Manifold Learning Benefits GANs

Yao Ni[*,†], Piotr Koniusz[*,§,†], Richard Hartley[†,♦], Richard Nock[♦,♣,†]
[†]The Australian National University [§]Data61/CSIRO [♦]Google Research

firstname.lastname@anu.edu.au

Abstract

In this paper[1], we improve Generative Adversarial Networks by incorporating a manifold learning step into the discriminator. We consider locality-constrained linear and subspace-based manifolds[2], and locality-constrained non-linear manifolds. In our design, the manifold learning and coding steps are intertwined with layers of the discriminator, with the goal of attracting intermediate feature representations onto manifolds. We adaptively balance the discrepancy between feature representations and their manifold view, which is a trade-off between denoising on the manifold and refining the manifold. We find that locality-constrained non-linear manifolds outperform linear manifolds due to their non-uniform density and smoothness. We also substantially outperform state-of-the-art baselines.

1. Introduction

Generative Adversarial Networks (GANs) [13] are powerful models for image generation [5, 19, 22], sound generation [11], image stylization [32] and destylization [46–49], super-resolution [62], feature generation [60, 65], *etc*. The original GAN learns to generate images [5, 19, 21, 22, 56] by performing the following min-max game:

$$\min_{\boldsymbol{\theta}_G} \max_{\boldsymbol{\theta}_D} \mathcal{J}(D_{|\boldsymbol{\theta}_D}, G_{|\boldsymbol{\theta}_G}), \tag{1}$$

where $\mathcal{J}(\cdot) = \mathbb{E}_{\boldsymbol{x}\sim p_x(\boldsymbol{x})}\log(D(\boldsymbol{x};\boldsymbol{\theta}_D)) + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}\log(1 - D(G(\boldsymbol{z})))$. Eq. (1) updates parameters $\boldsymbol{\theta}_D$ of discriminator $D(\boldsymbol{x};\boldsymbol{\theta}_D)$ to discriminate between samples from the data distribution $p_x(\boldsymbol{x})$ and generative distributions $p_g(G)$. Simultaneously, parameters $\boldsymbol{\theta}_G$ of generator $G(\boldsymbol{z};\boldsymbol{\theta}_G)$ are updated to fool the discriminator D. Thus, the noise distribution $p_z(\boldsymbol{z})$ becomes mapped to $p_x(\boldsymbol{x})$ via generator G.

However, GANs typically suffer from three problems: **1)** the training instability [25], **2)** the so-called mode collapse [44], and **3)** overfitting of the discriminator [58].

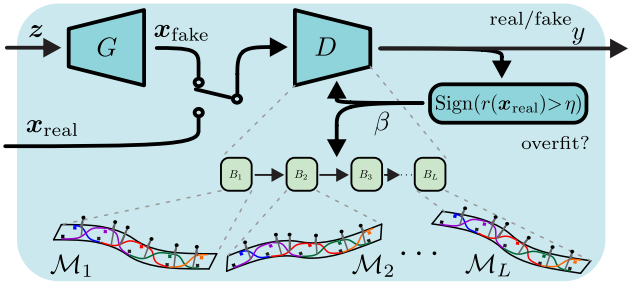


Figure 1. Our GAN pipeline. We equip the discriminator with residual blocks $B_1, \cdots, B_L$, each containing standard CNN operations, *e.g*. convolutions, ReLU, downsampling, residual link *etc*., and the manifold learner $\mathcal{M}_l$. Metaparameter β controls the degree of mixing block conv. features with their view recovered from the manifold. The 'overfit?' detector increases β when overfitting to $\boldsymbol{x}_{\text{real}}$ is suspected, which boosts the impact of manifold learners.

The training instability is an imbalanced competition of the generator and the discriminator due to non-overlapping support between the model distribution and the data distribution [6, 19, 25], leading to poor quality of generated data. Mode collapse has to do with sharply rising gradient around undesirable local equilibria [25, 44], resulting in generation of the same image. Finally, discriminator overfitting leads to excessive memorization and poor generalization.

Indeed, with an excessive number of parameters, the discriminator may memorize the training data instead of learning a meaningful distribution, leading to a high real/fake classification accuracy on the training dataset and a low accuracy on the validation split [5, 20, 66]. Webster *et al*. [58] argue such a phenomenon mainly affects the discriminator and is undetectable in the generator, with the exception of hybrid adversarial and non-adversarial methods [4] which impose the so-called consistency loss on a generator.

We also observed discriminator overfitting in baseline models *e.g*., doubling the number of parameters of discriminator resulted in training and validation FID scores of baseline GANs diverging at some intermediate training stage.

Thus, to reduce overfitting of the discriminator, we propose a data-driven feature manifold-learning step and intertwine it with layers of the discriminator. In this way, the discriminator learns the feature manifold at different levels of object abstraction, from fine to coarse, which limits

[*]Equal contribution. [♣] Brain team (richardnock@google.com).

[1]Code: `https://github.com/MaxwellYaoNi/LCSAGAN`.

[2]The coding spaces considered in this paper are loosely termed manifolds. In most cases they are not manifolds in the strict mathematical sense, but rather topological spaces such as varieties, or simplicial complexes. The word will be used only in an informal sense.

the complexity of parameter space and separates the signal from noise as both generated and real data are expressed on a common manifold. As a result, the generator diversifies the generated patterns according to their view on the manifold, on which the discriminator operates. The min-max game operates on a gradually learnt manifold (see Fig. 1).

Our contributions are threefold:

i. We intertwine locality-constrained and subspace-based feature encoding and dictionary learning steps [29, 34] with blocks of the GAN discriminator to exploit manifold learning in an end-to-end scenario.

ii. We employ a balancing term to help blocks of the discriminator learn the data-driven manifold from the encoder intertwined with them, while permitting some degree of freedom in the vicinity of that manifold (§2).

iii. We show that locality-constrained soft assignment coding (the best coder in our experiments) acts as a locally flexible denoiser [1] due to its Lipschitz continuity which we control to vary its operating mode between the ordinary k-means quantization and locality-constrained linear coding. This setting admits quantization of some feature space parts while approximately preserving linearity of other feature space parts (§5).

For contribution (i), we investigate Sparse Coding (SC) [31, 61], Non-negative Sparse Coding (SC_+) [16], Orthogonal Matching Pursuit (OMP) [8, 42], Locality-constrained Linear Coding (LLC) [55], Soft Assignment (SA) [3, 54], and Locality-constrained Soft Assignment (LCSA) [26–29, 34], and Hard Assignment (HA) [7, 51]. We provide formulations and discussion on properties of each coder in §4.

2. Problem Formulation

Figure 1 shows our pipeline (we skip conditional cues for brevity). We build on BigGAN [5], OmniGAN [69], MSG-StyleGAN [18], StyleGAN2 [22] but we equip the discriminator with the manifold learner which is metacontrolled to reduce overfitting. In §C of the supplementary material, we also study the combination of our method with DA [66], ADA [20] and LeCamGAN [53] in limited data scenario.

The discriminator $D(\boldsymbol{x};\boldsymbol{\theta}_D)$ of the GAN in Eq. (1) classifies input images as real or fake. Many architectures exist in the literature *e.g.*, GAN [13] uses the discriminator based on a convolutional network, whereas recent architectures *e.g.*, BigGAN [5], OmniGAN [69], MSG-StyleGAN [18] and StyleGAN2 [22] use residual discriminators with L residual blocks *e.g.*, see BigGAN [5] (their Fig. 16).

Let[3] $\boldsymbol{f} : \mathbb{R}^{d\times N} \times \mathbb{R}^{|\boldsymbol{\theta}_B|} \rightarrow \mathbb{R}^{d'\times N'}$ (where $|\boldsymbol{\theta}_B|$ is the size of the set of parameters $\boldsymbol{\theta}_B$) be a function realized by a single discriminator block with parameters $\boldsymbol{\theta}_B$, where d and d' are the number of input/output channels, $N = WH$ and $N' = W'H'$ are the number of input/output spatial locations in feature maps of the block. Often, N' may equal N.

[3] Our notations are explained in §A of the supplementary material.

