with annealed Langevin dynamics. The results are shown in Figure 7. As we can see, the models perform extremely well, especially in recovering the background for CIFAR-10 and the right half of the face for CelebA. Moreover, inpaintings for the same image are diverse, which showcases the generalisation capacity of the approach. Furthermore, one advantage of this method is that it is able to handle arbitrary occlusion shapes well (Figure 18 in the Appendix). A nice addition would be to quantify the quality of results by measuring signal-to-noise ratio, but we leave this as future work.

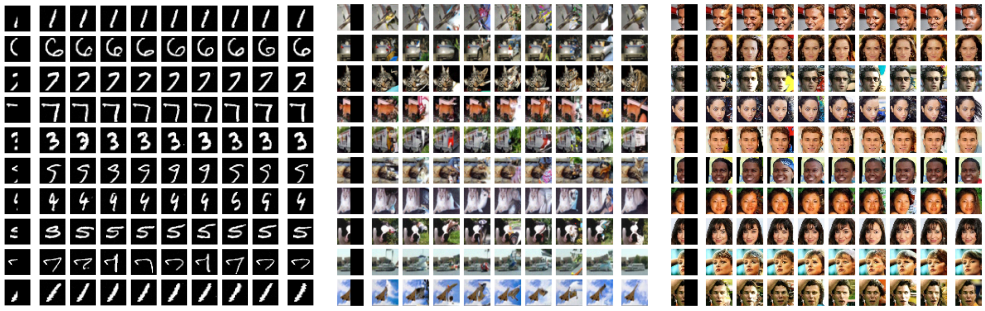
6 Additional experiments

After successfully reproducing most of the experiments, we thought of some interesting extensions. Specifically, we investigated the choice of sampling hyperparameters ϵ and T, annealing schedule and network architecture, which we identified as important but unexplored in the original work.

6.1 Sensitivity on sampling parameters

The authors report the interval $\epsilon \in (5 \cdot 10^{-6}, 5 \cdot 10^{-5})$ and $T = 100$ as robust hyperparameters for sampling. Motivated by sensitivity to ϵ in the toy examples, we investigate

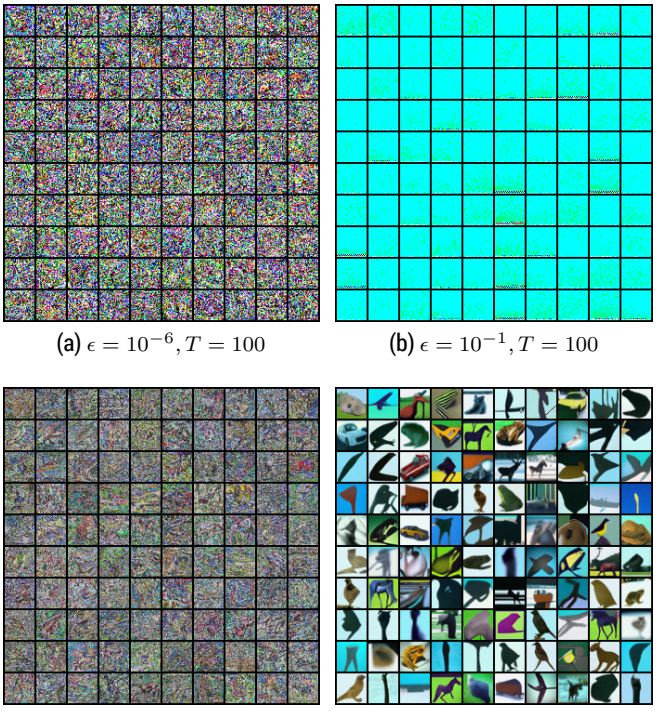
[6] We initially considered the number of residual blocks, but with 2 blocks the results did not improve.
[7] Disclaimer: We implemented this before the authors added nearest-neighbour calculations to their work on 29th Oct 2019, which additionally measures distance based on activations from the Inception network.



