Using Intermediate Forward Iterates for Intermediate Generator Optimization

Harsh Mishra[1] **Jurijs Nazarovs**[2] **Manmohan Dogra**[1] **Sathya N. Ravi**[1]

[1]Department of Computer Science, University of Illinois at Chicago
[1]Department of Statistics, University of Wisconsin Madison

Abstract

Score based models have recently been introduced as a richer framework to model distributions in high dimensions, and in general more suitable for generative tasks. In particular, in score based models a generative task is formulated using a parametric model (such as neural network) to directly learn the gradient of such high dimensional distributions, instead of the density functions themselves, as is done traditionally. From the mathematical point of view, such gradient information can be utilized in reverse by stochastic sampling to generate diverse samples. However, from a computational perspective, existing score based models can be efficiently trained only if the forward or the corruption process can be computed in closed form. By using the relationship between the process and layers in a feed forward network, we derive a backpropagation based procedure which we call *Intermediate Generator Optimization* to utilize intermediate iterates of the process with negligible computational overhead. The main advantage of IGO is that it can be incorporated in any standard autoencoder pipeline for generative task. We analyze the sample complexity properties of IGO to solve downstream tasks. We show applications of the IGO on two dense predictive tasks viz., image extrapolation, and point cloud denoising. Our experiments indicate that it is possible to obtain ensemble of generators for various time points using first order methods.

1 Introduction

Modular priors defined using generative models have enabled a broad spectrum of applications in our vision community. For instance, it is now a well accepted notion that ill-posedness in many problems such as image inpainting, denoising, compressed sensing can be mitigated effectively in practice using such priors. However, recent theoretical and empirical results suggest that the *range* of such priors is an important aspect that governs the performance of any downstream analysis that utilizes data from generative models. To see this, consider a Deep Generative Model (DGM) $G(z) : \mathbb{R}^k \to \mathbb{R}^n$ where z represents the random variable representing noise, n and k represent the generated data (image) and noise dimensions respectively. Then, it turns out that, as long as the range of G is incoherent with respect to the linear measurement matrix A, sparse linear inverse problems defined by $y = AG(z)$ can be solved efficiently using simple gradient descent schemes [11, 24].

Generative Adversarial Networks (GANs) were first shown to be successful in generating high resolution realistic natural images [36], and biomedical images [12] for augmentation purposes. Variational Autoencoders (VAE) is a popular alternative that is based on minimizing the *distortion* given by integral metrics such as KL divergence. In both GANs and VAE, the learning problem coincide – we seek to learn the process of generating new samples based on latent space modelled as random distributions [14, 29, 38]. Applications enabled by DGMs in vision include style transferring [39],

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

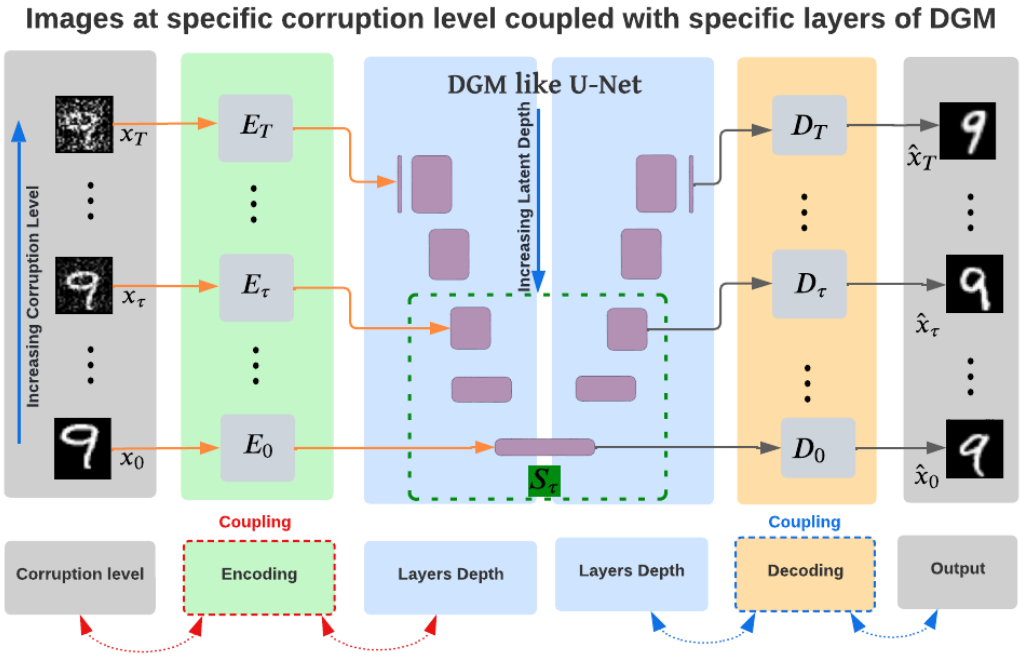


Figure 1: Figure shows our idea of coupling iterates from a given corruption process with Generators.

inpainting [6], image restoration and manipulation [25]. VAE architectures have been successfully deployed in temporal predictions settings owing to its robustness properties [22, 31]. Despite the excellent performance of DGMs like GAN and VAE, recent results indicate that there are some severe limitations with respect to their range [26].