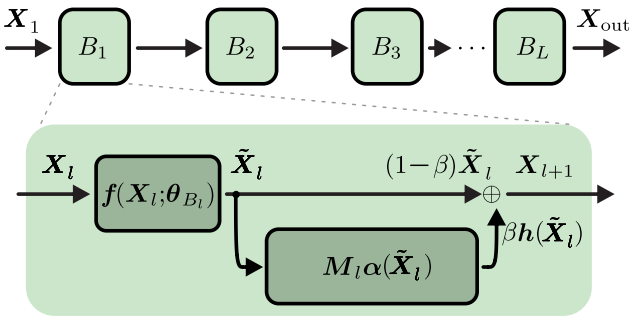


Figure 2. Blocks $B_1, \cdots, B_L$ of our discriminator contain a standard block denoted by $\boldsymbol{f}$ intertwined with the manifold learner. Metaparameter β controls the mixing balance between $\boldsymbol{f}$ and $\boldsymbol{h}$.

We introduce an encoding function $\boldsymbol{h}: \mathbb{R}^{d'\times N'} \rightarrow \mathbb{R}^{d'\times N'}$ that maps $\mathbb{R}^{d'\times N'}$ into a subset, usually nowhere dense or of small volume in $\mathbb{R}^{d'\times N'}$, which we sometimes refer to as the *feature space*. In the cases we consider this mapping derives from a mapping $\mathbb{R}^{d'} \rightarrow \mathbb{R}^{d'}$ applied independently and equally over the second dimension $\mathbb{R}^{N'}$. The encoding introduces an error, measured by $\|\boldsymbol{h}(\boldsymbol{X}) - \boldsymbol{X}\|_F \leq \epsilon$ where ϵ is called the reconstruction error.

In dictionary-based encoding, the function $\boldsymbol{h}$ relies on a dictionary $\boldsymbol{M} = [\boldsymbol{m}_1, \cdots, \boldsymbol{m}_k] \in \mathbb{R}^{d'\times k}$ containing k column vectors, the so-called dictionary atoms (sometimes called anchors), defining the underlying manifold $\mathcal{M}$, and $k \gg d'$ ensures the dictionary is overcomplete. Then, after solving the optimization problem,

$$(\boldsymbol{\alpha}, \boldsymbol{M}) = \underset{\boldsymbol{\alpha}', \boldsymbol{M}'}{\arg\min} \|\boldsymbol{X} - \boldsymbol{M}'\boldsymbol{\alpha}'\|_F^2 + \kappa\,\Omega(\boldsymbol{\alpha}', \boldsymbol{M}', \boldsymbol{X}), \quad (2)$$

where $\boldsymbol{\alpha} \equiv [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{N'}] \in \mathbb{R}^{k\times N'}$, the function $\boldsymbol{h}$ is defined by $\boldsymbol{h}(\boldsymbol{X}) = \boldsymbol{M}\boldsymbol{\alpha}$. Since $\boldsymbol{\alpha}$ depends on $\boldsymbol{X}$, we shall commonly write it as $\boldsymbol{\alpha}(\boldsymbol{X})$. The mapping $\boldsymbol{h}$ maps $\mathbb{R}^{d'}$ into a subset $\mathcal{M}$ of $\mathbb{R}^{d'}$, that we will call the feature manifold (or simply manifold).

The choice of $\Omega(\boldsymbol{\alpha}', \boldsymbol{M}', \boldsymbol{X})$ realizes some desired constraints via regularization (with $\kappa > 0$) for example, $\Omega(\boldsymbol{\alpha}', \boldsymbol{M}', \boldsymbol{X}) = \|\boldsymbol{\alpha}'\|_1$ encourages sparsity of $\boldsymbol{\alpha}$, while $\Omega(\boldsymbol{\alpha}', \boldsymbol{M}', \boldsymbol{X}) = \sum_n [\|\boldsymbol{x}_n - \boldsymbol{m}_1\|_2^2, \cdots, \|\boldsymbol{x}_n - \boldsymbol{m}_k\|_2^2]^T |\boldsymbol{\alpha}_n|$ encourages locality to express each $\boldsymbol{\alpha}_n$ w.r.t. $\text{Span}(\boldsymbol{m}_1, \cdots, \boldsymbol{m}_{k'})$, where $\boldsymbol{m}_1, \cdots, \boldsymbol{m}_{k'}$ are the k' nearest neighbors of $\boldsymbol{x}_n$.

We intertwine the encoding step with blocks of the discriminator as follows:

$$\boldsymbol{X}_{l+1} = (1-\beta)\tilde{\boldsymbol{X}}_l + \beta\,\boldsymbol{h}_l(\tilde{\boldsymbol{X}}_l) \quad (3)$$

where $\tilde{\boldsymbol{X}}_l = \boldsymbol{f}(\boldsymbol{X}_l; \boldsymbol{\theta}_{B_l})$ and $\boldsymbol{h}_l$ is the encoding function introduced just above, expressed in terms of a dictionary $\boldsymbol{M}_l$, whereas $\{\boldsymbol{\theta}_{B_l}\}_{l=1}^L$ and $\{\boldsymbol{M}_l\}_{l=1}^L$ are parameters of blocks of the discriminator and dictionaries for layers $1, \ldots, L$ respectively. Figure 2 illustrates Eq. (3) applied to blocks $B_1, \cdots, B_L$ of the discriminator.

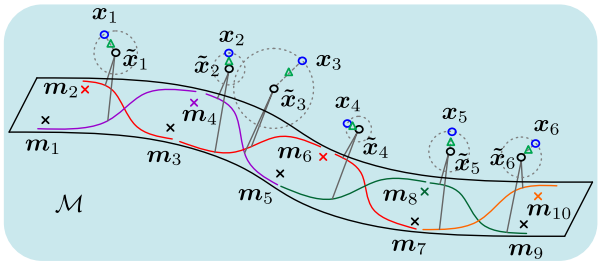


Figure 3. Our manifold setting. Atoms $\boldsymbol{m}_1, \cdots, \boldsymbol{m}_k$ (crosses $\times$) define the geometry of the manifold. Samples $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{N'}$ (blue $\circ$) are projected onto the manifold $\mathcal{M}$ via function $\boldsymbol{\alpha}(\boldsymbol{x})$ and then recovered via $\boldsymbol{h}(\boldsymbol{x})$, which produces recovered samples $\tilde{\boldsymbol{x}}_1, \cdots, \tilde{\boldsymbol{x}}_{N'}$ (black $\circ$). The grey circles indicate L_2 balls imposed by the proximity operator in Eq. (4) while green triangles $\triangle$ are trade-off samples between $\boldsymbol{f}$ and $\boldsymbol{h}$, controlled by β within L_2 balls. We note that radii of L_2 balls are controlled by γ but each ball can be larger or smaller depending whether this benefits the discrimination loss. Thus, where desired, trade-off samples are refined by $\boldsymbol{f}$ w.r.t. $\boldsymbol{h}$ within L_2 balls. Curly lines (sigmoid-like $\backsim$) indicate we use locally the sigmoid non-linearity.

At the same time, we prevent Eq. (3) from becoming a residual link by adding a reconstruction loss to the GAN objective:

$$\mathcal{J}_{\text{prox}} = \frac{\gamma}{L} \sum_{l=1}^{L} \boldsymbol{\epsilon}(\tilde{\boldsymbol{X}}_l; \boldsymbol{M}_l), \quad \text{where}$$
$$\boldsymbol{\epsilon}(\tilde{\boldsymbol{X}}_l; \boldsymbol{M}_l) = \|\tilde{\boldsymbol{X}}_l - \boldsymbol{h}_l(\tilde{\boldsymbol{X}}_l)\|_F^2 \,. \tag{4}$$

Metaparameters (β, γ) control the mixing balance and the proximity between $\boldsymbol{f}$ and $\boldsymbol{h}$.

Meta-adaptation of (β, γ). Discriminator overfitting can be detected with a hypothesis test on the expectation over decisions $r(\boldsymbol{x}_{\text{real}}) = \mathbb{E}[\text{Sign}(D(\boldsymbol{x}_{\text{real}}))]$ w.r.t. samples $\boldsymbol{x}_{\text{real}}$, defined as $\text{Sign}(r(\boldsymbol{x}_{\text{real}}) > \eta) \in \{-1, 0, 1\}$, where $\eta = 0.5$ is the threshold whose violation indicates potential overfitting, as the discriminator becomes increasingly good at distinguishing real datapoints [20]. Thus, to update (β, γ), we apply:

$$\beta_{t+1} = \beta_t + \Delta_\beta \cdot \text{Sign}\left(r(\boldsymbol{x}_{\text{real}}) > \eta\right), \tag{5}$$
$$\gamma_{t+1} = \gamma_0 + \Delta_\gamma \cdot \beta_{t+1}, \tag{6}$$

where $\beta_0 = 0.1$. $\Delta_\beta = 0.001$ ensures a gradual change of β by increasing contributions from $\boldsymbol{h}(\tilde{\boldsymbol{X}}_l)$ in Eq. (3) when overfitting is detected, and increasing contributions from $\tilde{\boldsymbol{X}}_l$ in Eq. (3) when overfitting vanishes. By setting $\gamma_0 = 0.1$, we ensure that the proximity loss in Eq. (4) is always enabled, and $0.01 \leq \Delta_\gamma \leq 3$ controls the strength of proximity.

