A Low Rank Gaussian Mixture Latent Model for Face Generation

Benjamin Samuth, Julien Rabin, Frédéric Jurie, David Tschumperlé

A Low Rank Gaussian Mixture Latent Model for Face Generation

Benjamin Samuth[0009−0002−4666−2475], Julien Rabin[0000−0003−3834−918X], Fréderic Jurie[0000−0002−2686−0020], and David Tschumperlé[0000−0003−3454−5079]

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
{name}.users.greyc.fr
{firstname.name}@unicaen.fr

Abstract. Generative modeling of natural images has seen significant progress, but large-scale foundation models raise concerns about environmental impact, privacy, and biases. This motivates investigating more efficient and interpretable generative models. This work proposes a simple latent parametric generative model focused on realistic face generation, a domain that has seen success with neural networks. The model uses a low-dimensional latent representation from a pre-trained autoencoder, and proceeds in two stages: (1) modeling the latent distribution as a mixture of multivariate Gaussians trained on a limited dataset, and (2) generating low-rank random codes from this prior and remapping them using nearest neighbor matching. Comparative experiments demonstrate the advantages of the proposed approach.

Keywords: Generative Modeling · Auto-encoder · Face Generation.

1 Introduction

The field of deep learning-based generative modeling of natural images has witnessed substantial advancements in recent years. Popular methods now achieve hyper-realistic image synthesis by combining visual and natural language cues, enabling their use for downstream tasks such as image editing.

These successful generative models rely on a few powerful techniques. The first is dimensionality reduction. Self-supervised representation learning, primarily built on autoencoding neural networks, allows for the compact representation of images in a low-dimensional embedding space. Careful architectural design enables an interesting trade-off between the fidelity of image reconstruction and the compression rate [9].

While dimensionality reduction is mandatory for designing realistic generative models [16], latent representations offer two main benefits for generative modeling. First, pre-training a decoder helps reduce the complexity of generative models, improving training computational time and data requirements, as demonstrated in various cases [30,9,31]. Additionally, the encoder helps extract

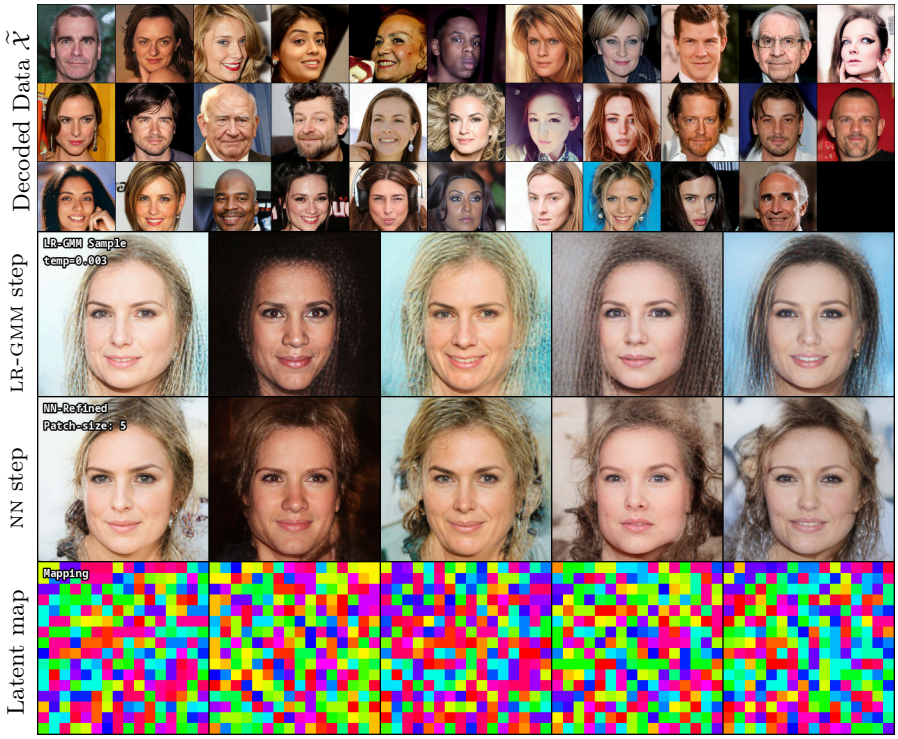


Fig. 1. Generated samples from a limited dataset of faces with LLR-GMM-NN, the proposed method described in Fig. 2. The LR-GMM step consists in estimating a low-rank latent gaussian mixture model fitted on a limited dataset $\mathcal{X}$ (here with only $N = 32$ training images $\widetilde{\mathcal{X}}$ showed after decoding by an auto-encoder). The dimensionality is controlled by a temperature parameter (here 0.3%). An additional NN step adds missing visual details by matching local features from the training set, captured by latent patches (with a size of 5). The latent map at the bottom indicates in color code the corresponding training sample used for each pixel.

perceptually meaningful features that achieve better results than the original color space for tasks like image comparison [10,18,48] and interpolation [23].

The most successful approaches in recent years have utilized or combined one of the following key techniques: i) Auto-regressive models, which sequentially predict masked pixels, either in the color space (as in PixelCNN [28]) or in the latent space (as in VQ-VAE [30]), often employing decoder-type transformer architectures [42] for this sequential prediction; ii) Adversarial training, where feed-forward neural network architectures trained using generative adversarial networks (GANs) [12,19] can produce highly realistic images; and iii) Diffusion models, iterative, denoising models that can generate realistic images, either in the color space [16], in the image latent representation [31], or in a multimodal representation [29].

The common thread among these successful techniques is their reliance on massive neural network architectures, often with billions of parameters, to attain

levels of realism that can deceive human perception. As a result, such large models require enormous amounts of training data and extensive computational resources during both training and inference.

However, this level of achievement comes with some remaining challenges. First, the use of over-parameterized deep neural networks has raised legitimate concerns about data privacy [46], which have also been shown to occur in generative modeling, particularly in GANs [14,44]. Some mechanisms, such as differential privacy [45], have been derived to mitigate these issues. Additionally, other generative models based on transformers have been shown to be able to memorize some training samples [9,27], a concern that has also been scrutinized for diffusion models [40,6].

Besides, the proliferation of such data-intensive and computationally demanding models raises numerous ethical and social concerns [22], including their environmental impact [3], legal implications concerning training with copyrighted content [38], and the presence of biases, illegal material [41], or corrupted samples [39] within the training dataset. These concerns underline the need to explore alternative approaches that are more resource-efficient, transparent, and less susceptible to ethical dilemmas.

In this context, this work aims to show that a simple latent generative model, both in terms of parameters and computing power, can achieve high performance in comparison to large state-of-the-art auto-regressive and diffusion models. Additionally, we further demonstrate that it allows using less data and yields a more explainable model. Lastly, we show that such a model can address other downstream tasks, such as image editing.

The proposed approach, illustrated in Figure 1, aims to generate plausible latent representations of images from a limited dataset of portrait examples, which can then be used to reconstruct the corresponding high-resolution images using a pre-trained decoder. The generative process is feed-forward and consists of two steps: i) Modeling the latent distribution of the limited dataset as a Gaussian mixture model (GMM). Combined with low-rank approximation (LR), this relatively simple model requires only a small number of parameters, thanks to the compact latent representation of the images, which allows for a limited training dataset. Yet, this approach still enables the imposition of long-range correlations in the final high-resolution images. ii) During inference, sparse latent codes are randomly generated to enforce the sharpness of the generated images. The second stage then consists of injecting missing details (high-frequency patterns) by iteratively projecting the latent pixels onto similar training samples, based on local comparisons (nearest-neighbor step or NN).