Why is diversity of G important for Downstream Tasks? To improve robustness of prediction, adversarial training [10, 20, 15] is a common strategy used in practice. In essence, in these settings, it is critical that the model performs well both on the given training distribution, and its corresponding adversarial distribution. While this has shown to be an effective strategy, there are severe limitations for large scale applications. Firstly this increases the overall training time of the parameters. Secondly, it also increases the memory required for inner optimization subroutine to sample from the adversarial distribution. **Our main insight is that if G can generate samples from both these distributions, then we can sidestep the inner optimization subroutine completely.** Note, that we do not claim that IGO can be used to generate adversarial samples for adversarial training purposes.

Our Contributions. Motivated by our applications in various large scale vision applications, (i) we propose a new fully differentiable framework that explicitly models the corruption process within DGMs, and which can be trained using standard backpropagation procedures like SGD and its variants; (ii) we analyze the statistical properties of our proposed framework called Intermediate Generator Optimization (IGO) using recently introduced Intermediate Layer Optimization (ILO) framework [5] for solving inverse problems and moreover; (iii) we show the utility of our procedure for image generation and prediction tasks; (iv) In the challenging, three dimensional point cloud setting, we identify potentially beneficial regularizers for improving the robustness profile of denoisers; (v) We also provide support for two simulatable processes, Lotka-Volterra [17] and Arnold's Cat Map [1], which can be used as Data augmentation schemes, see supplementary for more details.

Roadmap. In Section 2, we propose to use efficient discretization strategies as a tool to increase the range of DGMs. In Section 3, using Information Bottleneck Principle, we introduce the notion of intermediate generator regularizer for utilizing iterates of discrete process during training. We argue that our regularizer can be efficiently optimized, and analyze its sample complexity for linear inverse problems. We discuss various vision use cases of IGO such as image generation, dense extrapolation, and point cloud denoisers along with our experimental results in Section 4. We also

present a discussion about why SDE based loss function automatically induces diversity in samples Generated by a generative model in the supplement.

2 Improving Range via Discretization

Basic Setup. A Stochastic Differential Equations (SDE), where f and g are the diffusion and the drift functions respectively, and w is the Standard Brownian Motion is represented by 1:

$$\mathrm{d}\mathbf{x} = f(\mathbf{x}, t)\,\mathrm{d}t + g(t)\,\mathrm{d}w. \tag{1}$$

We can map data to a noise distribution using the forward SDE in equation (1) and reverse the SDE for generation using

$$\mathrm{d}\mathbf{x} = \left[f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right]\mathrm{d}t + g(t)d\bar{\mathbf{w}}, \tag{2}$$

where $\bar{\mathbf{w}}$ is the Brownian motion when time flows backwards from T to 0, and $\mathrm{d}t$ is an infinitesimal negative timestep. Crucially, as shown by [32], it is possible to learn the *gradients* $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ by training a time dependent score based model $s_{\boldsymbol{\theta}}(\mathbf{x}, t)$. In essence, an approximate model s will allow us to generate diverse, yet realistic samples using small amount of independent random noise controlled by g.

During training, we seek to solve the following distributional optimization problem:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_t \left[\lambda(t)\mathbb{E}_{\mathbf{x}(0)}\mathbb{E}_{\mathbf{x}(t)|\mathbf{x}(0)} L(s_\theta, x(t), t)\right], \tag{3}$$

where $L(s_\theta, x(t), t)$ is usually chosen as the squared loss given by $\left[\left\|s_{\boldsymbol{\theta}}(\mathbf{x}(t), t) - \nabla_{\mathbf{x}(t)} \log p_{0t}(\mathbf{x}(t) \mid \mathbf{x}(0))\right\|_2^2\right]$. After training, we may simply replace the log-likelihood in the reverse SDE by our model s in equation (2) for sampling purposes. We refer to this loss function as the standard (or usual) loss function

The key principle in Score based DGMs is that generator G learns the *gradient* of the log-likelihood function explicitly instead of the density function itself. There are two benefits associated to this approach: (i) statistical range of the generator can be naturally improved during sampling since gradients are finer information (function values can be approximated using gradients by Taylor's theorem, see Remark 1), and (ii) computationally tractable alternative to kernel density estimation since it does not require us to estimate the normalizing constant which is usually intractable in high dimensions.

Remark 1. *Why Gradients are considered to be finer information?*

Kernel Density Estimation – smooth estimator of Probability density function – is known to converge faster than histograms (function values). Hence Gradients are termed to be finer information than function values. Refer Chapter 20 in Larry Wasserman's All of Statistics [35].

Our Assumption on Forward Process. Given a stochastic differential equation model given in equation (1), our goal is to incorporate the individual trajectory information while training the score based model. In order to do so, we assume that we have access to an efficient discretization of the process in equation (1). In our case, we say that the forward SDE equation (1) can be effectively parameterised using the Euler-Maruyama (EM) method. We will use $\tilde{\mathbf{x}}$ to denote the approximation to $\mathbf{x}$ provided by the EM discretization. For a time-varying multiplicative process defined by $\mathrm{d}\mathbf{x}_t = a(\mathbf{x}_t, t)\,\mathrm{d}t + b(\mathbf{x}_t, t)\,\mathrm{d}w_t$, the EM algorithm executes the following iterations:

$$\tilde{\mathbf{x}}_{j+1} = \tilde{\mathbf{x}}_j + a(\tilde{\mathbf{x}}_j, t)\Delta t + b(\tilde{\mathbf{x}}_j, t) z\sqrt{\Delta t}, \tag{4}$$

where $\tilde{\mathbf{x}}_j$ denotes the $j-$th iterate and $z \sim \mathcal{N}(0, 1)$ is a sample drawn uniformly at random from the standard normal distribution. Note that the process defined in (1) satisfies our assumption here since we allow g to depend on X_t.