Discussion. Figure 3 shows our manifold setting, which exploits the interplay between Eq. (3) and Eq. (4). We alternately learn encoding of samples $\boldsymbol{X}_l$ on manifold $\mathcal{M}_l$ and refine dictionary $\boldsymbol{M}_l$. The proximity operator in Eq. (4) encourages samples $\tilde{\boldsymbol{X}}_l$ to stay in the proximity of their recovered view $\boldsymbol{h}(\tilde{\boldsymbol{X}}_l)$ by controlling L_2 balls around $\boldsymbol{h}(\tilde{\boldsymbol{X}}_l)$. Eq. (3) interpolates between $\boldsymbol{h}(\tilde{\boldsymbol{X}}_l)$ and $\tilde{\boldsymbol{X}}_l$ within L_2 balls. We opt for such a design as (i) $\boldsymbol{f}$ may refine $\boldsymbol{h}$ if the discriminator loss spotting real/fake inputs deems it useful, (ii) $\boldsymbol{h}$ is preferred when overfitting is detected, whereas $\boldsymbol{f}$ introduces refined patterns otherwise, (iii) $\boldsymbol{f}$ is encouraged to learn from the piecewise-smooth $\boldsymbol{h}$ (see §5).

3. Related Works

Modern GANs. Recent GANs build on GAN [13] or DCGAN [43] by improving generator or discriminator. A residual model [14] was improved by adding a self-attention block [63]. Progressive GAN [19] uses several levels of layers for increasingly finer image resolution. StyleGAN [21] maps the input to an intermediate latent space and controls the generator through adaptive instance normalization (so-called AdaIN). MSG-GAN passes multi-scale gradients from the discriminator to the generator [18].

Improving GANs. To address training instability, mode collapse and overfitting, researchers study (i) loss and distance formulations, (ii) regularization mechanisms and penalties, and (iii) architectural modifications.

Wasserstein GAN (Arjovsky *et al.* [2]) enjoys a good training stability. It was further improved by Gulrajani *et al.* [14] by penalizing the norm of gradient of the critic. Mean & Covariance GAN [38] matches the generated and real data distributions with first- and second-order statistics. A Maximum Mean Discrepancy GAN [33] matches distributions in the Reproducing Kernel Hilbert Space (RKHS).

Spectral Normalization GAN [36] applies normalization on weights to stabilize the discriminator. Spectral Regularization GAN [35] performs detection of so-called spectral collapse. Disconnected Manifold GAN [23] assumes that natural images lie on a union of disjoint manifolds. Feature Quantization GAN (FQGAN) [67] quantizes features of discriminator into a k-means based dictionary. The Denoising Feature Matching GAN [57] encourages proximity between the output of the generator and a denoising auto-encoder.

Our work differs from the Disconnected Manifold GAN which models an entire image distribution as a union of non-explicit manifolds (a collection of generators). FQGAN imposes quantization on features of the discriminator and Denoising Feature Matching GAN learns a denoising auto-encoder on real images to apply it on generated images.

In contrast, we model coarse-to-fine features extracted from multiple blocks of the discriminator, which capture different semantic levels of abstraction. We encourage these features to lie on explicit locality-constrained non-linear manifolds (each block of our discriminator has its own learner). We adaptively control the mixing levels of features and their views recovered from manifolds, and the smoothness of manifolds to prevent overfitting.

Limiting Overfitting. Augmentations (rotations, clipping) [17, 52, 64, 66] can limit overfitting, however, augmentation

artifacts leak into the generated images [20, 68]. Injecting noises into the discriminator [39,59] via dropout [50] forms an ensemble network, whereas we equip the discriminator with a data-manifold learner whose smoothness we control.

4. Preliminaries

Below, we explain GAN pipelines on which we build, and feature encoding and dictionary learning, our key tools.

4.1. Baseline GANs

BigGAN [5] combines a projection-based loss [37], spectral normalization [36], and self-attention [63]. The projection-based score is a trade-off between a class-wise cosine similarity and a class independent term:

$$s(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{y}^T\boldsymbol{V}D(\boldsymbol{x};\boldsymbol{\theta}_D) + \boldsymbol{f}'(D(\boldsymbol{x};\boldsymbol{\theta}_D);\boldsymbol{\theta}_{D'}), \quad (7)$$

where $\boldsymbol{y} \in \{0,1\}^C$ and $\|\boldsymbol{y}\|_1 = 1$, $\boldsymbol{V} \in \mathbb{R}^{C\times d'}$ is a bilinear compatibility matrix that associates output features $\boldsymbol{X}_{\text{out}}$ from D with the class label. As $\boldsymbol{y}$ is a one-hot vector, $\boldsymbol{V} \equiv [\boldsymbol{v}_1,\cdots,\boldsymbol{v}_C]^T$ contains linear projectors $\boldsymbol{v}_c$, one per class $c \in \{0,\cdots,C-1\}$. Function $\boldsymbol{f}' : \mathbb{R}^{d'} \to 1$ is realized by an FC layer with parameters $\boldsymbol{\theta}_{D'}$. Scores $s(\boldsymbol{x},\boldsymbol{y})$ are passed to a hinge-based loss with two components:

$$\mathcal{J}_{\text{discr}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p^{\text{real}}_{x\times y}(\boldsymbol{x},\boldsymbol{y})} \max(0, 1-s(\boldsymbol{x},\boldsymbol{y}))+$$
$$\mathbb{E}_{(\boldsymbol{z},\boldsymbol{y}_z)\sim p_{z\times y_z}(\boldsymbol{z},\boldsymbol{y}_z)} \max(0, 1+s(G(\boldsymbol{z},\boldsymbol{y}_z),\boldsymbol{y}_z)). \quad (8)$$

Our manifold-based pipeline combines loss $\mathcal{J}_{\text{prox}}$ from Eq. (4) with $\mathcal{J}_{\text{discr}}$ and the original $\mathcal{J}_{\text{gen}}$ from [5].

OmniGAN [69] uses a multi-label softmax loss (where a label vector $\boldsymbol{y} \in \{0,1\}^{C+2}$, $\|\boldsymbol{y}\|_1 = 2$ is a concatenation of the one-hot class label vector and one-hot real/fake vector):

$$\mathcal{J}_{\text{discr}} = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p^{\text{real}}_{x\times y}(\boldsymbol{x},\boldsymbol{y})} s(\boldsymbol{x},\boldsymbol{y})+$$
$$\mathbb{E}_{(\boldsymbol{z},\boldsymbol{y}_z)\sim p_{z\times y_z}(\boldsymbol{z},\boldsymbol{y}_z)} s(G(\boldsymbol{z},\boldsymbol{y}_z),\boldsymbol{y}_z) \quad \text{where} \quad (9)$$
$$s(\boldsymbol{x},\boldsymbol{y}) = \sum_{c=0}^{C+1} \log\left(1+e^{-\text{Sign}(y_c-0.5)\phi_c(\boldsymbol{x})}\right). \quad (10)$$

We note that $\boldsymbol{X}_{\text{out}}$ represents output features from D, $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{f}'(D(\boldsymbol{x};\boldsymbol{\theta}_D);\boldsymbol{\theta}_{D'})$, where function $\boldsymbol{f}' : \mathbb{R}^{d'} \to C+2$ is realized by an FC layer with parameters $\boldsymbol{\theta}_{D'}$.

4.2. Feature Coding and Dictionary Learning on Data-driven Manifolds

Below, we formalize feature encoding and dictionary learning approaches listed in §1. In experiments, we substitute a chosen coding step into function $\boldsymbol{h}(\boldsymbol{x})$ from §2. Moreover, $\boldsymbol{M} \equiv [\boldsymbol{m}_1,\cdots,\boldsymbol{m}_k] \in \mathbb{R}^{d'\times k}$ is a dictionary whose learning step is detailed at the bottom of §4.2, and $\boldsymbol{\alpha}(\boldsymbol{x})$ represents the encoding/mapping on the simplex.