The rest of the paper is organized as follows: The next section (Section 2) describes related work in latent generative modeling and exposes their limitations. Section 3 introduces the proposed two-step generative model, consisting of a combination of a latent low-rank Gaussian mixture (§ 3.2) and a nearest-neighbor projection (§ 3.3). The experimental section (Section 4) shows qualitative and quantitative results for face generation, demonstrating the advantages of the proposed method compared to several competitive methods from the lit-

erature. Finally, we conclude in Section 5 by discussing some perspectives and future lines of work.

2 Related Work

State-of-the-art in generative modeling of face. Our work focus on face generation which has been a long standing problem in generative image modeling. One of the first milestone in realistic portrait generation has been GAN-based approaches, starting with PG-GAN [20] which introduced a progressive adversarial training on a curated dataset of 30k registered HD images (CelebA-HQ). With a different architecture allowing for explicit image stylization, Style-GAN [21] improved upon this work with 70k high-quality images under permissive licenses (FFHQ). These generative models, mapping a random latent code to a realistic image, can be used as priors for image editing by implicitly exploiting the latent pre-image. However, such plug-and-play techniques requires gradient-based optimization or the posterior training of a dedicated encoder.

As mentioned in the introduction, a popular technique to reduce data dimension and model size and training time is the use of a latent generative representation [30,9,31]. In a first stage, an auto-encoder is trained on the target dataset to learn a compact latent representation (*i.e.* with a very small spatial resolution) *via* an encoder that is perceptually well reconstructed by a decoder. To achieve this result, variational auto-encoders (VAE [23]) have been widely used [30] and are often combined with some other techniques, such as GANs [9].

In a second stage, a latent generative model is trained on the same dataset. After training, the parametric model is used to generate a random but plausible latent code that is decoded to synthesize a realistic image.

Auto-regressive models for images has been popularized with PixelCNN [28] and combined with quantized latent representation in VQ-VAE [30] to produce realistic images. With the advent of transformers and attention-based architectures [42], token-based decoder transformers has been successfully combined with latent representation [9]. A limitation of such methods comes from their sequential nature, requiring a lot of computations and a large number of parameters to impose long range correlation in long sequence.

More recently, diffusion and score-based models have been popularized since the seminal work of [16] and its application for text-to-image generation scaled to large datasets [29,33], in combination of with auto-encoders [31]. While iterative in nature by requiring thousand of denoising steps, the distillation of such stochastic models has met some success recently to accelerate the generation by training a single step generative network [34]. Yet, these models are still large in nature. The sheer amount of parameters contained in a state-of-the-art model requires several weeks of training with relatively massive computing power.

Besides these computational requirements and the related environmental considerations, a growing number of ethical issues is related to the use of these large models as black-boxes (reproduction of copyrighted material, reproduction of bias from the training data, privacy of the training data, etc). Addressing these

concerns demand more explainable and transparent generative image models. In this work, we investigate this line of research in the context of frugal models.

Generative model with limited parameters and data. Child et al. [7] showed that simple multi-scale yet very deep VAE could compete with much larger auto-regressive models. While being able to generate high quality portrait, the proposed model is still limited to small resolution (256 pixels) with hundred of millions of parameters.

One common practice to avoid re-training large models from scratch is to finetune their parameters on another target dataset. In particular, [43] notes its efficiency for generative methods in adversarial networks. A recently more popular approach has been to train a smaller, low-rank, auxiliary model to adapt the parameters to the target distribution [17,32,47]. Yet, the auxiliary model uses the large model even during inference, which still requires huge amount of memory and computations.

Few-shot generation main goal is to be able to generate images from a limited dataset, with very scarce information. Training models becomes in principle more difficult as most architectures are in that case prone to overfitting or mode collapse. [26] proposes an adversarial network adapted to a low amount of data points. They introduce skip-layer excitation modules that weight high-resolution features with the low-resolution ones. This allows them a robust training with lower parameter count. [2] specializes models to the target distribution thanks to a quantization codebook that specifically encodes patches of the limited dataset. They then train an auto-regressive model that can only generate codes from that constrained codebook. Both approaches however require an entire retraining if the limited dataset were to change. Even though considerably lower than large models, the training process can take hours of computing for just a dozen images, for instance.

Last, our work share some similarities with Latent-Patch [35]. In both case, a shallow latent generative model is proposed to train from a limited dataset. However, these generative models are completely different in nature. In Latent-Patch, a multi-scale non-parametric auto-regressive model is used, inspired from patch-based texture synthesis algorithms and the PatchMatch approach introduced for image editing. A limitation of this approach is the lack of long-range correlation by the sequential nature of the synthesis, which can results in global inconsistencies, such as face asymmetries (different eyes' color, earring on only one side, etc), different hair styles depending on the spatial location, or inconsistent background. In contrast, our model requires a few parameters (around 1M, so 10 to 100 times lower than other methods) to be trained, and only requires two steps that are parallel and thus faster to infer.

3 Latent LR-GMM-NN Generative Model

Overview. Like many aforementioned state-of-the-art approaches, we propose a latent generative model which is built on top of a pre-trained auto-encoder. The

next section 3.1 gives more detail on learning such a latent representation on an auxiliary dataset $\mathcal{A}$. The following sections then introduce the different steps of the proposed generative model, coined LLR-GMM-NN in this document, that are summarized in Figure 2.

As exposed in section 3.2 about learning the latent parametric model, latent representations of the training images of a limited target dataset $\mathcal{X}$ are first collected and compressed using dimensionality reduction operator P_0, by means of a principal component analysis (PCA). A Gaussian Mixture Model (GMM) is then fitted to the latent distribution of the target dataset $\mathcal{X}$ to capture the desired spatial correlation at coarse resolution. The first stage of inference (LR-GMM-step) finally consists in generating a latent code, and combines a Gaussian Mixture Model (GMM) with Low-Rank (LR) samples using sparse operators P_k.

The following NN-step, introduced in section 3.3, stands for Nearest-Neighbor projection, and consists in injecting details in the generated latent sample by matching local features (latent patches) from training examples.

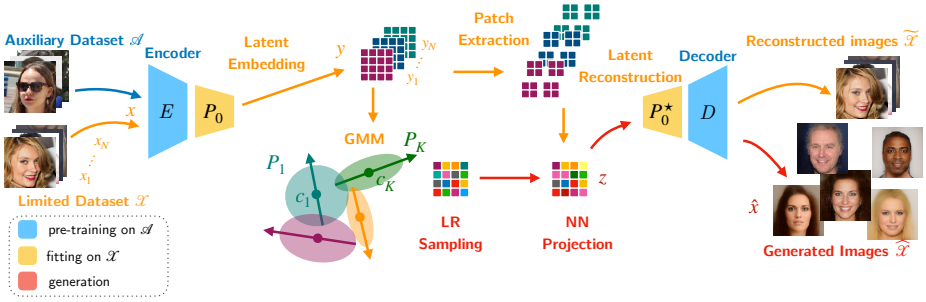


Fig. 2. Overview of the proposed LLR-GMM-NN method. A latent representation is obtained by pre-training an auto-encoder (E and D) on an auxiliary dataset $\mathcal{A}$. The model is fine-tuned on the target distribution of images by training a latent embedding (P_0 and $P_0^\star$) on an limited dataset $\mathcal{X}$ which reduce the dimension of the latent representation. A gaussian mixture model (GMM) is fitted on latent images, from which a low-rank approximation (LR) is used to generate random samples. A local Nearest-Neighbor (NN) projection based on patch-similarity is used to modify synthesized samples before decoding.