3 Efficient DGMs for Discrete Processes

In order to provide guarantees on DGM based downstream tasks such as solving inverse problems, we propose a new loss function to train the score based model s based on individual trajectories of

samples corrupted by the discretized process defined in (4). We begin by writing the empirical finite sample form of the optimization problem in (3) as follows,

$$\min_{\boldsymbol{\theta}} \sum_{\tau=t_1}^{t_T} \lambda(\tau) \sum_{i=1}^{n} \mathbb{E}_{\mathbf{x}_i(\tau)|\mathbf{x}_i} L\left(s_\theta, \mathbf{x}_i(\tau), \tau\right), \tag{5}$$

where T represents the discretization size, and $0 \leq t_i \leq 1 \; \forall \; i = 1, \ldots, T$. To solve the optimization problem in (5), the most popular choice is to use first order backpropagation type methods. Importantly, the worst case complexity of solving (5) scales linearly with the discretization size T, which is intractable in large scale settings. By chain rule, the efficiency of such algorithms strongly depends on the ease of evaluating the derivative of the forward operator, denoted by the conditional expectation. While previous works assume that there is a closed form solution to evaluate the conditional expectation, such an assumption is invalid in our examples discussed above. If $T = 1$, then we may simply simulate the dataset $\{\mathbf{x}_i\}$ using (4) to obtain $\tilde{\mathbf{x}}_i(t_1)$, and use it to compute the loss, and backpropagate.

3.1 Handling $T = 2$ case using IBP

Now we consider the setting in which we have access to only one intermediate iterate $\tilde{\mathbf{x}}_\tau$ where $\tau < t$ for some arbitrary $t \sim \mathcal{U}(0, 1)$ in our trajectory. Naively, we can train two DGMs in parallel, one each for $\tilde{\mathbf{x}}_t$ and $\tilde{\mathbf{x}}_\tau$, thus incurring twice the memory and time complexity, including resources spent for hyperparameter tuning. We propose a simpler alternative through the lens of so-called Information Bottleneck Principle (IBP) – the de-facto design principle used in constructing standard Autoencoders architectures such as U-Net.

Coupling Forward Processes with Intermediate Iterates. In feedforward learning, the main result due to IBP is that layers in a neural network try to *compress* the input while maximally preserving the relevant *information* regarding the task at hand [9]. For designing neural networks, this corresponds to choosing a sequence of transformations T_l such that the distance between successive transformations $d(T_l, T_{l-1})$ is not too big. In practice, we can ensure this by simply choosing dimensions of layerwise weight matrices by a decreasing function of layers (or depth). Intuitively, the idea is that if dimensions of T_l is a constant factor of T_{l-1}, then at optimality (after training), when random input passes via T_l, then a constant factor of noise available when it passed through T_{l-1} will be removed, in expectation. Mutual Information between two random variables is well defined only if the support of the random variables are exactly the same. So, if the weights of two successive layers are close to identity, then the support will be identical, the mutual information will be maximized, and vice-versa.

IGO for $\tilde{\mathbf{x}}_\tau$ iterates. By using the correspondence between the forward process defined in (4) and intermediate layers in a DGM, we will now define our loss function for intermediate iterates $\tilde{\mathbf{x}}_\tau$. We define the regularization function $\mathcal{R}$ as follows,

$$\mathcal{R}(\tilde{\mathbf{x}}_\tau, \tau) = L\left(E_\tau \circ s_\tau \circ D_\tau, \tilde{\mathbf{x}}_\tau, t\right), \tag{6}$$

where s_τ (dark green box in Figure 1) denotes the *restriction* of the score based model s to iterate $\tilde{\mathbf{x}}_\tau$, E_τ denotes a shallow encoder for $\tilde{\mathbf{x}}_\tau$, and similarly for the decoder D_τ. We will refer to the DGM defined by $E_\tau \circ s_\tau \circ D_\tau$ as the **Intermediate Generator**, see Figure 1. Intuitively, the regularization function $\mathcal{R}$ is defined to modify parameters of the model s to a specific set of connected layers given by s_τ. To see this, first note that the overall parameter space $\boldsymbol{\theta}$ can be seen as a product space over layers θ_l. Then, at any given training iteration, $\mathcal{R}$ function *restricts* the update to layers that are suitable for decoding $\tilde{\mathbf{x}}_\tau$.

Interpreting $\mathcal{R}$. Using small step size in EM guarantees us that a larger class of SDEs can be simulated see [23]) , so $\tilde{x} \sim x$. So, let us fix a sufficiently small Δt, so $\tilde{\mathbf{x}} \approx \mathbf{x}$ almost everywhere at any time t. Now, when the forward process (4) is a *Markov* process, that is, increments of g is independent of t, then the intermediate iterates do not carry any more information. In fact, in practice, it may be detrimental to the training process if the gradient directions are not sufficiently correlated. Unfortunately, this assumption is not true in our discrete setting considered here since both the drift a, and diffusion b functions are time-varying. In this case, $\mathcal{R}$ simply tries to revert time dependent noise process in the relevant part of the network s_τ using appropriate E_τ, and D_τ. Our regularization function provides time dependent noise information explicitly by using the intermediate iterate $\tilde{\mathbf{x}}_\tau$ during training in appropriate parts of DGM.