(a) MNIST (b) CIFAR-10 (c) CelebA

Figure 7. Inpainting for occluded images from the training set. The occluded image is given in the leftmost column, and the true image in the rightmost column.

these hyperparameters in more detail on real data from the CIFAR-10 dataset. We sampled with a smaller ϵ (Figure 8a) and a larger ϵ (Figure 8b), where the former yielded only noise since the step was not large enough to move away from the noisy initial point, and the latter resulted in samples being one-coloured because too large step size escaped allowed pixel values and was artificially clipped to the endpoints of pixel intervals. Similarly, a large T results in smoother images (Figure 8d), but with much fewer details, probably due to convergence to a local mode representing some kind of generic class image, and a small T (Figure 8c) being mostly noise with only slightly distinguishable shapes. While the reasoning here is speculative, we here would like to emphasise the sensitivity to choice of ϵ and T, as already seen in the toy GMM experiment in Section 4. A suggested future work might concern different annealing strategies where ϵ or T could be adjusted for different noise levels.



(a) $\epsilon = 10^{-6}, T = 100$ (b) $\epsilon = 10^{-1}, T = 100$

(c) $\epsilon = 2 \cdot 10^{-5}, T = 10$ (d) $\epsilon = 2 \cdot 10^{-5}, T = 10^3$

Figure 8. Samples from the best CIFAR-10 model with different values of ϵ and T.

6.2 Linear annealing schedule

We also train a model with linear annealing schedule instead of geometric on CIFAR-10 dataset. We hypothesise that geometric annealing might be too aggressive in the sense that most of noise values are low, which results in fine-tuning edges without capturing global features first. This is manifested in samples which were smooth but lacked a distinct shape of any expected object. From visual inspection of the results (Figure 9) we can see that intuition was correct, as these samples capture much more detail and complexity than with geometric annealing (Figure 5b). However, the images are visibly more noisy. We can explain this tendency using the inverse of the previous logic, namely that there are not enough noise levels in the low spectrum of values, thus precluding the sampling of sharper images. We believe some combination of these annealing schedules could solve this issue and result in very good samples, but also add to the complexity of the model.

(a) 50k iterations (b) 100k iterations

(c) 150k iterations (d) 200k iterations

Figure 9. Samples from model trained on CIFAR-10 with linear annealing schedule.

6.3 Different architecture of the score network

The authors emphasise the choice of network architecture as important role and leave experimenting with different designs as future work. We address this by replacing the U-Net structure with the ResNet from the toy example in Section 4 on CIFAR-10, but to avoid changing the architecture too much, we add conditional instance normalisation and dilated convolutions, as well as multiplying the number of filters by factor of 4 in each block to match the number of parameters in the RefineNet. Here we can only conclude that this simple architecture did not result in meaningful samples and that the choice of structure seems to be important. We believe the main reason for these subpar results is lack of skip connections as in U-Net type architectures. We also acknowledge that training hyperparameters would have to be tuned for this network to

meaningfully compare the results. However, when taken in combination with the results from the main experiments in Section 5, and assuming that our hypothesis – that minor differences in RefineNet architecture between our implementation and the one from the original paper contribute to significantly worse performance in terms of FID score – is correct, this would suggest that the method is perhaps overly reliant on architectural details. We suggest that more investigation is needed to determine exactly which aspects of the score network architecture are crucial for good performance.

(a) 100k iterations (b) 150k iterations (c) 200k iterations

Figure 10. Samples from a ResNet model trained on CIFAR-10.