3.1 Latent Representation

A generic auxiliary dataset $\mathcal{A} = \{a_j\}_{j=1}^M$ is considered to learn an appropriate latent representation of images. We consider color images of size $3 \times H \times W$, where H (height) and W (width) indicates spatial dimension.

Auto-Encoder. To achieve this goal, we follow the paradigm of many state-of-the-art methods by first training an auto-encoder (AE) composed of an embedding deep-network E (which is used to learn an appropriate latent generative model

later on), and a decoder D (used to synthesize color images from random latent codes). As this strategy has been proven to be effective to learn various generative models (see *e.g.* VQ-VAE [30], VQ-GAN [9], latent-diffusion [31], Latent-Patch [35]), we assume that we can use a generic on-the-shelf auto-encoder providing a latent representation y of images at coarse spatial information $d_{\text{latent}} = c \times h \times w$ (*i.e.* $h \ll H$). The spatial grid is referred to as $\Omega := \{1, \dots h\} \times \{1, \dots w\}$ in the following.

Finetuning. As proposed in [35], a limited set of N images $\mathcal{X} = \{x_i\}_{i=1}^N$ is used to fine-tune such a generic auto-encoder to the target distribution of images. In order to restrict the number of parameters of the generative model to be trained, an affine projector P_0 is learnt from PCA on latent c-dimensional features to further reduce the dimension to $q \ll c$. Note that this is done while keeping spatial resolution $h \times w$ of latent representation $y_i = E(x_i)$ of images, in such a way that $d_0 = qhw$ is limited to a few thousands dimension. Formally, such an operator operates on each latent-pixel position p and writes

$$\forall\, p \in \Omega,\ P_0 : y(p) \in \mathbb{R}^c \mapsto \text{diag}(\sqrt{s_0}^{-1}) V_0 (y(p) - \bar{y}) \in \mathbb{R}^q \tag{1}$$

where $\bar{y}$ is the average of latent pixels and $V_0 \in \mathbb{R}^{q \times c}$ is the q principal eigenvectors, normalized using the corresponding eigen-values $s_0 \in \mathbb{R}^q$. For decoding, the transpose of V_0 is used to reconstruct latent representation at the required dimension c:

$$\forall\, p \in \Omega,\ P_0^\star : z(p) \in \mathbb{R}^q \mapsto V_0^\top \text{diag}(\sqrt{s_0}) z(p) + \bar{y} \in \mathbb{R}^c. \tag{2}$$

3.2 A Low-Rank latent Mixture Model

This first stage aims at learning a parametric model capable of synthesising random latent representation that: i) impose correlations across the coarse resolution of embedded images, and ii) capture the diversity of the target distribution, iii) with a limited budget, in both required training data, memory and computation time. To achieve these goals, we combine a gaussian mixture model with low-rank approximations that is trained on the limited target dataset.

GMM. The training dataset $\mathcal{X}$ is embedded as N latent vectors $\mathcal{Y} = \{y_i = P_0(E(x_i)) \in \mathbb{R}^{d_0}\}_{i=1}^N$. An expectation-maximization (EM) algorithm is used to fit the Gaussian Mixture Model (GMM) into K components, so that the training samples $\mathcal{Y}$ are split into K clusters $\{\mathcal{C}_k\}_{k=1}^K$. Each component is indexed by k and is parameterized with a multinomial probability π_k and with a multi-variate gaussian $\mathcal{N}(c_k, \Sigma_k)$. Mean $c_k \in \mathbb{R}^{d_0}$ and covariance $\Sigma_k \in \mathbb{R}^{d_0 \times d_0}$ are empirically estimated from the training data from $\mathcal{C}_k$. Due to the limited amount of data and the reduced dimension (*i.e.* N d_0-dimensional vectors), this training step is quite fast (a few seconds for small dataset such as $N = 100$), with the additional acceleration from GPU-parallelization (see *e.g.* [25]).

Recall that drawing a random sample from this GMM consists in

1. randomly choosing the index $k \in \{1..K\}$ using the multinomial probability law $\pi = (\pi_k)_{k=1}^K$,
2. sampling a random sample z using

$$\varepsilon \sim \mathcal{N}(0, I_{d_0}) \mapsto z = P_k \varepsilon + c_k \ \in \mathbb{R}^{d_0}, \tag{3}$$

where ε is a random latent variable drawn from a standard normal distribution, and P_k is a lower triangular matrix from the Cholesky decomposition of the covariance matrix, such that $\Sigma_k = P_k P_k^\top$.

LR *sampling.* While using a large number K of clusters allows to fit arbitrarily well the latent distribution, the number of parameters grows linearly with K. As a result, and in addition to restricting K to avoid overfitting, we propose to further reduce the number of parameters required to encode each Gaussian component by performing a supplementary dimensionality reduction. Relying again on PCA, a low-rank (LR) approximation of the covariance matrix Σ_k is used to limit the latent dimension of each component k to d_k. We have found that adapting d_k to the probability π_k give better visual results, as shown in experiments. In practice, we substitute the matrix $P_k \in \mathbb{R}^{d_0 \times d_k}$ in (3) which is defined, similarly to (1), from the matrix V_k of d_k principal eigenvectors of Σ_k as follows: $\forall\, k = 1..K$,

$$P_k : \varepsilon \in \mathbb{R}^{d_k} \mapsto z = \operatorname{diag}(\sqrt{s_k}^{-1}) V_k \varepsilon + c_k \ \in \mathbb{R}^{d_0} \tag{4}$$

As studied in experiments, this dimension reduction offers some control over the quality versus the diversity of generated samples.

3.3 Refinement step with iterated NN-projection

The second step of the generative process is non-parametric. In our setting, for each spatial position $p \in \{1, \dots h\} \times \{1, \dots w\}$, the random latent feature $z(p) \in \mathbb{R}^q$ generated from cluster indexed by k (as in (3)) is substituted with the nearest-neighbor (NN) latent feature $y_j(p)$ at the same position p and in the same cluster k. To define a relevant comparison of latent feature while taking into account the context, we consider the local $\omega \times \omega$ square neighborhood around p, that is $\forall\, p \in \Omega$

$$\text{NN} : z(p) \in \mathbb{R}^q \mapsto y_j(p) \text{ with } j = \underset{i=1..N \text{ s.t. } y_i \in \mathcal{C}_k}{\operatorname{argmin}} \|\Phi z(p) - \Phi y_i(p)\|^2 \tag{5}$$

where Φ is the patch-extractor defined as, $\forall\, p \in \{\rho, \dots h-\rho\} \times \{\rho, \dots w-\rho\}$

$$\Phi : z(p) \in \mathbb{R}^q \mapsto (z(p+h))_{h \in \{-\rho..\rho\}^2} \in \mathbb{R}^{q \times \omega^2}, \tag{6}$$

considering patches with a discrete radius ρ, such that $\omega = 2\rho + 1$. Some special care is required at the boundary of the domain for which zero-padding may be used for efficiency. Observe as well that only training patches in the same cluster as the query and at the same location p are involved in the NN search (5). When considering non-registered data, such restriction can be like discarded but increase complexity, similarly to attention modules.