Extension to Multiple Iterates $T > 2$. It is straightforward to extend our regularization function when we are given with more than one intermediate iterate from the discretized EM algorithm by,

$$R_0^T(\tilde{\mathbf{x}}) = \sum_{\tau=t_1}^{t_T} L\left(E_\tau \circ s_\tau \circ D_\tau, \tilde{\mathbf{x}}_\tau, t\right). \tag{7}$$

Implementation. IGO utilizes less computational resources in the following sense – if E_τ, and D_τ, relatively shallow networks (as shown in the figure), then the cost of computing gradients with $\tilde{\mathbf{x}}_\tau$ is negligible compared to $\tilde{\mathbf{x}}$, thus achieving cost savings by design.

3.2 Theoretical Analysis for Downstream Tasks

We utilize the key result in ILO to analyze our procedure for downstream tasks. To that end, we assume that the parameters of s, E_τ and D_τ are fixed (given by a pretrained model), and show that IGO is suitable for solving inverse problems under low sample settings.

Necessary Condition on DGM. We follow the same observation model as in [5] given by $y = Ax + \mathfrak{n}$ where $\mathfrak{n}$ is a random variable representing noise, and x is the unknown. It is well known that when the measurement matrix A satisfies certain probabilistic requirements, then it is possible to solve for x using a single (sub-)gradient descent scheme [3, 33]. This is specified using the S-REC condition in DGM based prior modeling using CSGM algorithm [2]. An example which satisfies the probabilistic requirements is when the entries in A are distributed according to Gaussian distribution, as mentioned in [5].

In order to analyze $\mathcal{R}$ for prior modeling purposes, we assume that after training the generator has a compositional structure given by $G_1 \circ G_2$, which is followed in most standard architectures. The following observation is crucial in analyzing IGO for compressive sensing purposes.

Observation 1. (Range expansion due to E_τ.) *Setting the intermediate generator (of s) to be G_τ, the range of our pretrained model G_1 is increased to,*

$$\underline{G} := G_\tau(B_2^k(r_1)) \oplus E_\tau(B_2^{k_\tau}(r_\tau)), \tag{8}$$

where $\oplus$ denotes the Minkowski Set sum, and k_τ corresponds to the dimensions of intermediate code associated with $\tilde{\mathbf{x}}_\tau$.

With the increased range in Observation 1, we show that using the overall generator trained with IGO as a prior can be sample efficient in the following lemma. We make the same assumptions as in Theorem 1. in [5] on the entries of A, and smoothness of G_2.

Lemma 1. *Assume that the Lipschitz constant $\underline{G}$ is $\underline{L}$, and we run gradient descent to solve for the intermediate vector (as in CSGM). Assume that the number of measurements m in y is at least*

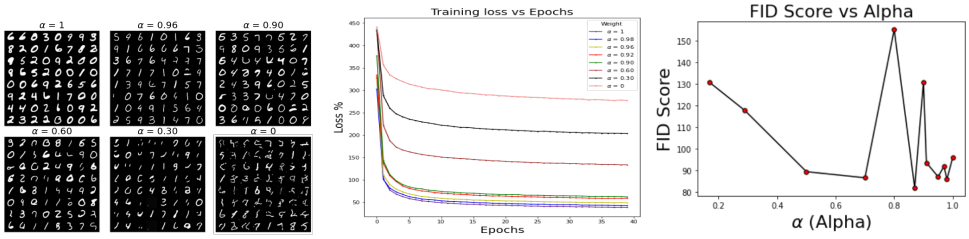
$$\min(k \log(L_1 L_2 r_1/\delta), k_\tau \log(\underline{L} L_2 r_\tau/\delta)) + \log p, \tag{9}$$

where p is the input dimensions of G_2 or the intermediate vector. Then, for a fixed p, we are guaranteed to recover an approximate x with high probability (exponential in m).

Please see supplement for the proof of the signal recovery lemma 1. Our proof uses the (now) standard $\epsilon-$net argument over the increased range space [34]. In essence, if the Lipschitz constant of the intermediate generator $\underline{\mathrm{L}}$ is lower than the lipschitz constant of G_1, then we get an improved sample complexity bound for IGO.

Intuitively, lemma 1 states that the number of samples required during downstream processing depends on two factors: 1. *latent dimension size* denoted by the $\log p$ term which remains the same with or without intermediate iterates, and 2. *smoothness* denoted by the Lipschitz constants $L_1, \underline{\mathrm{L}}, L_2$. While we have no control over the size p, if the intermediate generator is smoother, and performs well on training data y, then gradient descent succeeds in recovering the "missing" entries with fewer number of samples. In the case when intermediate iterates are not useful, then our framework allows us to simply ignore the intermediate generator without losing performance. In other words, our framework preserves hardness of recovery – easy problems remain easy.

Remark 2 (Difference between proposed IGO and ILO.)**.** *The input of the ILO algorithm is pretrained DGM with the goal of tuning the noise distribution for better image generation using gradient descent schemes. In IGO, we use the knowledge of forward process to optimize the parameters of the DGM to increase its range* **during** *training.*



(a) Generated Samples (b) Training Loss vs Epochs (c) FID Scores

Figure 2: IGO based Experimental Results on MNIST dataset with one intermediate iterate, produced using EM Algorithm with Arnold Cat Map as the drift function, and scaled Gaussian as the Diffusion function.