Hard Assignment (HA) [7, 51]. This encoder assigns each $\boldsymbol{x}$ to its nearest $\boldsymbol{m}$ by solving the following optimisation problem:

$$\boldsymbol{\alpha}(\boldsymbol{x}) = \underset{\boldsymbol{\alpha}'\in\{0,1\}^k}{\arg\min} \|\boldsymbol{x}-\boldsymbol{M}\boldsymbol{\alpha}'\|_2^2, \quad \text{s.t. } \|\boldsymbol{\alpha}'\|_1 = 1. \quad (11)$$

If $\boldsymbol{M}$ is formed by k-means clustering, HA becomes an equivalent of the quantizer from FQGAN [67].

Sparse Coding (SC) [31,61] & Non-negative Sparse Coding (SC$_+$) [16]. SC encodes $\boldsymbol{x}$ as a sparse linear combination of atoms $\boldsymbol{M}$ by optimising the following objective:

$$\boldsymbol{\alpha}(\boldsymbol{x}) = \underset{\boldsymbol{\alpha}'}{\arg\min} \|\boldsymbol{x}-\boldsymbol{M}\boldsymbol{\alpha}'\|_2^2 + \kappa\|\boldsymbol{\alpha}'\|_1, \quad (12)$$

whereas SC$_+$ additionally imposes a constraint that $\boldsymbol{\alpha}' \geq 0$. Both SC and SC$_+$ encode $\boldsymbol{x}$ on a subset of $\boldsymbol{M}$ of size controlled by the sparsity term.

Orthogonal Matching Pursuit (OMP) [8, 42]. This encoder expresses $\boldsymbol{x}$ as a sparse linear combination of atoms $\boldsymbol{M}$ by optimising the following objective:

$$\boldsymbol{\alpha}(\boldsymbol{x}) = \underset{\boldsymbol{\alpha}'}{\arg\min} \|\boldsymbol{x}-\boldsymbol{M}\boldsymbol{\alpha}'\|_2^2, \quad \text{s.t. } \|\boldsymbol{\alpha}'\|_0 \leq \tau, \quad (13)$$

where the pseudo-norm $\|\boldsymbol{\alpha}'\|_0$ ensures the count of non-zero coefficients of $\boldsymbol{\alpha}'$ is at most τ. Unlike SC and SC$_+$, $\|\boldsymbol{\alpha}'\|_0$, the penalty enforces a strict limit on the number of non-zero elements in $\boldsymbol{\alpha}'$, but the problem itself is NP-hard.

Approximate Locality-constrained Linear Coding (LLC) [55]. LLC expresses $\boldsymbol{x}$ as a linear combination of k' nearest neighbor atoms of $\boldsymbol{x}$ selected from $\boldsymbol{M}$, forming subspaces of size k' on a piecewise-linear manifold:

$$\boldsymbol{\alpha}(\boldsymbol{x}) = \underset{\boldsymbol{\alpha}'}{\arg\min} \|\boldsymbol{x}-\boldsymbol{M}\boldsymbol{\alpha}'\|_2^2, \quad \text{s.t. } \boldsymbol{1}^T\boldsymbol{\alpha}' = 1, \quad (14)$$

and $\boldsymbol{\alpha}'$ is further constrained by $\alpha'_i = 0$ unless $\boldsymbol{m}_i$ is one of the k' closest neighbors of $\boldsymbol{x}$.

Soft Assignment (SA) [3,54] & Locality-constrained Soft Assignment (LCSA) [26, 29, 34]. SA expresses $\boldsymbol{x}$ as the membership probability (concept known from GMM [3]) of $\boldsymbol{x}$ belonging to each $\boldsymbol{m}_i$ in $\boldsymbol{M}$ under equal mixing probability and equal variance σ of GMM. SA is given as:

$$\boldsymbol{\alpha}(\boldsymbol{x};\boldsymbol{M},\sigma) = S_\sigma(\|\boldsymbol{x}-\boldsymbol{m}_1\|_2,\cdots,\|\boldsymbol{x}-\boldsymbol{m}_k\|_2), \quad (15)$$

where S_σ is the softmax function $S_\sigma : \mathbb{R}^k \to \Delta^{k-1}$, where Δ^{k-1} is the probability simplex and:

$$S_\sigma(d_1,\ldots,d_k)_j = \frac{\exp(-d_j^2/2\sigma^2)}{\sum_i \exp(-d_i^2/2\sigma^2)}. \quad (16)$$

This model yields largest values of α'_i for atoms $\boldsymbol{m}_i$ in $\boldsymbol{M}$ that are close Euclidean neighbors of $\boldsymbol{x}$. However, $\alpha_i(\boldsymbol{x}) > 0$ even for $\boldsymbol{m}_i$ that are far from $\boldsymbol{x}$. For this reason, SA is not strictly locality-constrained.

LCSA differs from SA by setting $\alpha_i(\boldsymbol{x}) = 0$ unless $\boldsymbol{m}_i$ is among the k' nearest-neighbor atoms for $\boldsymbol{x}$. The denominator of Eq. (16) performs normalization, that is the summation runs over the k' nearest neighbors. Thus, LCSA maps $\mathbb{R}^d$ onto a set of probability simplices $\Delta^{k'-1}$. As LCSA was the best in our experiments, we analyze it in §5.

Dictionary Learning (DL). For the above coders, we employ a class-agnostic dictionary learning objective which

follows Eq. (2). Let some $\boldsymbol{\alpha}(\boldsymbol{X}) \equiv [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{N'}]$, then:

$$\boldsymbol{M} = \arg\min_{\boldsymbol{M}'} \|\boldsymbol{X} - \boldsymbol{M}'\boldsymbol{\alpha}\|_F^2, \tag{17}$$

where $\boldsymbol{M}'$ can be constrained to contain atoms $\|\boldsymbol{m}'_i\|_2 \leq 1$ if codes $\boldsymbol{\alpha}$ have non-restricted L_2 norm *e.g.*, for OMP.

Inverse of $\boldsymbol{\alpha}$. To reproject $\boldsymbol{\alpha}(\boldsymbol{X})$ from the manifold $\mathcal{M}$ into the Euclidean space, we simply compute $\tilde{\boldsymbol{X}} = \boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{X})$.

Implementation Remarks. Coding methods, dictionary learning, and their implementations are detailed in §J of the supplementary material. For dictionary learning, we detach $\boldsymbol{X}$ and $\boldsymbol{\alpha}$, and run 1 iteration of gradient descent per mini-batch w.r.t. each $\boldsymbol{M}$ (no big gain for ≥ 2 iterations). For SC and SC$_+$, we detach $\boldsymbol{X}$ and all $\boldsymbol{M}$, and let 5 iterations of gradient descent (no gain for ≥ 6 iterations). LLC has a closed-form solver [55]. Our efficient OMP solves the system of linear equations (no matrix inversion). SA/LCSA enjoy a fast closed-form recipe. LLC and LCSA use the partial sort algorithm for selecting k' nearest neighbors. We detach $\boldsymbol{M}\boldsymbol{\alpha}$ to compute the proximity loss in Eq. (4).

5. Theoretical Analysis of LCSA

As LCSA is the best encoder in our experiments, we focus on its theoretical properties below. All proofs for theories listed below are in §L of the supplementary material.

In this section, we use the following notation. Suppose a dictionary of atoms $\boldsymbol{M} = [\boldsymbol{m}_i]$ in $\mathbb{R}^d$ is given, and $2^{\boldsymbol{M}}_{k'}$ denotes the set of all subsets of size k' of $\boldsymbol{M}$. We define by $\mathrm{NN}_{k'} : \mathbb{R}^d \to 2^{\boldsymbol{M}}_{k'}$ to be the set-valued function that takes a point $\boldsymbol{x}$ to its set of k' nearest neighbor atoms. A maximal subset of $\mathbb{R}^d$ on which $\mathrm{NN}_{k'}(\boldsymbol{x})$ is constant is called a Voronoi cell. For a given set U in $2^{\boldsymbol{M}}_{k'}$, then, a Voronoi cell $(\mathrm{NN}_{k'})^{-1}(U)$ is a subset of $\mathbb{R}^d$ consisting of the all the points for which U is the set of k' nearest neighbors. The collection of all Voronoi cells constitutes a decomposition of $\mathbb{R}^d$ into disjoint polyhedral regions.

In the case where the set of k' nearest elements of $\boldsymbol{M}$ is not unique, we leave the set $\mathrm{NN}_{k'}(x)$ undefined. Thus, the Voronoi cells are disjoint open polyhedral regions such that $\mathrm{NN}_{k'}(\boldsymbol{x})$ is constant on each cell. The complement in $\mathbb{R}^d$ of the set of Voronoi cells is a subset of a finite set of hyperplanes in $\mathbb{R}^d$.