Comparison with path-based synthesis. Last, note that this NN projection builds upon the observation from [36,35] that plausible latent code can be obtained by sampling directly local neighborhoods (or patches) of spatial resolution $\omega \times \omega$ from the training data. These approaches have been motivated by earlier patch-based techniques, such as the seminal work of [8] for texture synthesis, and the efficient algorithm of [1] for image editing.

However, contrarily to [35] where large patches from the training set are sequentially copied to generate a new latent representation similar to a patch-work, and to some extent to auto-regressive models [30,9] or diffusion models with attention mechanisms [31], the proposed nearest-neighbor projection only copies a latent feature $y_j(p)$ and is independent for each spatial position, and thus can be performed in parallel. This allows some significant speed-up of the latent synthesis and avoids overfitting the training set as shown in experiments.

Some links with non-local mean filtering and attention mechanism are further discussed in the appendix.

4 Experiments and Applications

In this section, we first describe the experimental setting (§ 4.1). Then, we show some quantitative and qualitative results of the proposed approach for face generation in section 4.2. A comparative analysis with other methods is also carried out. Additional results, including an ablation study demonstrating the role of each generative step together with the impact of some hyper-parameters are proposed the supplementary material.

4.1 Experimental setting

Auto-encoder. We use the VQ-GAN auto-encoder [9], an architecture originally conceived for transformer-based generation. It notably combines an adversarial loss with a quantization codebook, learning a both compressed and meaningful latent representation. The auxiliary dataset $\mathcal{A}$ in our setting is FFHQ [21], a large dataset of faces high-quality faces.

The auto-encoder is trained using color images at resolution $H = W = 256$ pixels (*i.e.* for a total dimension $3HW = 196,608$). The latent representation of an image has a $h = w = 16$ pixels resolution with $c = 256$ channels (*i.e.* for a total dimension $d_{\text{latent}} = 65,536$).

Fine-tuning The feature dimension reduction of the auto-encoder using PCA (as detailed in § 3.1) is performed on latent pixels of the limited target encoded dataset $E(\mathcal{X})$, in which $\mathcal{X}$ is a random subset of N images from CelebA-HQ [20], to learn the projector P_0 (1). Doing so allows us to reduce the channel dimension of pixels from $c = 256$ to $q = 8$ with minimal information loss. In the following, the default size of this limited dataset is set to $N = 1024$, but experiments will show that the model is sufficiently robust for N to range from as low as $N = 32$ (as in the few-shot regime shown in Figure 1) to the full dataset with $N = 28k$.

Fitting the generative model. During the training phase of the proposed generative model (LR-GMM), several hyper-parameters need to be set. By default, we fit $K = 8$ components with dimensions $d_k = \lfloor \lambda \pi_k d_0 \rfloor$ where temperature parameter $\lambda = 0.12$. Optimization is performed using standard EM algorithm (here we rely on the non-parallel implementation of `sklearn.mixture.GaussianMixture`).

Memory and computations. Additional details about memory footprint and computation time are provided in the supplementary material. In a nutshell, the proposed model is both fast to train (few seconds for $N = 1000$ images) and to sample from (few milliseconds on a GPU).

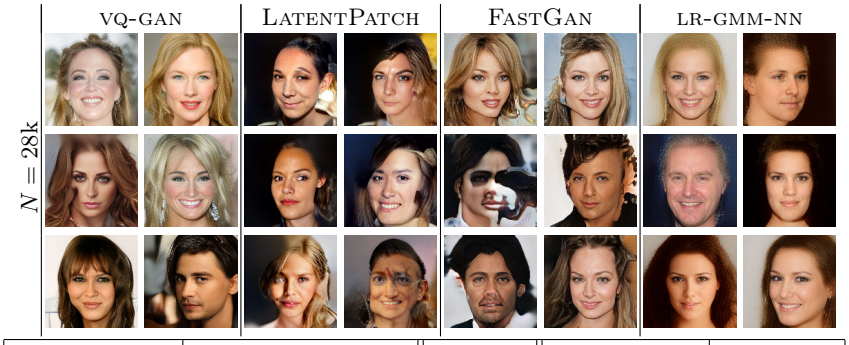
4.2 Face Generation

In this section, we discuss quantitative and qualitative results of the proposed method for portrait generation in two regimes (limited or large dataset), and we provide comparisons with relevant methods.

Fine-tuning on a small dataset. Figure 1 illustrates the two steps of proposed generative model when resorting to only $N = 32$ random training images from CelebAHQ. In this setting, only $K = 1$ component is used. Random samples obtained from the LR-GMM step alone synthesize the appropriate main face features (eyes, nose and mouth) with plausible spatial correlation. However, some other regions is the image may be unrealistic, such as the face contour and the hair, the lack of details, the existence of artificial high-frequency patterns, etc. The NN projection step (here with only one iteration) aims at improving this aspect by replacing local features with examples from the training images. This observation is confirmed by several metrics reported later on. in Table 1. The "Latent map" (at the bottom of Fig. 1 with an arbitrary colormap) corresponds to the index of the nearest neighbor; in addition to the visual comparison with the training images (top of Fig. 1), it demonstrates that the model is not overfitting training samples, even locally.

Quantitative analysis. Now we consider a large dataset in order to compute objective metrics such as FID [15] and IPR [24]. Recall that the Fréchet Inception Distance measures the dissimilarity between empirical distributions of real and synthesized samples, and that Improved Precision-Recall aims at measuring the tradeoff between quality and diversity of synthetic images. For this purpose, a total of 10k images are generated with several methods. Results reported in Table 1 for $N = 1024$ and $N = 28$k first show that both steps (LR-GMM and NN) of the proposed method are required to improve the quality of samples. The comparison with LatentPatch demonstrates that the proposed method can achieve similar performance without overfitting the training set (which is not penalized by these metrics). Besides, comparisons with two larger parametric models (FASTGAN [26] and VQ-GAN [9]) that requires hours of training show that our method is competitive. Extensive visual comparisons in supplementary material corroborate these results: the proposed method yields high quality samples

(high precision) but lacks some diversity (lower recall). In particular, the proposed method struggles to create realistic hair pattern, which is penalized by the FID.