Leveraging a diverse G for generic Downstream Tasks. Consider a feature extractor (or an encoder) induced by an appropriate SDE guided DGM, that has the ability to extract *relevant* features from a mixture of two distributions, for example, training, and adversarial. In this setting, if we know the distribution from which a given feature has been sampled from, then we can hope to predict the sample at an optimal error rate by using a neural network with sufficient layers. Indeed, when the mixtures have a natural dependency (given by SDE (1)), then we may simply use weight sharing instead of training two separate models, as shown in Figure 5 - left. Thus, achieving memory as well as some time savings. Assume that a DGM has total of P parameters. Then, training $|T|$ DGMs each for a time step of a discretization size $|T|$ would require $O(P|T|)$ parameters. However in our approach, say we require p extra parameters for E_τ, D_τ, then the total number of parameters in our IGO framework is $O(P + (p-1)|T|) = O(P + |T|)$, for small values of p – as is the case in our experimental settings – this can be a huge reduction in the number of parameters $O(P|T|)$ in standard framework vs $O(P + |T|)$ in our IGO framework.

4 Experiments

In this section we explore different applications of IGO across varying setups to showcase our idea of utilizing an intermediate iterate $\tilde{\mathbf{x}}_\tau$. First, we train a score based generative model from scratch under the presence of non-linear corruption process such as Arnold Cat map in Section 4.1. Then, show the utility of intermediate features for prediction. In Section 4.2, we tackle the challenging trajectory prediction tasks, in particular, we modify the ODE to take inputs from the intermediate, and final iterates during training. Finally, In Section 4.3 we demonstrate the use of IGO in a *dense* prediction task of denoising posed as a supervised learning problem. Here, a multi-layer perceptron is utilized as a score based model for denoising purposes.

4.1 IGO with Nonsmooth Drift on U-Net

4.1.1 Setup Details

In standard score based generation, the transition kernel is assumed to be a multivariate Gaussian distribution. Given a clean image $\mathbf{x}(0)$, we can apply the kernel using closed form evaluation to obtain the noisy image $\mathbf{x}(t)$ for random $t \sim \mathcal{U}(0, 1)$. Neural networks are then used to estimate the score, also known as the time-dependent gradient field, to reverse the corruption process.

Here, we start off with explicitly modeling our corruption process using Arnold's Cat Map as the drift function of our SDE, to setup Eqn. (4). Our goal here is to utilize our proposed Intermediate Generator Optimization, for training purposes. To do so, we simulate the forward SDE till some random time t and store an (additional) intermediate iterate for every trajectory to obtain $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_\tau$ ($\tau = t/2$). The two iterates are then used to set the training objective based on Eqn. (5) with our proposed regularizer in Eqn. (6). As per Figure 1, the intermediate iterate $\tilde{\mathbf{x}}_\tau$ has its own intermediate pathway s_τ into our overall model s.

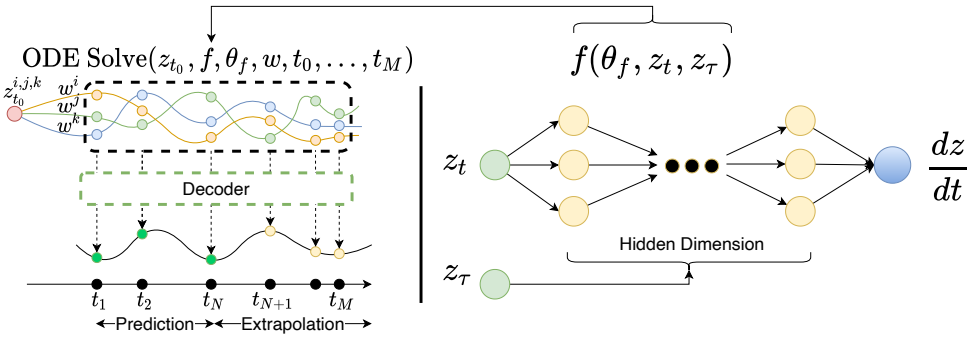


Figure 3: Incorporation of intermediate iterates in Mixed Effect Neural ODE for analyzing the dynamics of Panel data.

4.1.2 Experimental results

We choose a simple setting with MNIST dataset to test our IGO formulation. A U-net [30] architecture which uses Gaussian random features to encode each trajectory, forms the backbone of our score model. During the forward pass, our simulated iterates $\mathbf{x}_t$ and $\tilde{\mathbf{x}}_\tau$. are passed into the network, refer Figure (1). We use a convolution layer with a kernel of size 5 and stride 2 as the intermediate encoder E_τ and a de-convolution layer with kernel of size 6 and the stride 2 as the intermediate decoder D_τ. The training is done using a convex combination of loss, defined using the intermediate and the final iterate, $\tilde{\mathbf{x}}_\tau$ and $\mathbf{x}_t$. In Figure (2), $1-\alpha$ refers to weight given to the gradient of the regularization function $\mathcal{R}$. So, the lesser the value of α, the more the weight of the intermediate iterates in the objective function for training the score network. The trained score model is then provided to the RK45 ODE Solver (*scipy*) for sampling.