7 Conclusions

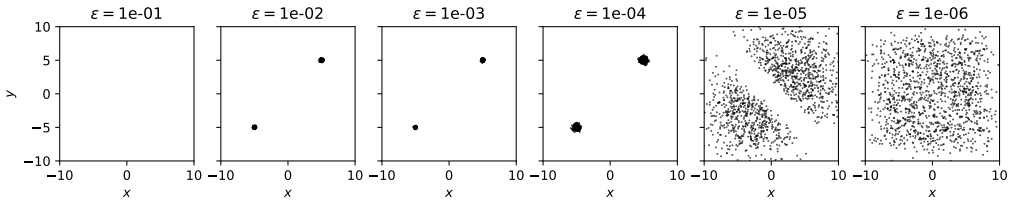
We can conclude that the main results from [1] concerning image generation and inpainting are qualitatively reproducible and that the method indeed does yield visually good samples and inpainting results. However, we did not manage to obtain state-of-the-art Inception score on CIFAR-10 (nor comparable FID) as in [1], perhaps due to mentioned differences in the score network (which additional results show to be an important aspect of the method) or highly variable evaluation metric. When it comes to toy experiments used to investigate challenges and motivate assumptions of the proposed method, we report partial irreproducibility due to incorrectly reported noise levels for the annealed sampling from the given GMM. This observation led us to investigate the effect of different hyperparameters on sampling for CIFAR-10, which showed relatively high sensitivity to ϵ and T. We also experimented with a different architecture, and linear annealing schedule. While with the former we did not manage to generate reasonable samples (perhaps due to simplicity of the architecture), the latter yielded more detailed (but slightly more noisy) samples than the default geometric annealing, thus paving the path for future improvements.

References

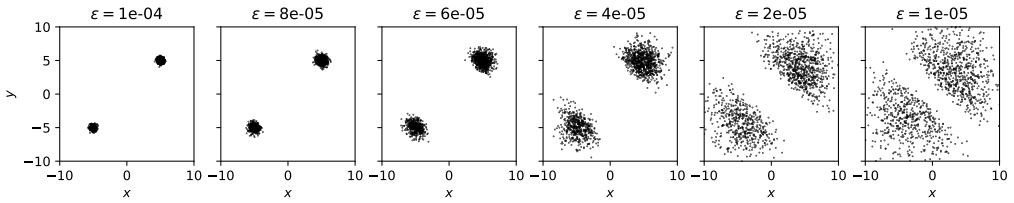
1. Y. Song and S. Ermon. "Generative Modeling by Estimating Gradients of the Data Distribution." In: **CoRR** abs/1907.05600 (2019). arXiv: 1907.05600.
2. Y. Song, S. Garg, J. Shi, and S. Ermon. "Sliced Score Matching: A Scalable Approach to Density and Score Estimation." In: **CoRR** abs/1905.07088 (2019). arXiv: 1905.07088.
3. P. Vincent. "A Connection Between Score Matching and Denoising Autoencoders." In: **Neural Computation** 23.7 (July 2011), pp. 1661–1674.
4. M. Welling and Y. W. Teh. "Bayesian Learning via Stochastic Gradient Langevin Dynamics." In: **Proceedings of the 28th International Conference on International Conference on Machine Learning**. ICML'11. Bellevue, Washington, USA: Omnipress, 2011, pp. 681–688.
5. G. Lin, A. Milan, C. Shen, and I. D. Reid. "RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation." In: **CoRR** abs/1611.06612 (2016). arXiv: 1611.06612.

6. C. H. Lin, C.-C. Chang, Y.-S. Chen, D.-C. Juan, W. Wei, and H.-T. Chen. "COCO-GAN: Generation by Parts via Conditional Coordinating." In: **CoRR** abs/1904.00284 (2019). arXiv: 1904.00284.

Appendix



(a) $\epsilon \in \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$.



(b) $\epsilon \in \{10^{-4}, 8 \cdot 10^{-5}, 6 \cdot 10^{-5}, 4 \cdot 10^{-5}, 2 \cdot 10^{-5}, 10^{-5}\}$.

Figure 11. Samples obtained with different values of ϵ for toy data with annealed Langevin dynamics. Even after expanding the search in the promising interval $\epsilon \in \{10^{-4}, 10^{-5}\}$, we did not find a value of ϵ yielding satisfying results.