SA and LCSA encoding. Given a dictionary $\boldsymbol{M} = [\boldsymbol{m}_i]$ in $\mathbb{R}^d$ consisting of k' elements, with $k' \leq d+1$, we consider the function $\boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x})$ where $\boldsymbol{\alpha}(\boldsymbol{x}) = S_\sigma(\boldsymbol{x})$ is the softmax mapping Eq. (15). (The dictionary may be the dictionary of k' nearest neighbors of some point $\boldsymbol{x}$.)

Proposition 1 *If $\sigma > 0$, the mapping $\boldsymbol{x} \mapsto \boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x})$ is a smooth fibration from $\mathbb{R}^d$* **onto** *the interior of the simplex Δ with vertices $\boldsymbol{m}_i$. The fibre of this mapping is equal to the linear subspace of $\mathbb{R}^d$ normal to the affine space spanned by the $\boldsymbol{m}_i$.*

A fibre is the set of points that map to the same point in $\Delta^{k'-1}$ under this mapping.

Proposition 2 *The following properties of LCSA hold true:*

1. *If $\sigma \to 0$ or $k' = 1$, the $\boldsymbol{\alpha}$ codes converge to the HA solution (quantization).*
2. *$\boldsymbol{\alpha}(\boldsymbol{x})$ is an approximately linear coding of $\boldsymbol{x}$ in the proximity of $\boldsymbol{\mu}(\boldsymbol{x}) = \frac{1}{k'}\sum_{\boldsymbol{m}' \in \mathrm{NN}_{k'}(\boldsymbol{x})} \boldsymbol{m}'$.*
3. *For $\boldsymbol{x}$ with nearest neighbor atoms $\{\boldsymbol{m}_i\} = \mathrm{NN}_{k'}(\boldsymbol{x})$ with mean $\boldsymbol{\mu}(\boldsymbol{x})$, and nearest neighbor atom $\mathbf{n}(\boldsymbol{x}) = \mathrm{NN}_1(\boldsymbol{x})$, the reconstruction error satisfies*
   $\|\boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x}) - \boldsymbol{x}\|_2 \leq \max\Big(\|\boldsymbol{x} - \mathbf{n}(\boldsymbol{x})\|_2, \|\boldsymbol{x} - \boldsymbol{\mu}(\boldsymbol{x})\|_2\Big)$.
4. *The reconstruction error varies smoothly on each Voronoi cell. For $\boldsymbol{x}$ and $\boldsymbol{x}'$ in the same Voronoi cell and Δ the simplex with vertices $\mathrm{NN}_{k'}(\boldsymbol{x})$, we have*
   - *Local Lipschitz continuity: if $\sigma > 0$, then*
     $$\|\boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x}) - \boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x}')\| \leq K\|\boldsymbol{x} - \boldsymbol{x}'\|$$
     where $K = D^2/\sigma^2$ and D is diameter of simplex Δ (the maximum distance between its vertices). The Lipschitz condition holds for $\|\cdot\|_1$ and $\|\cdot\|_2$ norms.
   - *The biggest change of the reconstruction error on a Voronoi cell for HA ($\sigma = 0$) is less than or equal to D.*
5. *The LCSA encoding $\boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x})$ is non-continuous at the boundaries of the Voronoi regions.*

Proposition 3 *Our design fulfils the principles of GAN with Denoising Auto-Encoder (DAE) [1] with loss:*

$$\mathcal{L}_{dae} = \frac{1}{N}\sum_{n=1}^{N}\Big(\Big\|\boldsymbol{r}(\boldsymbol{x}_n) - \boldsymbol{x}_n\Big\|_2^2 + \sigma'^2\Big\|\frac{\partial \boldsymbol{r}(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\boldsymbol{x}_n}\Big\|_2^2\Big), \tag{18}$$

where $\boldsymbol{r}(\boldsymbol{x}_n)$ is the reconstruction of $\boldsymbol{x}_n$ akin to our $\boldsymbol{h}(\boldsymbol{x}_n)$ in Eq. (4). More importantly, σ'^2 specifies the noise variance. Specifically, we note the following:

6. *Not surprisingly, the proximity loss in Eq. (4) fulfils somewhat similar role to $\|\boldsymbol{r}(\boldsymbol{x}_n) - \boldsymbol{x}_n\|_2^2$ in Eq. (18).*
7. *LCSA implicitly fulfils the denoising role (apart from locality-constrained non-linear coding) of $\sigma'^2\Big\|\frac{\partial \boldsymbol{r}(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big\|_F^2$, with σ of LCSA and σ' of DAE related* i.e., *σ'^2 is proportional to our σ^2. To this end, we notice that DAE penalises the Frobenius norm of the Jacobian matrix $\Big\|\frac{\partial \boldsymbol{r}(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big\|_F$ by increasing σ'^2. We penalise the spectral norm of Jacobian matrix $\Big\|\frac{\partial \boldsymbol{M}\boldsymbol{\alpha}(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big\|_2 = K$ via de facto controlling the Lipschitz constant $K = D^2/\sigma^2$.*

Discussion. The blue box below explains how properties of LCSA contribute to discriminator training, and how they let LCSA inherit the best properties of other coding methods.

LCSA as a trade-off between HA, LLC, SA, DAE. Prop. 2 and 3 show that LCSA balances extremes of other coders and inherits their best properties. Prop. 2.1 shows LCSA may act as HA (extreme way to guide $\boldsymbol{f}$). HA is a localized coder with big reconstruction error. In FQGAN, HA stabilized GAN. Prop. 2.2 shows that LCSA may act as LLC (localized linear coder with very low reconstruction error) in the central parts of Voronoi cells (weak way to guide $\boldsymbol{f}$). Prop. 2.3 shows that large dictionary is bad, making LCSA a 'free learner' as $\boldsymbol{f}$ (easy to overfit).

LCSA is locally-adaptive denoiser. Prop. 2.4 shows each Voronoi cell specialises in how much it denoises $\boldsymbol{f}$ based on the Lipschitz constant $K = D^2/\sigma^2$ (diameter D varies in each cell due to the dictionary). Denoising limits high frequencies of signal and its complexity. Prop. 3.7 shows LCSA denoises by DAE-like mechanism.

6. Experiments

We evaluate our method on CIFAR-10 & CIFAR-100 [30] & ImageNet [9] (conditional GAN) and Oxford-102 Flowers [40] and FFHQ [21] (unconditional setting). We show our LCSA harmonizes with BigGAN [5], OmniGAN [69], MSG-StyleGAN [18] and StyleGAN2 [22].

Datasets. CIFAR-10 has $50K$ and $10K$ training and testing images (32×32) from 10 classes, whereas CIFAR-100 has 100 categories. ImageNet has $1.2M$ and $50K$ training and validation images with $1K$ classes. We center-crop and downscale its images to 64×64 and 128×128 pixels. Oxford-102 Flowers contains $8K$ images of 102 fine-grained flower species. We center-crop its images and resize to 256×256. FFHQ dataset provides $70K$ human face images at multiple resolutions (we opt for 256×256). Following [20], we augmented the $70K$ dataset to $140K$ with x-flips.

Evaluation Metrics. We generate $50K$ images per dataset to compute the commonly used Inception Score [45] and Fréchet Inception Distance (FID) [15]. Mean/standard dev. are computed over 5 runs, where both reported. We report tFID, computed between $50K$ generated images and all training images. For CIFAR-10/CIFAR-100/ImageNet, we also compute vFID between $10K$/$10K$/$50K$ generated images and $10K$/$10K$/$50K$ real testing (val. on ImageNet) images. For Oxford-102 Flowers/FFHQ, we calculate FID between $10K$/$50K$ fake images and the entire training set.

6.1. Network Architecture and Hyper-parameters

We build on OmniGAN/BigGAN/StyleGAN2 for CIFAR-10. For CIFAR-100/ImageNet (64×64), we experiment with OmniGAN/BigGAN. For ImageNet (128×128), we build upon OmniGAN given OmniGAN consistently outperforms BigGAN. We employ MSG-StyleGAN as our baseline for Oxford-102 Flowers. For FFHQ, we build upon StyleGAN2 (see §I of the supplementary material).