Takeaway. As seen in Figure 2(a), the samples generated do have some structural deformities, indicating that the range of the generators is improved due to our setup. Moreover, note that our IGO scheme is robust with respect to α as shown by the gradual degradation in performance as we put more weight on the gradients provided by the intermediate iterates. Surprisingly, even in this simple setting we find that the Fretchet Inception Distance (FID) [13] and the optimal weight α need not have a monotonic relationship as shown in Figure 2(c).

4.2 IGO on Panel Data

4.2.1 Setup Details

One of the applications of our proposed Intermediate Generator Optimization is modelling the trajectory of Differential Equation, which describes the dynamic process. We use the setup in [22], that is, we consider that dynamic process is modelled by changes in the latent space z as $\dot{z}(t) = D(z,t)\mathbf{w}$, where $D(z,t)$ is a neural network and $\mathbf{w}$ is a corresponding mixed effect (random projection describing flexibility (stochasticity) of dynamics). We propose to extend $D(z,t)$ with our IGO, as $D(z_t, z_\tau)$ using a neural network f, see Figure 3 for details on loss computation with z_τ.

4.2.2 Experimental results

We evaluate our approach on two temporal vision dataset, representing the dynamics of Panel Data. Namely, we apply our proposed method to the variation of MuJoCo Hopper and Rotation MNIST.

MuJoCo Hopper The dataset represents the process with simple Newtonian physics, which is defined by the initial position, velocity and number of steps. To generate data we randomly chose an initial position and sample velocity vector uniformly from $[-2, 2]$. We evaluate our model on interpolation (3 steps) and extrapolation (3 steps) and provide visualization of one of the experiment in Figure 4. The MSE for interpolation was 0.0289, while for Extrapolation was 0.2801. In comparison ODE2VAE [22]'s interpolation MSE is at 0.0648.

Rotation MNIST We evaluate our approach on a more complicated version of rotating MNIST dataset. We construct a dataset by rotating the images of different handwritten digits, and recon-

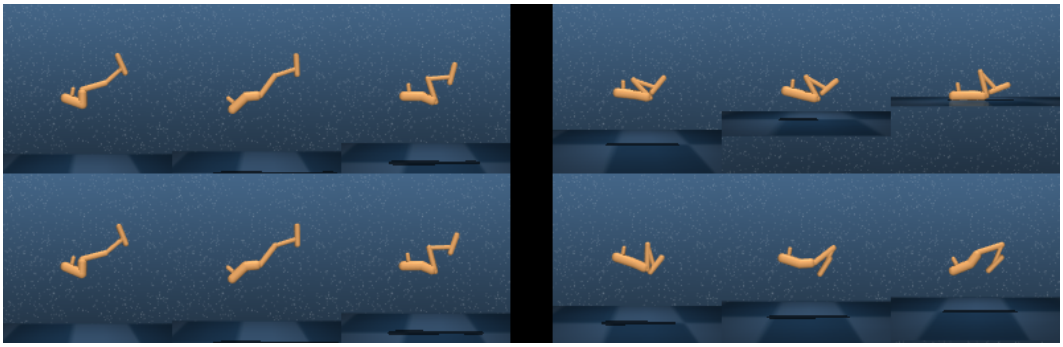
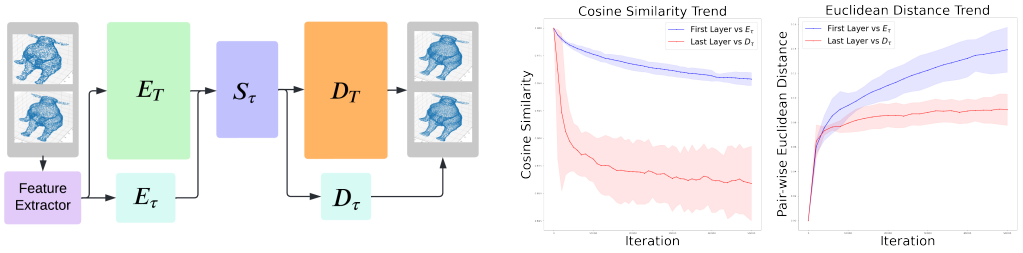


Figure 4: One of the Hopper samples: **Top** is the true data, **Bottom** is the prediction. Before the black line is observed data, after it are the extrapolated samples previously not seen by the model.

structing trajectory of the rotation in interpolation and extrapolation setups. For a sampled digit we randomly choose an angle from the range $[-\pi/4, \pi/4]$ and apply it at all time steps. In addition, to make our evaluation more robust and close to a practical scenario, we spread out the initial points of the digit, by randomly rotating a digit by angles from $-\pi/2$ to $\pi/2$. The generated 10K samples of different rotating digits for 20 time steps were split it in two equal sets: interpolation and extrapolation. Similar to previous experiment MuJoCo Hopper, we evaluate our model on interpolation (10 steps) and extrapolation (10 steps). In this case the MSE for interpolation and extrapolation were 0.0082 and 0.1545 respectively. In comparison NODE [22] is 0.0074 and 0.1661 respectively. Whereas MEODE [22] is 0.0057 and 0.1641 respectively.

Takeaway. Clearly, we can see that by introducing IGO in the model, we get a good generation ability even for extrapolation in the future time steps.