Datasize $\lvert\mathcal{X}\rvert$	**Method**	FID (↓)	Precision (↑)	Recall (↑)
$N = 1024$	LatentPatch [35]	58.4	0.55	0.01
	LR-GMM	73.8	0.47	0.03
	LR-GMM-NN	58.1	0.52	0.07
$N = 28$k	VQ-GAN* [9]	10.5	0.52	0.51
	LatentPatch [35]	35.1	0.54	0.19
	FastGan [26]	31.8	0.50	0.04
	LR-GMM	84.1	0.53	0.01
	LR-GMM-NN	45.0	0.68	0.06

Table 1. FID [15] and Precision-Recall [24] metrics for different methods. Our method (LR-GMM-NN) is tested in two settings: $N = 1024$ and $N = 28,000$ images. LR-GMM indicates the proposed method without the refinement NN step, demonstrating its interest. See the supplementary material for more visual samples. *Note that both the transformer and auto-encoder of VQ-GAN have been trained on $\mathcal{X}$ rather than $\mathcal{A}$.

Comparison with LatentPatch. Figure 3 offers a comparison of both approaches when considering the same dataset of $N = 1024$ images. We already reported some methodological difference with LatentPatch [35] in section 3.3, completed with numerical (computation time) in the supplementary material where our approach is reported to yield a 30× speed-up. Recall that LatentPatch is a non-parametric patch-based generative model that does not require any training, and that aims at being explainable. To do so, similarly to other patch-based generative models such as [13], it explicitly synthesizes new image samples by copying local regions from examples images. This can be seen from the Nearest-Neighbor index, as defined by (5) and referred to the "latent map" (in this case, the colormap is cluster-dependant). In contrast with LatentPatch which copies large regions of latent features to achieve realistic portrait generation, the proposed method combines only latent pixels from training samples,

thus with more variety without overfitting some training samples. As a result, thanks to the randomness of samples from the GMM, we only make use of the nearest feature for the refinement, rather than sampling randomly from the top k-NN as in LATENTPATCH or in VQ-GAN. For all methods, the use of a latent representation then allows for a nice blending of the selected features.

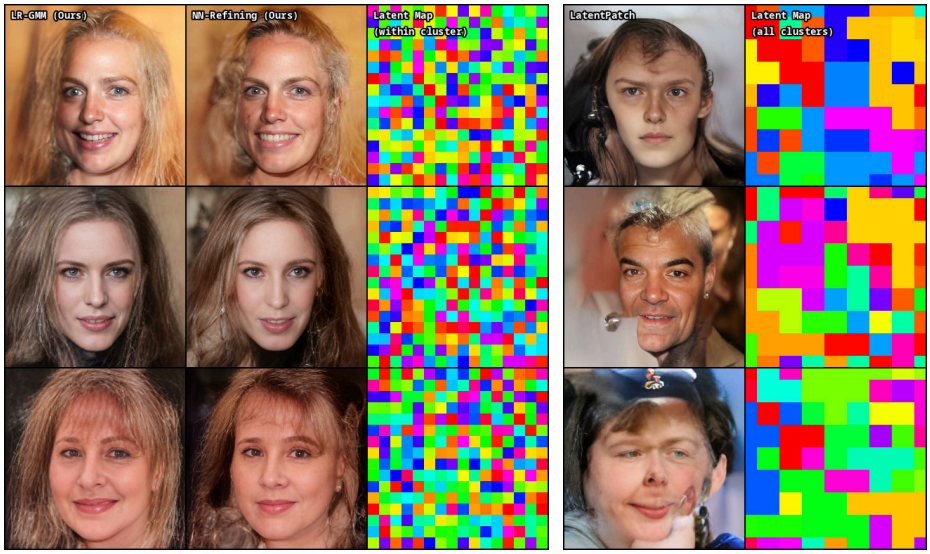


Fig. 3. Comparisons between LLR-GMM-NN on the left, and LATENTPATCH [35] on the right, both fitted on the same $N = 1024$ dataset. The proposed approach benefits from training a small parametric model to impose spatial coherence.

4.3 Ablation study

The supplementary material includes an ablation analysis that offers extensive visual comparisons, corroborating and complementing the performance scores reported in earlier experiments. This is supplemented with additional experiments that discuss the significance of each stage of the proposed generative model, the role of various hyperparameters, and the computation time.

5 Discussion and perspectives

Summary. A new approach has been presented for face-generation from limited dataset. Rather than training a large auto-regressive model such as VQ-VAE or VQ-GAN, our approach shows that in this context a simpler parametric model can be efficiently learned from a few samples. The proposed latent generative model mainly consists in two stages. The first one fits a gaussian mixture model

from the limited dataset. This model is then reduced in dimension to accelerate training and inference, but also improve quality results. This simplistic model provides random samples with the desired spatial correlations. The final stage consists in improving the quality of the synthesized samples by local nearest-neighbor projections. Experimental analysis demonstrates the speed-up of the proposed method in comparison with some previous work from the literature, while still providing similar or better visual quality.

It is noteworthy that previous works, such as [11,37], have already demonstrated the benefits of combining regularized deterministic auto-encoders with Gaussian Mixture Models to enhance vanilla Variational auto-encoder models. Our approach differs from [37], where latent dimensions are assumed to be independent, enabling the estimation of univariate GMMs during training. Similar to [11], we demonstrate that a multivariate GMM can be post-estimated to capture correlations between latent variables and improve sample quality. However, our work also diverges from [11] in several key ways. First, we employ dimensionality reduction to enhance sample quality, as shown in the ablation study in the supplementary material. Additionally, our method leverages the spatial information in the latent representation to refine local statistics using NN projections. Furthermore, instead of utilizing the entire training dataset used for the regularized AE, we demonstrate that a limited sample is sufficient to fit the latent model of an AE-GAN to produce realistic samples.

Limitations and perspectives. The proposed method however suffers from a few shortcomings when compared to some previous methods. The first one is related to the resolution of synthesized images that is same as training samples, like most generative models. This is yet not the case with auto-regressive models that can generate latent codes of arbitrary size, which is handy for other tasks such as generating landscapes. An interesting line of research in such context would be to extend our model by considering GMM with limited range for spatial correlations. In comparison with latentPatch, another current limitation of the proposed model is its extension to conditional generation. With an auto-regressive model, a latent code can be easily provided as a context. In our setting, this is also achievable but not as straightforward as one need to train a conditional gaussian noise, which is an interesting perspective left for future work.

Acknowledgements This work is funded by the project ANR-19-CHIA-0017.

Appendix – Supplementary Material for "A Low Rank Gaussian Mixture Latent Model for Face Generation"

Benjamin Samuth, Julien Rabin, Fréderic Jurie, and David Tschumperlé

Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

This appendix is composed of the following sections:

- Section 1 shows additional visual results from training the model on a large dataset and comparisons with three generative models;
- Section 2 discusses the relationship between the proposed refinement step and some related filtering methods (non-local means and attention-based mechanism);
- Section 3 reports the number of parameters of the model and compare computation time for training and inference with an other generative model:
- Section 4 discusses the interest of each stage of the generative model and the role of some major hyper-parameters.

1 Additional visual results

Figures 1 and 2 show high quality generated samples from training the proposed LR-GMM-NN model with the full training set of CelebA-HQ ($N = 28k$). In this experiment, $K = 64$ components are used to fit the GMM.