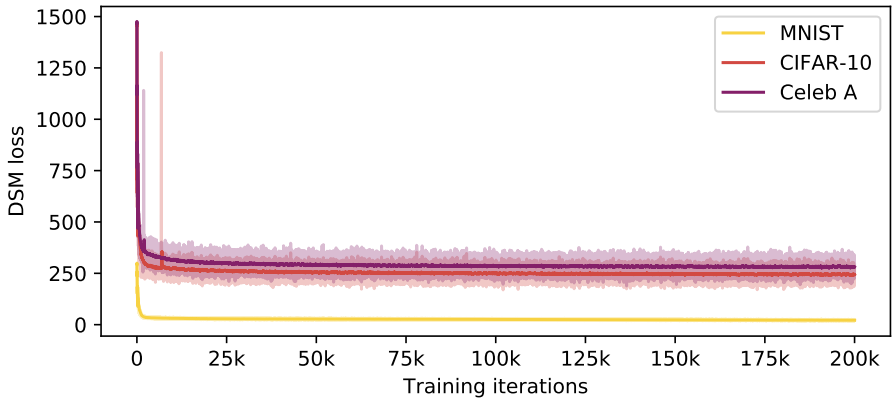


Figure 12. Loss curves for training real model

(a) CIFAR10 **(b)** CelebA

Figure 13. FID scores on 1000 samples for the model trained up to given iteration

	Dataset		
	MNIST	CIFAR-10	CelebA
Downloading data (required once)	28.7s	50.4s	715.4s
Training with RefineNet	2.35it/sec	1.73it/sec	1.73it/sec
Sampling (1 image)	19s	23s	23s
Sampling (100 images)	112s	158s	158s
Sampling (1000 images)	910s	1398s	1398s

Table 1. Time required for different components of the main experiment for each dataset. MNIST was run on a P100 GPU, while the other two datasets were run on a V100 GPU.