(a) (b) (c)

Figure 5: **a)** : IGO Model for 3D point cloud denoising. **b)** : Using Cosine Similarity and Pairwise Euclidean distance to compare the weights of E_t against E_τ, and D_t against D_τ

4.3 IGO in 3D Processing

4.3.1 Setup Details

We use a standard deep learning based point cloud denoiser that comprises of a Feature Extraction Unit and a Score Estimation Unit. The feature extractor is tasked to learn local as well as non-local features for each point in the noisy point cloud data, provided as the input. Here, the score estimator provides point-wise scores, using which the gradient of the log-probability function can be computed. We adopt the same feature extraction unit as well as the Score Estimation Unit used in [19]. Here we introduce intermediate iterates to test whether we can generate different versions of the denoised samples while using pretrained score based model provided by [19], see Figure 5 - a).

4.3.2 Experimental results

In the beginning of the denoising step, we have with us the given noisy input, $\tilde{\mathbf{x}}_t$ and its intermediate iterate $\tilde{\mathbf{x}}_\tau$. Our architecture is modelled using a 3 layer MLP as in [19], and the newly added intermediate layers, E_τ and D_τ. The newly added layers replicate the already present first and the

# Points	10K (Sparse)				50K (Dense)			
Noise	1%		2%		1%		2%	
Model	CD	P2M	CD	P2M	CD	P2M	CD	P2M
Bilateral [8]	3.646	1.342	5.007	2.018	0.877	0.234	2.376	1.389
Jet [4]	2.712	0.613	4.155	1.347	0.851	0.207	2.432	1.403
MRPCA [21]	2.972	0.922	3.728	1.117	**0.669**	**0.099**	2.008	1.033
GLR [37]	2.959	1.052	3.773	1.306	0.696	0.161	1.587	0.830
PCNet [27]	3.515	1.148	7.467	3.965	1.049	0.346	1.447	0.608
DMR [18]	4.482	1.722	4.982	2.115	1.162	0.469	1.566	0.800
Score-Based PCD [19]	**2.521**	**0.463**	**3.686**	**1.074**	0.716	0.150	**1.288**	**0.566**
Ours ($\alpha = 0.8$)	2.710	0.593	4.292	1.524	0.954	0.320	2.456	1.517

Table 1: Comparison against other denoising algorithms. The CD as well as the P2M scores are multiplied by $1e{+}4$.

last layer perceptrons but are only dedicated to the intermediate layers. We initialize the weights of E_τ and D_τ to be half of the weights of the pre-trained layers. The training was done using a convex combination of loss, defined using the intermediate and the final iterate, $\tilde{\mathbf{x}}_t$ and $\tilde{\mathbf{x}}_\tau$. We use α to denote this convex combination hyperparameter. The lesser the value of α, the more the weight of the intermediate iterates in the objective function for training the score network. The network was tested using the PU-Net test-set provided by [19]. The results in Table 1 compares the Chamfer Distance (CD) [7] and Point-to-Mesh Distance (P2M) [28] of the denoised point clouds from D_τ, for $\alpha{=}0.8$. Additional results for higher noise levels can be found in the supplementary.

4.3.3 Finding new Generators using X_τ

Here we validate our hypothesis that we can train diverse generators using intermediate iterate. In order to quantify the spread of our overall model, we use the cosine similarity metric suggested for generalization purposes [16]. Here, we compute the cosine similarities between the weights of E_T and E_τ (also, D_T and D_τ). Recall, D_T is the usual decoder which is the last layer and D_τ is its corresponding intermediate decoder. We also calculate the pair-wise Euclidean distance between the respective weights, see Figure 5 - b) , c).

Takeaway. Firstly, in low noise settings we can train point cloud denoisers that can compete with the State of the Art models. Secondly, we observed that as we fine tune our model the cosine similarity between D_T and D_τ decreases by 10% and the Euclidean between E_T and E_τ increases by 15%, in $50k$ iterations —two dissimilar decoders that perform well on the training dataset.

5 Discussion and Conclusions

The main motivation of the current submission is the development of a new deep learning based generators, beyond density estimation. In this paper, we have identified an interesting connection between concepts that are primarily used in discrete dynamical systems and learning problems with a compositional structure. Our formulation seeks to increase the range of generators and readily extends to problems that are naturally formulated as a distributional optimization problem. Our method of coupling the layers of a DGM with noise levels, through the regularization loss, can be easily used as a plug and play support in cases where we have access to the intermediate iterates.

From the theoretical perspective, our results indicate that it is indeed possible to be more sample efficient while solving inverse problems using our IGO construction by slightly modifying standard assumptions made on DGMs. We can implement our regularizer on any end-to-end differentiable DGM with minor code modifications as shown in two 2D image and one 3D point cloud settings. Empirically, we present a relationship between α and metrics such as FID showing that the landscape of the parameters has interesting properties, and can be exploited for efficient optimization purposes. We believe that abstract ideas from dynamical systems are very much relevant in for decision making in vision settings as more and more pretrained models are deployed on form factor and/or edge devices to make on-the-fly decisions. Our results provides us an affirmative evidence that ideas from dynamical systems will be of at most importance when each such decision has nontrivial consequences (say due to presence of adversarial noise).

References

[1] J. Bao and Q. Yang. Period of the discrete arnold cat map and general cat map. *Nonlinear Dynamics*, 70(2):1365–1375, 2012.

[2] A. Bora, A. Jalal, E. Price, and A. G. Dimakis. Compressed sensing using generative models. In *International Conference on Machine Learning*, pages 537–546. PMLR, 2017.

[3] E. J. Candes and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE transactions on information theory*, 52(12):5406–5425, 2006.