Model	d'	IS ↑	tFID↓	vFID ↓
BigGAN†	256	9.14	7.05	–
FQGAN†		9.16	6.16	–
OmniGAN†		9.63	5.52	–
CR-GAN		–	–	11.48
ContraGAN		–	–	10.32
ICR-GAN		–	–	9.21
OmniGAN+LCSA		**9.88**±0.02	**4.09**±0.10	**8.16**±0.07
BigGAN	512	9.36	8.16	12.16
FQGAN		9.38	7.65	11.72
OmniGAN		9.70	6.88	10.65
OmniGAN+LCSA		10.02±0.05	3.36±0.06	7.40±0.06
StyleGAN2+ADA		10.14±0.09	2.42±0.04	6.54±0.06
StyleGAN2+ADA+LCSA		**10.18**±0.06	**2.32**±0.05	**6.36**±0.10
OmniGAN+LCSA	1024	**10.21**±0.03	**2.94**±0.02	**6.98**±0.04

Table 1. Results on CIFAR-10. We combine OmniGAN and StyleGAN2+ADA with LCSA. † are results collected from [69].

Model	d'	IS ↑	tFID ↓	vFID ↓
BigGAN†	256	10.89	10.18	–
FQGAN†		10.62	8.23	–
OmniGAN†		13.51	8.14	–
TAC-GAN		9.34	7.22	–
OmniGAN+LCSA		**13.60**±0.11	**6.24**±0.09	**11.02**±0.13
BigGAN	512	11.44	10.16	15.24
FQGAN		11.05	7.76	12.70
OmniGAN		12.78	9.13	13.82
OmniGAN+LCSA		**13.71**±0.03	**5.22**±0.10	**9.98**±0.08
OmniGAN+LCSA	1024	**13.88**±0.12	**4.97**±0.09	**9.72**±0.08

Table 2. Comparison of OmniGAN+LCSA with others on CIFAR-100. † are results collected from [69].

6.2. Results of Image Generation

The generated images for each dataset are given in §K of the supplementary material.

CIFAR-10. Table 1 shows results on OminGAN+LCSA, which outperforms the baseline OmniGAN by 0.25 and 1.43 on the IS and tFID metrics ($d'=256$). With $d'=512$, we outperform OmniGAN by 0.32 and 3.52. We obtain further improvements with $d'=1024$ while baselines struggle to converge. Comparisons of different models with $d'=1024$ are in the §B of the supplementary material.

CIFAR-100. Table 2 shows results on OmniGAN+LCSA against the state of the art. For $d'=256$, our method gains 0.09 and 1.9 on the IS and tFID metrics over the baseline OmniGAN. For $d'=512$, we outperform OmniGAN by 0.93 and 3.91 (IS and tFID). Further gains are achieved for $d'=1024$ while other methods struggle to converge. We include TAC-GAN [12] for comparison.

ImageNet (64×64). Table 3 shows that BiGAN+LCSA outperforms BigGAN by 2.97 and 3.01 (tFID & vFID). OmniGAN+LCSA improves OmniGAN by 6.86 and 2.72 (IS & tFID), which is the state of the art in GANs.

ImageNet (128×128). Table 4 shows OmniGAN+LCSA improves OmniGAN by 23.32 and 2.28 (IS & tFID).

Oxford-102 Flowers. Figure 4(a) shows that MSG-StyleGAN+LCSA improves MSG-StyleGAN by 5.46 on FID. Moreover, Figure 4(b) shows that at around iteration

Model	IS ↑	tFID ↓	vFID ↓
Inst. Sel. GAN [10]	**43.30**	9.07	–
BigGAN	34.50	8.96	8.80
FQGAN	33.14	8.27	8.15
BigGAN+LCSA	33.29	**5.99**	**5.79**
OmniGAN	70.59	7.09	7.66
OmniGAN+LCSA	**77.45**	**4.26**	**4.94**

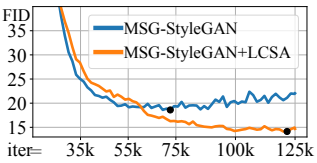
Table 3. Results on ImageNet (64×64). We combine OmniGAN and BigGAN with LCSA. We set $d' = 384$.

Model	IS ↑	tFID ↓	vFID ↓
BigGAN†	104.57	9.19	9.18
OmniGAN†	190.94	8.30	8.93
OmniGAN‡	169.13	7.11	7.30
OmniGAN+LCSA	**192.45**	**4.83**	**5.24**

Table 4. Results on ImageNet (128×128) with $d' = 384$. † stands for quoting from [69], ‡ are results reproduced by us.

Model	Iters	FID ↓
MSG-ProGAN†	$53K$	28.27
MSG-StyleGAN†	$50K$	19.60
MSG-StyleGAN	$125K$	18.59
Ours	$125K$	**13.13**

(a) FID of different models.



(b) Training progress.

Figure 4. Results on Oxford-102 Flowers. (a) FID of different models († are results collected from [18]). (b) FID w.r.t. the iteration number for MSG-StyleGAN and MSG-StyleGAN+LCSA on Oxford-102 Flowers. Black dots indicate the minimum FID.

Data	StyleGAN2	+ADA	+LeCam	+bCR	+LCSA	+LCSA+bCR
70k	5.28	4.30	-	3.79	3.83	**3.47**
140k	3.71	3.81	3.66	3.53	3.32	**3.20**

Table 5. The FID ↓ results on FFHQ (256×256) dataset.

number $75K$, MSG-StyleGAN starts diverging while MSG-StyleGAN+LCSA (Ours) continues to decrease FID.

FFHQ (256×256). Table 5 shows that StyleGAN2 with LCSA improves the FID of StyleGAN2 by 1.55/0.39 on $70K/140K$ dataset. Moreover, our LCSA can also be combined with bCR [68] to improve the performance.

Data-limited Generation on CIFAR-10/100. We provide comprehensive experiments (10% and 20% data) in §C of the supplementary material. We summarize our findings as: 1) the augmentation-based ADA [20] and DA [66] leak augmentation artifacts to the generator, while ADA+LCSA and DA+LCSA alleviate this issue (see Figure 24 in the supplementary material). 2) LCSA harmonizes with ADA, DA and LeCam loss [53]. 3) we achieve the state of the art on this limited data setting.

6.3. Impact of Hyperparameters

Below, we conduct ablations on CIFAR-100 for $d' = 512$.

Nearest neighbors k', σ and k. Figure 5(a) shows results on OmniGAN+LCSA w.r.t. k' on CIFAR-100, which verifies that the locality-constrained manifold can be constructed with $8 \leq k' \leq 32 \ll k$ with a sufficiently small reconstruction error below the worst case (Prop. 2.3). For ImageNet, $k' = 8$ also led to best results. Figure 5(b) verifies that $1 \leq \sigma \leq 1.5$ provides the best trade-off in terms of smoothness (quantization *vs*. linearization) (Prop. 2.1, 2.2 and 2.4). Figure 5(c) verifies the benefit of dictionary overcompleteness $d' \ll k$ as $k = 1024$.

Blocks $l \in \{1, \cdots, L\}$ and LCSA. Figure 5(d) shows the results for various combinations of injection of LCSA into blocks of discriminator. It appears that injecting LCSA into all blocks yields the lowest tFID/vFID, which verifies that constructing multiple manifolds at coarse-to-fine semantic level controls the complexity of the discriminator.

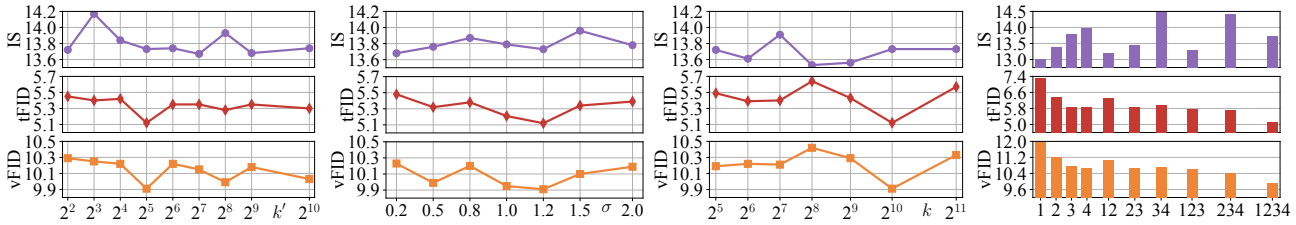
Metaparameter β and impact of η. β in Eq. (5) controls the mixing of conv. features obtained by $\boldsymbol{f}$ and their view $\boldsymbol{h}$ recovered from the manifold, as in Eq. (3). Figure 6(a) shows that in early iterations, blocks of discriminator learn conv. features. Around $25K$ iterations, the model starts oscillating between prevention of overfitting and refining conv. features, as indicated by the green curve that continues to gradually grow for $\eta = 0.5$ (*c.f.* flat curve for $\eta = 0.3$). Figure 6(b) also shows that $\eta = 0.5$ is a universally good threshold.