A visual comparison with other methods (LATENTPATCH [35], FASTGAN [26] and VQ-GAN [9]) is provided in Figures 3 and 4. Some additional technical details about this experiment:

- all generative models are trained on the full training set of CelebA-HQ (dataset $\mathcal{X}$) at a resolution of 256 pixels;
- we have trained FASTGAN from scratch on $\mathcal{X}$ using the official code[1] with default parameters (with a total of more than 29M trainable parameters, requiring 1 hour of training);
- LATENTPATCH and LR-GMM-NN use the auto-encoder architecture of VQ-GAN that is pre-trained on FFHQ (auxilliary dataset $\mathcal{A}$). The generative model of LATENTPATCH is parameter-free, and LR-GMM-NN uses approximately 1M parameters.
- we have used an official trained version of the generative model from VQ-GAN[2], meaning that both the auto-encoder and the transformers are trained on CelebA-HQ (dataset $\mathcal{X}$); the latent generator itself has more than 800M training parameters and requires several days of training.

[1] https://github.com/odegeasslbc/FastGAN-pytorch
[2] https://github.com/CompVis/taming-transformers

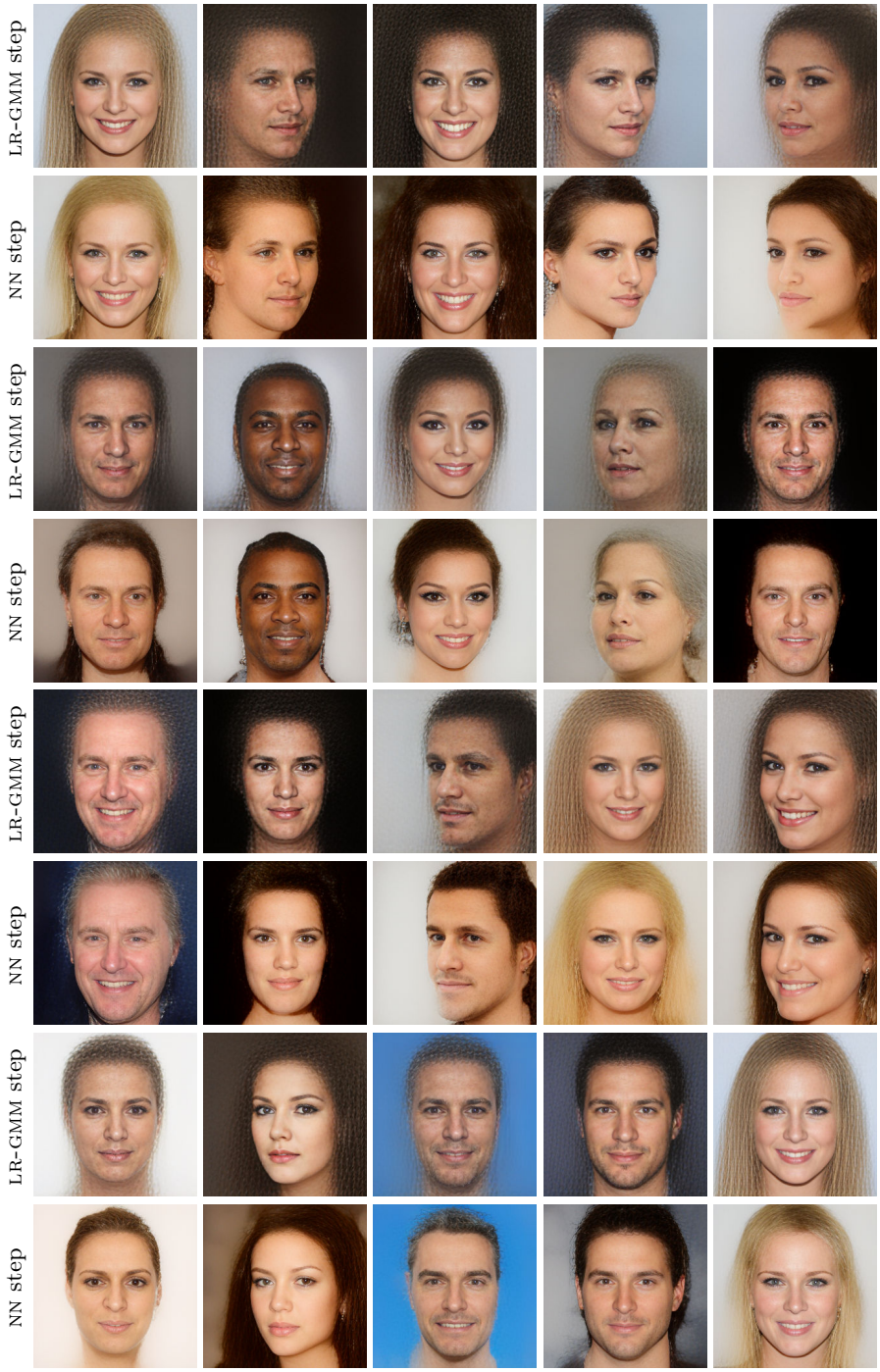


Fig. 1. Generated samples after the first (LR-GMM) and second (NN) stage of the proposed latent model.

Fig. 2. Generated samples after the first (LR-GMM) and second (NN) stage of the proposed latent model.

Fig. 3. Visual comparison of generated samples from our method (LR-GMM-NN), LatentPatch [35], FastGAN [26] and VQ-GAN [9], using the same training dataset as in Figs. 1 and 2.

Fig. 4. Visual comparison of generated samples from our method (LR-GMM-NN), LatentPatch [35], FastGAN [26] and VQ-GAN [9].

2 NN-projection and latent image filtering

In this section, we compare the proposed NN-projection stage with non-local filtering and attention mechanisms.

Link to Non-local filtering. The proposed NN-step is actually closely related to the Non-Local-Means filter [5] (NLM) which would writes in our setting as:

$$\text{NLM} : z(p) \in \mathbb{R}^q \mapsto \frac{1}{C(p)} \sum_q z(q) w(p,q) \text{ with } w(p,q) = e^{-\frac{\|\Phi z(p) - \Phi z(q)\|^2}{h^2}},$$

where $C(p) = \sum_q w(p,q)$. In our pipeline, the NN projection (Eq. 5 in the paper) is first obtained by setting the scaling parameter h (which acts as a temperature) to 0, and the averaged "noisy" samples $z(q)$ are replaced "clean" samples y_i from the same cluster and spatial location.

Link to Attention. Note that, by extension, the NN-projection is also related to attention mechanisms, where queries and keys correspond to patches, and values correspond to training local features, again restricted to a subset of values to speed-up computations. In the popular softmax-attention formulation, negative distances are replaced by scalar product. Besides, our approach is analogous to transformers in [9] in the fact that local features are copied: the key difference here is that VQ-GAN transformers implicitly copy features from a limited codebook (which is shared across all spatial location), while our approach use a limited set of features that is however different in each spatial location. Learning a limited set of key patches to reduce memory footprint is an interesting research direction that we did not explore in this work.

Iterated filtering. As such, this step can be iterated to improve prediction, similarly to NLM [4] or attention modules in transformers [42]. However, only one iteration has been used in all experiments, as the improvement is not worth the overhead.