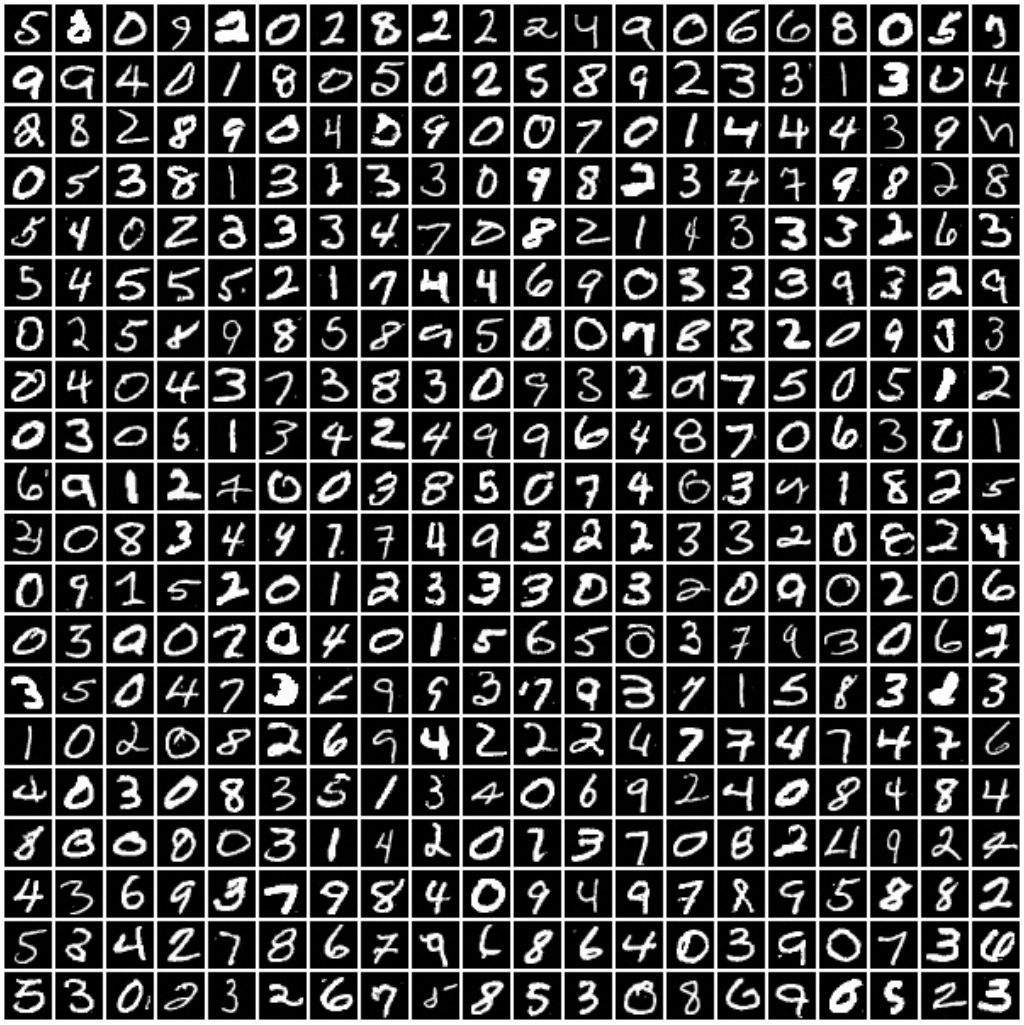
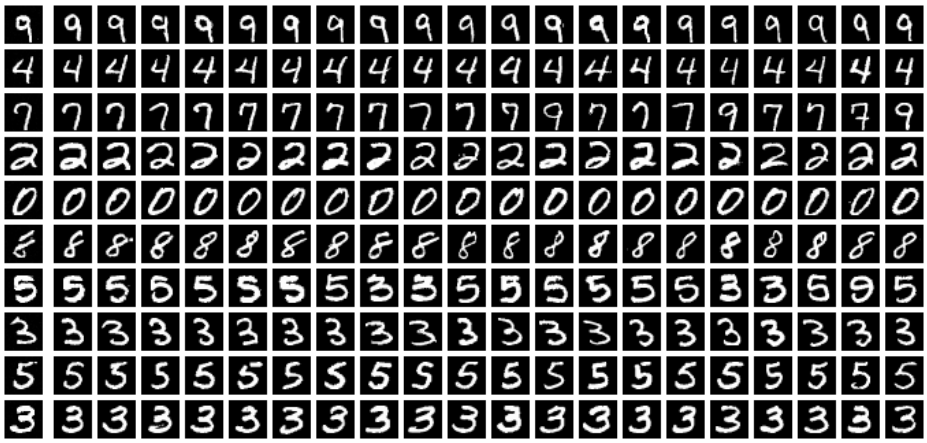


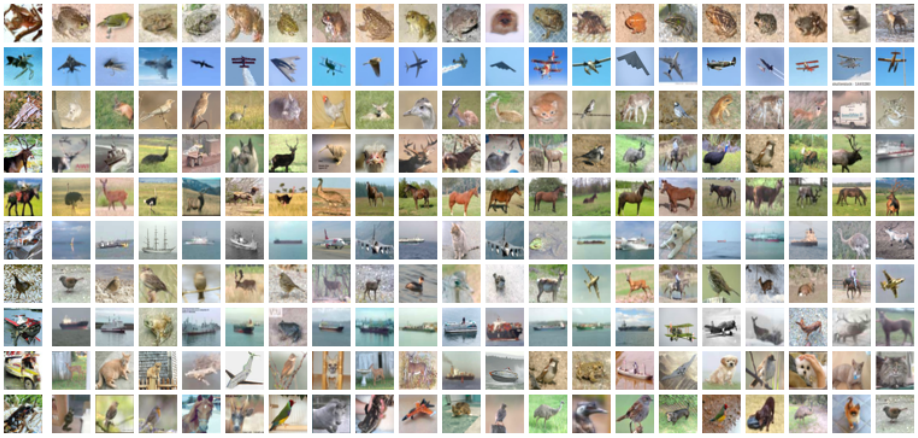
Figure 14. Extended samples from MNIST

Figure 15. Extended samples from CIFAR10

Figure 16. Extended samples from CelebA



(a) MNIST



(b) CIFAR-10

(c) CelebA

Figure 17. Samples (leftmost column) and their nearest neighbours from training set w.r.t. l_2 distance.