6.4. Analysis and Ablation studies

We analyse our method on CIFAR-10/100 ($d' = 512$).

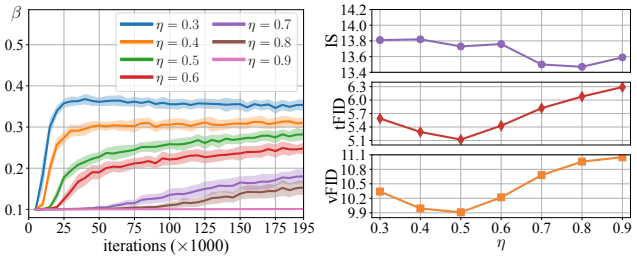
Preventing Discriminator Overfitting. Figure 7 verifies that the discriminator of standard methods yields high real/fake accuracy on training images but low accuracy on testing images. Thus, they overfit to the training set (see the large discrepancy between real and fake images). Thus, standard methods diverge early (see FID) but our method continues learning, as shown in Figure 8 and Figure 4(b).

Impact of Different Components. In Table 6, we ablate (i) dictionary learning (Eq. (17)), (ii) the adaptive mixing input Eq. (3) and (iii) proximity loss Eq. (4). We conduct experiments with settings: (1) ACM: removing our manifold learner and the proximity loss, and changing Eq. (3) to be $\boldsymbol{X}_{l+1} = (1-\beta)\tilde{\boldsymbol{X}}_l$ to adaptively control the magnitude of $\tilde{\boldsymbol{X}}_l$; (2) LCSA($\gamma = 0$): removing the proximity loss but keeping the adaptive mixing input; (3) LCSA($\beta = 0$): removing the adaptive mixing input but keeping the adaptive proximity loss; (4) LCSA(EMA): replacing our dictionary learning with the exponential moving average; (5) LCSA(fixed (β, γ)): fixed meta-controller (β, γ) (different combination of (β and γ) is given in §G of the supplementary material). Table 6 shows that (i) LCSA($\gamma = 0$) is better than ACM and OmniGAN which verifies the benefit of adaptive mixing input. (ii) LCSA($\beta = 0$) is better than OmniGAN which verifies the importance of controlling the complexity of learning. (iii) EMA performs worse than our dictionary learning. (iv) As fixed (β, γ) scores lower thus the adaptive meta-controller is useful. In §H (supplementary material), LCSA yields larger errors at foregrounds and smaller at backgrounds, thus intertwining step (mixing in-



(a) k' (b) σ (c) k (d) $l \in \{1, \cdots, L\}$

Figure 5. Ablation studies on CIFAR-100 w.r.t. k' (nearest neighbors), σ (controls the Lipschitz constant of LCSA), k (dictionary size), and blocks l which use LCSA. We indicate metrics such as the IS ↑, tFID ↓ and vFID ↓.



(a) β w.r.t. iteration. (b) η w.r.t. iteration.

Figure 6. Evolution of metaparameter β and the impact of η which controls the behavior of the detector of overfitting.

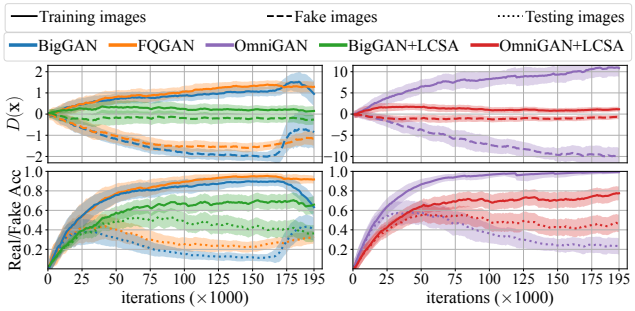


Figure 7. Discriminator predictions on CIFAR-10. We plot the output of discriminator on training images and generated fake images. We test Acc (real/fake accuracy predicted by the discriminator) on the training images and testing images. We use different colors to represent different models. Solid/dash/dot lines indicate training/fake/testing images.

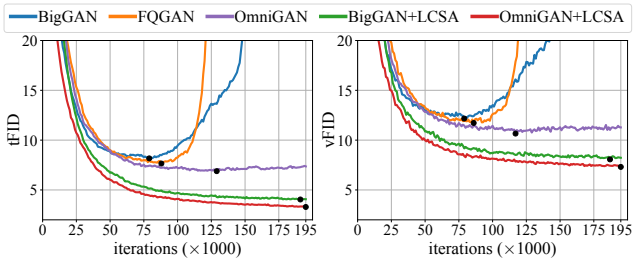


Figure 8. Evolution of tFID and vFID for different models on CIFAR-10. Black dots indicate the minimum FID.

put) helps refine/denoise features before input to next layer.

Different Encoders. Table 6 compares coding methods, each applied to all $l = 1, \cdots, 4$ blocks of discriminator. LCSA is the best performer followed by SA and LLC or SC, which validates that using locality-constrained soft assignment whose continuity is controlled via the Lipschitz constant (Prop. 2.4) inverse-proportional to σ^2 is more robust than locally-linear coding, due to the quantization *vs.* linear reconstruction trade-off and the role of σ^2 in denoising (Prop. 3). We also tried replacing the LCSA coder with Denoising Auto-Encoder (DAE) [1] (the setup is in §F of the supplementary material). LCSA achieves better tFID and vFID than locality-constrained linear coding, subspace learning, and DAE. Figure 25 of the supplementary material shows the variance computed over LCSA codes for indivisual images. While visually mundane regions have low variance (*i.e.*, single code of $\boldsymbol{\alpha}$ encodes it), the visually diverse regions have variance that is somewhat higher for LCSA than other coders, preventing overfitting where it matters.

Method	CIFAR-10			CIFAR-100		
	IS ↑	tFID ↓	vFID ↓	IS ↑	tFID ↓	vFID ↓
OmniGAN	9.70	6.88	10.65	12.78	9.13	13.82
+ACM	9.79	6.03	9.99	12.82	10.61	15.51
+LCSA($\gamma = 0$)	9.80	5.15	8.35	13.42	8.12	12.78
+LCSA($\beta = 0$)	9.73	4.77	8.77	13.64	5.54	10.37
+LCSA(EMA)	9.91	3.83	7.89	13.71	5.63	10.40
+LCSA(fixed(β, γ))	10.01	4.06	8.03	13.64	5.46	10.29
+LCSA	10.09	**3.29**	**7.31**	13.73	**5.12**	**9.91**
+HA	9.88	4.52	8.49	13.68	5.52	10.33
+SC$_+$	10.09	3.53	7.65	13.82	5.49	10.31
+SC	10.14	3.48	7.46	13.68	5.40	10.20
+OMP	10.02	3.83	7.83	13.58	5.52	10.40
+LLC	10.04	3.77	7.75	13.61	5.35	10.14
+SA	10.00	3.46	7.45	13.76	5.31	10.19
+DAE	10.08	3.89	7.91	13.65	5.41	10.22

Table 6. Results for ablation studies on CIFAR-10 & CIFAR-100.

7. Conclusions

We have applied data-manifold learning (LCSA) in the coarse-to-fine manner to conv. features of discriminator to prevent its overfitting by adaptively balancing the trade-off between denoising on the mainfold and refining the mainfold (intertwining and dictionary learning steps), controlling the complexity of learning (the proximity loss). Locality-constrained soft assignment is controlled via the Lipschitz constant, resulting in a trade-off between quantization and linear coding, yielding state-of-the-art results.

Acknowledgements. We thank CSIRO's Machine Learning and Artificial Intelligence Future Science Platform (MLAI FSP) and China Scholarship Council (CSC) for the support.