3 Model Scale and Computational Efficiency

Number of parameters and memory footprint. Recall that the model is composed of three main parts that requires training data. Two of them necessitates to fit some parameters:

- fine-tuning the auto-encoder with P_0 (and $P_0^\star$) requires $(c+1)d_0$ parameters, where with $d_0 = hwq = 2048$, to store the mean and normalized eigenvectors; using default values from experimental section § 4, this amounts to a total of approximately 526k parameters;
- fitting a latent GMM with K components requires to store $K(d_k+1)d_0$ values; using $K = 10$ and a constant dimension $d_k = 48$ amounts approximately to 1M parameters;

Last, the iterated NN-projections (in Eq. 5) do not involve training parameters, but some memorization of latent patches. Considering a limited dataset of $N = 1000$ images, this represents a total of $Nd_0 \approx 2$M floating values to store. This figure can be however drastically reduced by taking advantage of the quantized latent space from VQ-GAN: using a codebook with $n = 1024$ vectors only requires $hwN = 256$k integers to store. Although we did not use the codebook in any of the experiments of this work, in practice we did not notice any significant loss in quality by visual inspection when introducing this quantization. An additional benefit from using a codebook would be to accelerate the computation of the NN-projection, although it would necessitate the storage and pre-computation of the $\frac{n(n+1)}{2}$ distances between quantized features.

Operation	LatentPatch [35]	LR-GMM-NN
P_0 Compute	0.127 ms	
Encoding $\mathcal{X}$	4.363 s	
Training	N/A	9.051 s
Total (Setup)	**4.490 s**	**13.541 s**
Sampling (/image)	581 ms	0.338 ms
Refining (/image)	N/A	11 ms
Decoding (/image)	11 ms	
Total (Generation)	**592 ms**	**22.338 ms**

Table 1. Computation time of our method, compared to Latent-Patch [35]. In this test, we encode a limited dataset $\mathcal{X}$ of size $N = 1024$. For an additional training time, we are able to make the generation process roughly 30 times faster.

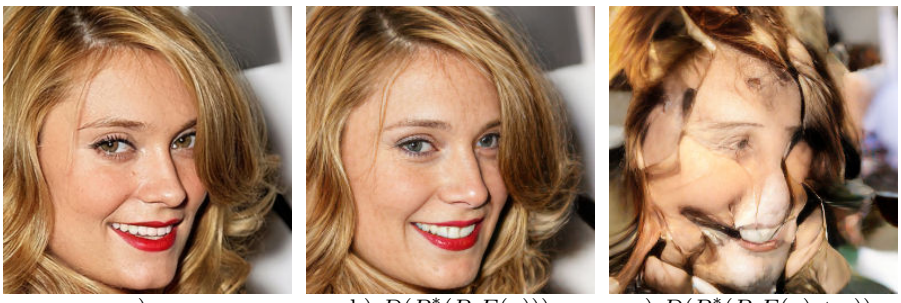
Computational Time. Table 1 reports computational time for training the proposed model from a dataset with $N = 1024$ images and generating one random sample. Note that all operations are performed on GPU, except fitting the GMM which is done on CPU.

As a reference, a comparison with the sequential non-parametric approach of LatenPatch [35] is provided. As expected, our parametric model requires a few more seconds for setting up, as we need to fit the parametric model, but is also much faster to sample from.

4 Ablation study

Latent representation of the auto-encoder. Figure 5 shows that even a small random perturbation $\varepsilon \sim \mathcal{N}(0, I_{d_0})$ of the latent representation of an encoded image $P_0E(z)$ generates completely out-of-distribution images. This demonstrates the necessity of training a parametric model to learn a useful prior to define a generative model, as further discussed in the following paragraph.

Besides, the comparison of Figs. 5 (a) and (b) validates the fact that the dimension reduction on features with P_0 does not degrade image quality, even if some noticeable difference occurs (such as eyes or skin color, high frequency pattern in hairs, or slight pose change).



a) x b) $D(P_0^*(P_0E(x)))$ c) $D(P_0^*(P_0E(x)+\varepsilon))$

Fig. 5. Manipulation in the latent space of the auto-encoder. (a) image from the training set $\mathcal{X}$ (b) decoded image from the auto-encoder (encoder E and decoder D) using the dimension reduction operators (P_0 and P_0^*). (c) synthesized image from a small gaussian perturbation $\varepsilon \sim \mathcal{N}(0, I_{d_0})$.

Role of the GMM for latent modeling. The encoder being trained using a variational auto-encoder, one could expect the latent distribution to be close to the standard gaussian distribution prescribed in the training loss derived from ELBO [23]. Figure 6(a) invalidates this assumption as random latent samples using this prior does not yield realistic synthesis of portraits. In addition, we found using other simple priors such as random codes from the codebook of VQ-GAN (Fig. 6(b)) does not improve the quality. Figure 6(c) shows the result of sampling directly latent pixels from the training set, and is somehow equivalent to our model with only the NN projection from a random initialization (rather than using LR-GMM). In contrast, using a prior on the complete latent representation (Fig. 6(d)) allows to get plausible spatial correlation.

Centroids Figure 7's top row shows the decoded centroids $D(P_0^\star c_k)$ obtained from the EM algorithm in the latent space for $K = 8$ and $N = 1024$ images. As expected, each average component of the GMM captures main and sharp facial features, such as pose, gender, skin or hair color. On the contrary, parts of the image that have the most variability (background, ears, hair, etc) are blurry and can exhibit high frequency patterns due to the dimension reduction from P_0. The bottom row of Fig. 7 displays directly the latent code, using a colormap based on the three first channels among $q = 8$ (recall that these channels correspond to normalized eigen-vectors defined from P_0).

Impact of low-rank sampling. The experiment in Fig. 8 scrutinizes the role of the dimension reduction parameter d_k. Using (4), random gaussian samples are

a) $z(p) \sim \mathcal{N}(0, I_{d_0})$ b) $z(p) \sim \mathcal{U}(\mathcal{C})$ c) $z(p) \sim \mathcal{U}(\{y_i(p)\}_i)$ d) $z \sim \mathcal{N}(\mu_k, \Sigma_k)$

Fig. 6. Decoded image from random latent samples using different priors: (a) corresponds to sampling each latent pixel from a standard gaussian; (b) corresponds to uniformly sampling each latent pixel from the codebook; (c) corresponds to uniformly sampling each latent pixel from training samples (*i.e.* NN step alone); (d) corresponds to sampling a latent code from a component of a gmm (*i.e.* LR-GMM step alone).

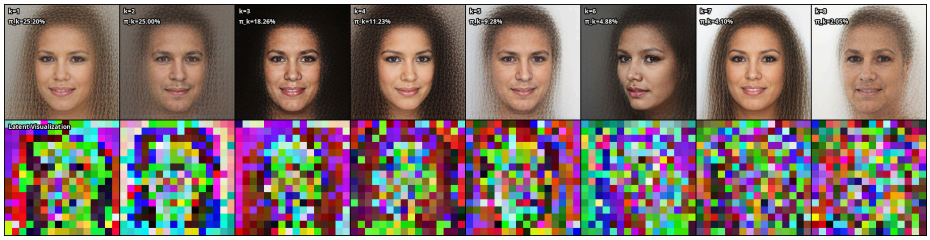


Fig. 7. Decoded means of the K components estimated from $N = 1024$ images of CelebA-HQ.

drawn using the same random seed but with different matrices P_k obtained for different values of the parameter d_k in the range $[1, d_0]$. Decoded latent samples shows that d_k offers a tradeoff between quality and the diversity of the generated faces.