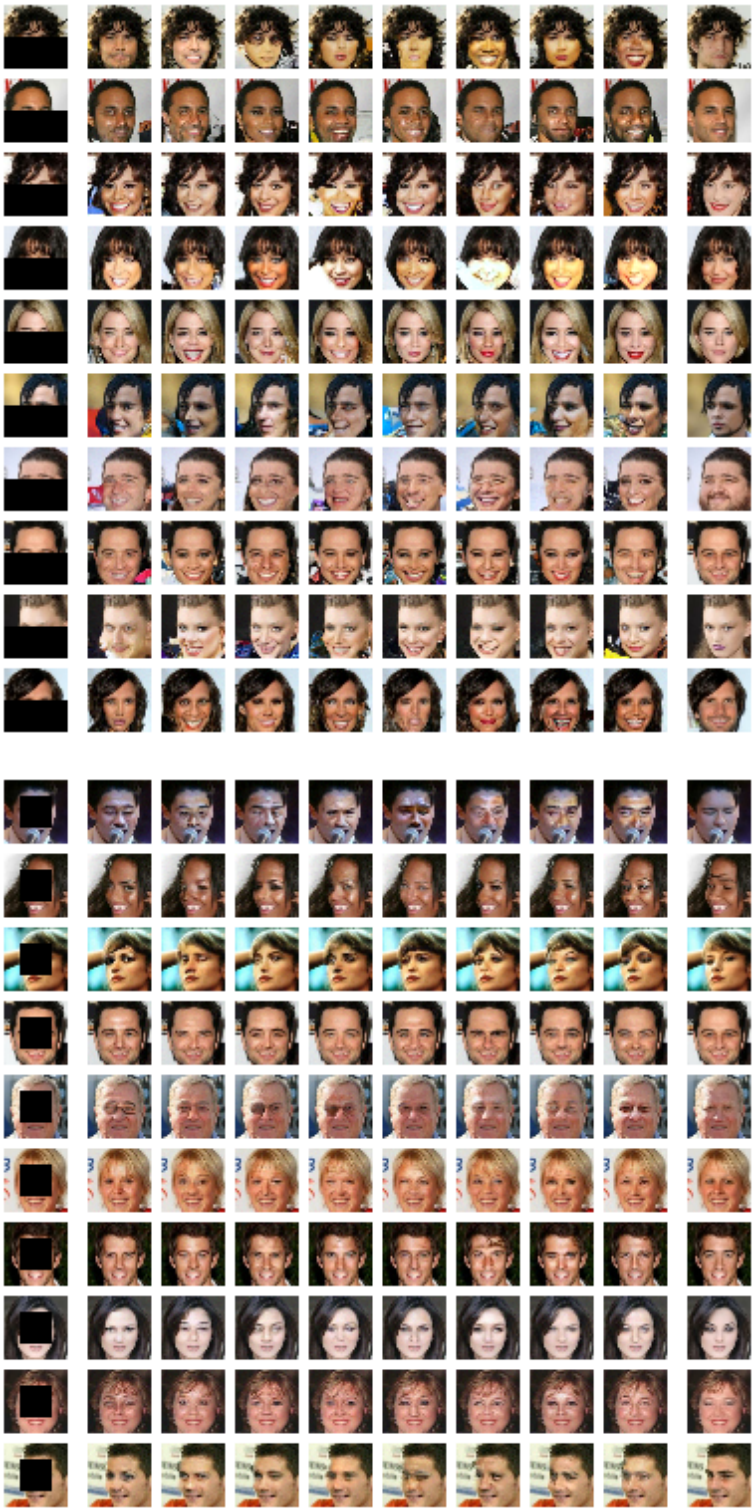


Figure 18. Recontruction with two more different patterns of occlusions on CelebA. Note that unlike the left-right occlusions shown in the main results, utilising face symmetry is not possible with these occlusions, once again proving the method has a good generative capacity.