[4] F. Cazals and M. Pouget. Estimating Differential Quantities using Polynomial fitting of Osculating Jets. *Computer Aided Geometric Design*, 22(2):121–146, 2005. URL `https://hal.inria.fr/inria-00097582`.

[5] G. Daras, J. Dean, A. Jalal, and A. Dimakis. Intermediate layer optimization for inverse problems using deep generative models. In *International Conference on Machine Learning (ICML)*, 2021.

[6] U. Demir and G. Unal. Patch-based image inpainting with generative adversarial networks. *arXiv preprint arXiv:1803.07422*, 2018.

[7] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image, 2016.

[8] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. In *ACM SIGGRAPH 2003 Papers, SIGGRAPH '03*, ACM SIGGRAPH 2003 Papers, SIGGRAPH '03, pages 950–953, 2003. ISBN 1581137095. doi: 10.1145/1201775.882368. null ; Conference date: 27-07-2003 Through 31-07-2003.

[9] Z. Goldfeld and Y. Polyanskiy. The information bottleneck problem and its applications in machine learning. *IEEE Journal on Selected Areas in Information Theory*, 1(1):19–38, 2020.

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples, 2014. URL `https://arxiv.org/abs/1412.6572`.

[11] P. Hand, O. Leong, and V. Voroninski. Phase retrieval under a generative prior. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 9154–9164, 2018.

[12] V. Harutyunyan, A. Aivazyan, E. Weber, Y. Kim, Y. Park, and S. Subramanya. High-resolution x-ray diffraction strain-stress analysis of gan/sapphire heterostructures. *Journal of Physics D: Applied Physics*, 34(10A):A35, 2001.

[13] 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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf`.

[14] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

[15] R. Huang, B. Xu, D. Schuurmans, and C. Szepesvari. Learning with a strong adversary, 2015. URL `https://arxiv.org/abs/1511.03034`.

[16] G. Jin, X. Yi, L. Zhang, L. Zhang, S. Schewe, and X. Huang. How does weight correlation affect the generalisation ability of deep neural networks. *arXiv preprint arXiv:2010.05983*, 2020.

[17] D. Kelly. Rough path recursions and diffusion approximations. *The Annals of Applied Probability*, 26(1):425–461, 2016.

[18] S. Luo and W. Hu. Differentiable manifold reconstruction for point cloud denoising. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 1330–1338, 2020.

[19] S. Luo and W. Hu. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4583–4592, October 2021.

[20] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

[21] E. Mattei and A. Castrodad. Point cloud denoising via moving rpca. *Computer Graphics Forum*, 36, 2017.

[22] J. Nazarovs, R. Chakraborty, S. Tasneeyapant, S. N. Ravi, and V. Singh. A variational approximation for analyzing the dynamics of panel data. In *Uncertainty in artificial intelligence: proceedings of the... conference. Conference on Uncertainty in Artificial Intelligence*, volume 2021. NIH Public Access, 2021.

[23] A. Neuenkirch, M. Szolgyenyi, and L. Szpruch. An adaptive euler–maruyama scheme for stochastic differential equations with discontinuous drift and its convergence analysis. *SIAM Journal on Numerical Analysis*, 57(1):378–403, 2019.

[24] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

[25] X. Pan, X. Zhan, B. Dai, D. Lin, C. C. Loy, and P. Luo. Exploiting deep generative prior for versatile image restoration and manipulation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[26] O. Poursaeed, T. Jiang, H. Yang, S. Belongie, and S.-N. Lim. Robustness and generalization via generative adversarial training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15711–15720, 2021.

[27] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov. Pointcleannet: Learning to denoise and remove outliers from dense point clouds. In *Computer Graphics Forum*, volume 39, pages 185–203. Wiley Online Library, 2020.

[28] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. Accelerating 3d deep learning with pytorch3d, 2020.

[29] A. Razavi, A. van den Oord, and O. Vinyals. Generating diverse high-fidelity images with vq-vae-2. In *Advances in neural information processing systems*, pages 14866–14876, 2019.

[30] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[31] Y. Rubanova, R. T. Chen, and D. Duvenaud. Latent odes for irregularly-sampled time series. *arXiv preprint arXiv:1907.03907*, 2019.

[32] 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. URL https://openreview.net/forum?id=PxTIG12RRHS.

[33] J. A. Tropp. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

[34] R. Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[35] L. Wasserman. *All of statistics : a concise course in statistical inference*. Springer, New York, 2010. ISBN 9781441923226 1441923225. URL http://www.amazon.de/All-Statistics-Statistical-Inference-Springer/dp/1441923225/ref=sr_1_2?ie=UTF8&qid=1356099149&sr=8-2.

[36] H. Wu, S. Zheng, J. Zhang, and K. Huang. Gp-gan: Towards realistic high-resolution image blending. In *Proceedings of the 27th ACM international conference on multimedia*, pages 2487–2495, 2019.

[37] J. Zeng, G. Cheung, M. Ng, J. Pang, and C. Yang. 3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model. *IEEE Transactions on Image Processing*, 29:3474–3489, 2020. doi: 10.1109/TIP.2019.2961429.

[38] M. Zhang, S. Jiang, Z. Cui, R. Garnett, and Y. Chen. D-vae: A variational autoencoder for directed acyclic graphs. *arXiv preprint arXiv:1904.11088*, 2019.

[39] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.