References

[1] Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *JMLR*, 15(110):3743–3773, 2014. 2, 5, 8, 11, 13

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *ICML*, pages 214–223, 2017. 3

[3] Jeff A Bilmes. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models. Technical report, 1998. 2, 4

[4] Piotr Bojanowski, Armand Joulin, David Lopez-Pas, and Arthur Szlam. Optimizing the latent space of generative networks. In *ICML*, volume 80, pages 600–609, 2018. 1

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019. 1, 2, 4, 6, 14

[6] Casey Chu, Kentaro Minami, and Kenji Fukumizu. Smoothness and stability in gans. In *ICLR*, 2020. 1

[7] Gabriella Csurka, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. Visual categorization with bags of keypoints. In *Workshop on statistical learning in computer vision, ECCV*, 2004. 2, 4

[8] Geoffrey Davis, Stéphane Mallat, and Zhifeng Zhang. Adaptive time-frequency decompositions with matching pursuits. *Optical Engineering*, 33, 1994. 2, 4

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6, 11

[10] Terrance DeVries, Michal Drozdzal, and Graham W Taylor. Instance selection for gans. *NeurIPS*, 2020. 7

[11] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In *ICLR*, 2019. 1

[12] Mingming Gong, Yanwu Xu, Chunyuan Li, Kun Zhang, and Kayhan Batmanghelich. Twin auxilary classifiers gan. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*, pages 1330–1339. 2019. 6

[13] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 1, 2, 3

[14] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *NeurIPS*, pages 5769–5779, 2017. 3

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, volume 30, pages 6626–6637, 2017. 6

[16] Patrik O Hoyer. Non-negative sparse coding. In *Proceedings of the 12th IEEE workshop on neural networks for signal processing*, pages 557–565. IEEE, 2002. 2, 4

[17] Jongheon Jeong and Jinwoo Shin. Training {gan}s with stronger augmentations via contrastive discriminator. In *ICLR*, 2021. 3

[18] Animesh Karnewar and Oliver Wang. Msg-gan: Multi-scale gradients for generative adversarial networks. In *CVPR*, pages 7799–7808, 2020. 2, 3, 6, 7, 14

[19] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of GANs for improved quality, stability, and variation. In *ICLR*, 2018. 1, 3

[20] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. *NeurIPS*, 33, 2020. 1, 2, 3, 4, 6, 7, 11, 12, 14

[21] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, 2019. 1, 3, 6, 22

[22] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, pages 8110–8119, 2020. 1, 2, 6

[23] Mahyar Khayatkhoei, Ahmed Elgammal, and Maneesh Singh. Disconnected manifold learning for generative adversarial networks. In *NeurIPS*, pages 7354–7364, 2018. 3

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014. 14

[25] Naveen Kodali, Jacob Abernethy, James Hays, and Zsolt Kira. On convergence and stability of gans. *arXiv:1705.07215*, 2017. 1

[26] Piotr Koniusz and Krystian Mikolajczyk. Soft Assignment of Visual Words as Linear Coordinate Coding and Optimisation of its Reconstruction Error. *ICIP*, 2011. 2, 4

[27] Piotr Koniusz, Fei Yan, Philippe-Henri Gosselin, and Krystian Mikolajczyk. Higher-order Occurrence Pooling on Mid- and Low-level Features: Visual Concept Detection. Technical report, INRIA, Sept. 2013. 2

[28] Piotr Koniusz, Fei Yan, Philippe-Henri Gosselin, and Krystian Mikolajczyk. Higher-order occurrence pooling for bags-of-words: Visual concept detection. *TPAMI*, 2016. 2

[29] Piotr Koniusz, Fei Yan, and Krystian Mikolajczyk. Comparison of Mid-Level Feature Coding Approaches And Pooling Strategies in Visual Concept Detection. *CVIU*, 2012. 2, 4

[30] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009. 6, 11

[31] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Ng. Efficient sparse coding algorithms. *NeurIPS*, pages 801–808, 2006. 2, 4

[32] Chuan Li and Michael Wand. Precomputed real-time texture synthesis with markovian generative adversarial networks. In *ECCV*, 2016. 1

[33] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabas Poczos. Mmd gan: Towards deeper understanding of moment matching network. In *NeurIPS*, volume 30, pages 2203–2213, 2017. 3

[34] Liu Lingqiao, Lei Wang, and Xinwang Liu. In Defence of Soft-assignment Coding. *ICCV*, 2011. 2, 4

[35] Kanglin Liu, Wenming Tang, Fei Zhou, and Guoping Qiu. Spectral regularization for combating mode collapse in gans. In *ICCV*, pages 6382–6390, 2019. 3

[36] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 3, 4

[37] Takeru Miyato and Masanori Koyama. cgans with projection discriminator. In *ICLR*, 2018. 4

[38] Youssef Mroueh, Tom Sercu, and Vaibhava Goel. Mcgan: Mean and covariance feature matching gan. In *ICML*, pages 2527–2535, 2017. 3

[39] Yao Ni, Dandan Song, Xi Zhang, Hao Wu, and Lejian Liao. Cagan: Consistent adversarial training enhanced gans. In *IJCAI*, pages 2588–2594, 2018. 4

[40] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, pages 722–729. IEEE, 2008. 6

[41] Dylan M. Paiton, David Schultheiss, Matthias Kuemmerer, Zac Cranko, and Matthias Bethge. The geometry of adversarial subspaces, 2022. 26

[42] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *ACSSC*, pages 40–44, 1993. 2, 4

[43] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv:1511.06434*, 2015. 3

[44] Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, and Thomas Hofmann. Stabilizing training of generative adversarial networks through regularization. *arXiv:1705.09367*, 2017. 1

[45] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, pages 2234–2242. 2016. 6

[46] Fatemeh Shiri, Xin Yu, Piotr Koniusz, and Fatih Porikli. Face destylization. In *DICTA*, pages 1–8. IEEE, 2017. 1

[47] Fatemeh Shiri, Xin Yu, Fatih Porikli, Richard Hartley, and Piotr Koniusz. Identity-preserving face recovery from portraits. In *WACV*, 2018. 1

[48] Fatemeh Shiri, Xin Yu, Fatih Porikli, Richard Hartley, and Piotr Koniusz. Identity-preserving face recovery from stylized portraits. *IJCV*, 127(6-7):863–883, 2019. 1

[49] Fatemeh Shiri, Xin Yu, Fatih Porikli, Richard Hartley, and Piotr Koniusz. Recovering faces from portraits with auxiliary facial attributes. In *WACV*, pages 406–415, 2019. 1

[50] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014. 4

[51] Hugo Steinhaus. Sur la division des corps matériels en parties. *Bull. Acad. Pol. Sci., Cl. III*, 4:801–804, 1957. 2, 4

[52] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. On data augmentation for gan training. *TIP*, 30:1882–1897, 2021. 3

[53] Hung-Yu Tseng, Lu Jiang, Ce Liu, Ming-Hsuan Yang, and Weilong Yang. Regularing generative adversarial networks under limited data. In *CVPR*, 2021. 2, 7, 11, 12

[54] Jan van Gemert, Jan-Mark Geusebroek, Cor Veenman, and Arnold Smeulders. Kernel Codebooks for Scene Categorization. *ECCV*, 5304:696–709, 2008. 2, 4

[55] Jinjun Wang, Jianchao Yang, Kai Yu, Fengjun Lv, Thomas Huang, and Yihong Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010. 2, 4, 5

[56] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, pages 8798–8807, 2018. 1

[57] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. 2016. 3

[58] Ryan Webster, Julien Rabin, Loic Simon, and Frederic Jurie. Detecting overfitting of deep generative networks via latent recovery. In *CVPR*, 2019. 1

[59] Xiang Wei, Zixia Liu, Liqiang Wang, and Boqing Gong. Improving the improved training of wasserstein gans. *ICLR*, 2018. 4

[60] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *CVPR*, 2018. 1

[61] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, pages 1794–1801, 2009. 2, 4

[62] Xin Yu, Basura Fernando, Bernard Ghanem, Fatih Porikli, and Richard Hartley. Face super-resolution guided by facial component heatmaps. In *ECCV*, pages 217–233, 2018. 1

[63] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *ICML*, pages 7354–7363, 2019. 3, 4

[64] Han Zhang, Zizhao Zhang, Augustus Odena, and Honglak Lee. Consistency regularization for generative adversarial networks. In *ICLR*, 2020. 3

[65] Ruixiang Zhang, Tong Che, Zoubin Ghahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *NeurIPS*, volume 31, pages 2365–2374, 2018. 1

[66] Shengyu Zhao, Zhijian Liu, Ji Lin, Jun-Yan Zhu, and Song Han. Differentiable augmentation for data-efficient gan training. *NeurIPS*, 33, 2020. 1, 2, 3, 7, 11, 12

[67] Yang Zhao, Chunyuan Li, Ping Yu, Jianfeng Gao, and Changyou Chen. Feature quantization improves gan training. In *ICML*, pages 11376–11386, 2020. 3, 4, 14

[68] Zhengli Zhao, Sameer Singh, Honglak Lee, Zizhao Zhang, Augustus Odena, and Han Zhang. Improved consistency regularization for gans. In *AAAI*, volume 35, pages 11033–11041, 2021. 4, 7

[69] Peng Zhou, Lingxi Xie, Bingbing Ni, Cong Geng, and Qi Tian. Omni-gan: On the secrets of cgans and beyond. In *ICCV*, pages 14061–14071, 2021. 2, 4, 6, 7, 14