HARMONIZING MAXIMUM LIKELIHOOD WITH GANS FOR MULTIMODAL CONDITIONAL GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in conditional image generation tasks, such as image-to-image translation and image inpainting, are largely contributed by the success of conditional GAN models, which are often optimized by the joint use of the GAN loss with the reconstruction loss. However, we show that this training recipe shared by almost all existing methods always leads to a suboptimal generator and has one critical side effect: lack of diversity in output samples. In order to accomplish both training stability and multimodal output generation, we propose novel training schemes with a new set of losses that simply replace the reconstruction loss, and thus applicable to any conditional generation tasks. We show this by performing thorough experiments on image-to-image translation, super-resolution, and image inpainting tasks with Cityscapes, and CelebA dataset. A quantitative evaluation also confirms that our methods achieve great diversity of outputs while retaining or even improving the quality of images.

1 INTRODUCTION

Recently, much research has led a huge progress on *conditional image generation*, whose typical tasks include image-to-image translation (Isola et al. (2017)), image inpainting (Pathak et al. (2016)), super-resolution (Ledig et al. (2017)), and video prediction (Mathieu et al. (2016)). At the core of such advances is the success of conditional GANs (Mirza & Osindero (2014)), which improve GANs by allowing the generator to take an additional code or condition to control the modes of the data being generated. However, training GANs, including conditional GANs, is highly unstable and easy to collapse (Goodfellow et al. (2014)). To mitigate such instability, almost all previous models in conditional image generation exploit the reconstruction loss such as ℓ_1/ℓ_2 loss in addition to the GAN loss. Indeed, using these two types of losses is synergetic in that the GAN loss complements the weakness of the reconstruction loss that output samples are blurry and lack high-frequency structure, while the reconstruction loss offers the training stability required for convergence.

In spite of its success, we argue that it causes one critical side effect; the reconstruction loss (*e.g.* ℓ_1/ℓ_2 loss) aggravates the mode collapse, a notorious problem of GANs. In conditional generation tasks, which are intrinsically to learn one-to-many mapping, the model is expected to generate diverse outputs from a single input, depending on some stochastic variables (*e.g.* many realistic street scene images for a single segmentation map (Isola et al., 2017)). Nevertheless, such noise input rarely generates any diversity in output, and surprisingly many previous methods omit random noise source in their models. Most papers never mention the necessity of random noise, and a few others report that the model completely ignores the noise even if it is fed into the model. For example, Isola et al. (2017) state that the generator simply learns to ignore the noise, and even dropout fails to incur meaningful output variation.

The objective of this paper is to propose a new set of losses that can replace the reconstruction loss with losing neither the visual fidelity nor diversity in output samples. The core idea is to use maximum likelihood estimation (MLE) loss (*e.g.* ℓ_1/ℓ_2 loss) to predict conditional statistics of real data distribution instead of applying it directly to the generator as done in most existing algorithms. Then, we use this prediction to assist GAN training by enforcing the statistics of generated distribution to match the predicted statistics.

Our major contributions are three-fold. First, we show that there is a significant mismatch between the GAN loss and the reconstruction loss, thereby the model cannot achieve the optimality w.r.t. both

losses. Second, we propose novel training schemes named MLMM and MCMLE that enable the model to accomplish both training stability and multimodal output generation. Our methods simply replace the reconstruction loss, and thus applicable to any conditional generation tasks. Finally, we show the effectiveness and generality of our methods through extensive experiments on three generation tasks. Our methods outperform baselines in terms of realism and diversity.

2 Related Works

Conditional Generation Tasks. Since the advent of GANs (Goodfellow et al. (2014)) and conditional GANs (Mirza & Osindero (2014)), there has been a large body of work in conditional generation tasks. A non-exhaustive list includes *image translation* (Isola et al., 2017; Wang et al., 2017; 2018; Zhang et al., 2017b; Lyu et al., 2017; Sangkloy et al., 2017; Xian et al., 2017; Zhang et al., 2017a; Liu et al., 2017), *image inpainting* (Pathak et al., 2016; Iizuka et al., 2017; Ouyang et al., 2018; Olszewski et al., 2017; Yu et al., 2018; Sabini & Rusak, 2018; Song et al., 2018; Li et al., 2017; Yang et al., 2018; Yeh et al., 2017), *super-resolution* (Ledig et al., 2017; Xie et al., 2018; Mahapatra et al., 2017; Xu et al., 2017; Bulat et al., 2018; Sajjadi et al., 2017), *video prediction* (Mathieu et al., 2016; Jang et al., 2018; Lu et al., 2017; Villegas et al., 2017; Zhou & Berg, 2016; Bhattacharjee & Das, 2017; Vondrick & Torralba, 2017).

However, existing models have one common limitation: lack of stochasticity for diverse output. In spite of the fact that the tasks are to be one-to-many mapping, they ignore random noise input which is necessary to generate diverse samples from a single input. A number of works such as (Mathieu et al., 2016; Isola et al., 2017; Xie et al., 2018) have tried injecting random noise into their models but discovered that the models discard it and instead learn a deterministic mapping.