Fig. 8. Illustration of low-rank sampling. Limiting the dimension d_k of the GMM components accelerates and reduces the memory footprint of the model while also offering a tradeoff between diversity and quality.

References

1. Barnes, C., Shechtman, E., Finkelstein, A., Goldman, D.B.: Patchmatch: A randomized correspondence algorithm for structural image editing. ACM Trans. Graph. **28**(3), 24 (2009)
2. Ben-Moshe, L., Benaim, S., Wolf, L.: FewGAN: Generating from the joint distribution of a few images. In: IEEE ICIP. pp. 751–755 (2022)
3. Berthelot, A., Caron, E., Jay, M., Lefèvre, L.: Estimating the environmental impact of generative-ai services using an lca-based methodology (2023)
4. Brox, T., Cremers, D.: Iterated nonlocal means for texture restoration. In: International Conference on Scale Space and Variational Methods in Computer Vision. pp. 13–24. Springer (2007)
5. Buades, A., Coll, B., Morel, J.M.: A non-local algorithm for image denoising. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). vol. 2, pp. 60–65. Ieee (2005)
6. Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramèr, F., Balle, B., Ippolito, D., Wallace, E.: Extracting training data from diffusion models. arXiv preprint arXiv:2301.13188 (2023)
7. Child, R.: Very deep vaes generalize autoregressive models and can outperform them on images. In: International Conference on Learning Representations (2020)
8. Efros, A., Leung, T.: Texture synthesis by non-parametric sampling. In: Proceedings of the Seventh IEEE International Conference on Computer Vision. vol. 2, pp. 1033–1038 vol.2 (1999)
9. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12873–12883 (June 2021)
10. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2414–2423 (2016)
11. Ghosh, P., Sajjadi, M.S.M., Vergari, A., Black, M., Scholkopf, B.: From variational to deterministic autoencoders. In: International Conference on Learning Representations (2020)
12. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
13. Granot, N., Feinstein, B., Shocher, A., Bagon, S., Irani, M.: Drop the GAN: In defense of patches nearest neighbors as single image generative models. In: Proceedings of the IEEE/CVF CVPR. pp. 13460–13469 (June 2022)
14. Hayes, J., Melis, L., Danezis, G., De Cristofaro, E.: Logan: Membership inference attacks against generative models. Proceedings on Privacy Enhancing Technologies (2019)
15. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)
16. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
17. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)

18. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14. pp. 694–711. Springer (2016)
19. Kang, M., Zhu, J.Y., Zhang, R., Park, J., Shechtman, E., Paris, S., Park, T.: Scaling up gans for text-to-image synthesis. In: Proceedings of the IEEE/CVF CVPR. pp. 10124–10134 (2023)
20. Karras, T., Aila, T., Laine, S., Lehtinen, J.: Progressive growing of gans for improved quality, stability, and variation. In: International Conference on Learning Representations (2018)
21. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE/CVF CVPR. pp. 4401–4410 (2019)
22. Kenthapadi, K., Lakkaraju, H., Rajani, N.: Generative ai meets responsible ai: Practical challenges and opportunities. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 5805–5806 (2023)
23. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
24. Kynkäänniemi, T., Karras, T., Laine, S., Lehtinen, J., Aila, T.: Improved precision and recall metric for assessing generative models. Advances in Neural Information Processing Systems **32** (2019)
25. Lee, S.X., Leemaqz, K.L., McLachlan, G.J.: A simple parallel em algorithm for statistical learning via mixture models. In: 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA). pp. 1–8. IEEE (2016)
26. Liu, B., Zhu, Y., Song, K., Elgammal, A.: Towards faster and stabilized gan training for high-fidelity few-shot image synthesis. In: International Conference on Learning Representations (2020)
27. Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A.F., Ippolito, D., Choquette-Choo, C.A., Wallace, E., Tramèr, F., Lee, K.: Scalable extraction of training data from (production) language models. arXiv preprint arXiv:2311.17035 (2023)
28. van den Oord, A., Kalchbrenner, N., Espeholt, L., kavukcuoglu, k., Vinyals, O., Graves, A.: Conditional image generation with pixelcnn decoders. In: Advances in Neural Information Processing Systems. vol. 29. Curran Associates, Inc. (2016)
29. Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., Chen, M.: Hierarchical text-conditional image generation with clip latents. arXiv:2204.06125 (2022)
30. Razavi, A., van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. In: Advances in Neural Information Processing Systems. vol. 32 (2019)
31. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF CVPR. pp. 10684–10695 (2022)
32. Ruiz, N., Li, Y., Jampani, V., Pritch, Y., Rubinstein, M., Aberman, K.: Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 22500–22510 (June 2023)
33. Saharia, C., Chan, W., Saxena, S., Li, L.e.a.: Photorealistic text-to-image diffusion models with deep language understanding. In: Advances in Neural Information Processing Systems. vol. 35, pp. 36479–36494. Curran Associates, Inc. (2022)
34. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. In: International Conference on Learning Representations (2021)

35. Samuth, B., Rabin, J., Tschumperlé, D., Jurie, F.: Latentpatch: A non-parametric approach for face generation and editing. In: 2023 IEEE International Conference on Image Processing (ICIP). pp. 1790–1794. IEEE (2023)
36. Samuth, B., Tschumperlé, D., Rabin, J.: A patch-based approach for artistic style transfer via constrained multi-scale image matching. In: 2022 IEEE International Conference on Image Processing (ICIP). pp. 3490–3494 (2022)
37. Saseendran, A., Skubch, K., Falkner, S., Keuper, M.: Shape your space: A gaussian mixture regularization approach to deterministic autoencoders. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems (2021)
38. Shan, S., Cryan, J., Wenger, E., Zheng, H., Hanocka, R., Zhao, B.Y.: Glaze: Protecting artists from style mimicry by {Text-to-Image} models. In: 32nd USENIX Security Symposium (USENIX Security 23). pp. 2187–2204 (2023)
39. Shan, S., Ding, W., Passananti, J., Zheng, H., Zhao, B.Y.: Prompt-specific poisoning attacks on text-to-image generative models. arXiv preprint arXiv:2310.13828 (2023)
40. Somepalli, G., Singla, V., Goldblum, M., Geiping, J., Goldstein, T.: Diffusion art or digital forgery? investigating data replication in diffusion models. arXiv preprint arXiv:2212.03860 (2022)
41. Thiel, D.: Identifying and eliminating csam in generative ml training data and models. Tech. rep., Technical report, Stanford University, Palo Alto, CA (2023)
42. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
43. Wang, Y., Wu, C., Herranz, L., Van de Weijer, J., Gonzalez-Garcia, A., Raducanu, B.: Transferring gans: generating images from limited data. In: Proceedings of the European conference on computer vision (ECCV). pp. 218–234 (2018)
44. Webster, R., Rabin, J., Simon, L., Jurie, F.: Detecting overfitting of deep generative networks via latent recovery. In: Proceedings of the IEEE/CVF CVPR. pp. 11273–11282 (2019)
45. Yoon, J., Jordon, J., van der Schaar, M.: PATE-GAN: Generating synthetic data with differential privacy guarantees. In: International Conference on Learning Representations (2019)
46. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: International Conference on Learning Representations (2017)
47. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: Proceedings of the IEEE/CVF CVPR. pp. 3836–3847 (2023)
48. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)