Multimodality Enhancing Models. It is not fully understood yet why conditional GAN models fail to learn the full multimodality of data distribution. Recently, there has been a series of attempts to incorporate stochasticity in conditional generation as follows.

(1) *Conditional VAE-GAN.* VAE-GAN (Larsen et al., 2016) is a hybrid model to combine the decoder in VAE (Kingma & Welling, 2014) with the generator in GAN (Goodfellow et al., 2014). Its conditional variants have been also proposed such as CVAE-GAN (Bao et al., 2017), BicycleGAN (Zhu et al., 2017), and SAVP (Lee et al., 2018a). These models harness the strengths of the two models, output fidelity by GANs and diversity by VAEs, and can produce a wide range of realistic images. Intuitively, the VAE structure drives the generator to exploit latent variables to represent multimodality of the conditional distribution.

(2) *Disentangled representation.* Huang et al. (2018) and Lee et al. (2018b) propose to learn disentangled representation for multimodal unsupervised image-to-image translation. These models split the embedding space into a domain-invariant space for sharing information across domains and a domain-specific space for capturing styles and attributes. The models encode an input to the domain-invariant embedding and sample domain-specific embedding from some prior distribution. By feeding the two embeddings into the decoder of the target domain, the model can generate diverse samples in a target domain.

Conditional VAE-GANs and disentangling-based methods both leverage the latent variable to prevent the model from discarding the multimodality of output samples. In this paper, we present a simpler and orthogonal direction to achieve multimodal conditional generation by introducing novel loss functions that can replace the reconstruction loss.

3 Loss Mismatch of Conditional GANs

We briefly review the objective of conditional GANs in section 3.1, and discuss why the two loss terms cause loss of modality in the sample distribution of the generator in section 3.2.

3.1 Preliminary: Objective of Conditional GANs

The goal of conditional GANs is to learn to generate samples that are indistinguishable from real data for a given input. The objective of conditional GANs usually consists of two terms, the GAN

loss $\mathcal{L}_{\mathrm{GAN}}$ and the reconstruction loss $\mathcal{L}_{\mathrm{Rec}}$.

$$\mathcal{L} = \mathcal{L}_{\mathrm{GAN}} + \lambda \mathcal{L}_{\mathrm{Rec}} \tag{1}$$

Another popular loss term is the perceptual loss (Johnson et al., 2016; Bruna et al., 2016; Ledig et al., 2017). While the reconstruction loss encodes the pixel-level distance, the perceptual loss is defined as the distance between the features encoded by neural networks. Since they share the same form (*e.g.* ℓ_1/ℓ_2 loss), we consider the perceptual loss as a branch of the reconstruction loss.

The loss $\mathcal{L}_{\mathrm{GAN}}$ is defined to minimize some distance measure (*e.g.* JS-divergence) between the true and generated data distribution conditioned on input x. The training scheme is often formulated as the following minimax game between the discriminator D and the generator G.

$$\min_G \max_D = \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x,z)))] \tag{2}$$

where each data point is a pair (x, y), and G generates outputs given an input x and a random noise z. Note that the discriminator observes x, which are crucial for the performance (Isola et al., 2017).

The most widely used reconstruction losses in conditional GAN literature are the ℓ_1 (Isola et al., 2017; Wang et al., 2017) and ℓ_2 loss (Pathak et al., 2016; Mathieu et al., 2016). Both losses can be formulated as follows with $p = 1, 2$ respectively.

$$\mathcal{L}_{\mathrm{Rec}} = \mathcal{L}_p = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_p^p]. \tag{3}$$

These two losses naturally stem from the maximum likelihood estimations (MLEs) of the parameters of Laplace and Gaussian distribution, respectively. The likelihood of dataset $(\mathbb{X}, \mathbb{Y})$ assuming each distribution is defined as follows.

$$P_{\mathrm{L}}(\mathbb{Y}|\mathbb{X};\boldsymbol{\theta}) = \prod_{i=1}^{N} \frac{1}{2b} \exp(-\frac{|y_i - f_{\boldsymbol{\theta}}(x_i)|}{b}),\ P_{\mathrm{G}}(\mathbb{Y}|\mathbb{X};\boldsymbol{\theta}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y_i - f_{\boldsymbol{\theta}}(x_i))^2}{2\sigma^2})$$

where N is the size of the dataset, and f is the model parameterized by $\boldsymbol{\theta}$. The central and dispersion measure for Gaussian are the mean and variance σ^2, and the correspondences for Laplace are the median and mean absolute deviation (MAD) b. Therefore, using ℓ_2 loss leads the model output $f_{\boldsymbol{\theta}}(x)$ to become an estimate of the conditional average of y given x, while using ℓ_1 loss is equivalent to estimate the conditional median of y given x (Bishop, 2006). Note that the model $f_{\boldsymbol{\theta}}$ is trained to predict the mean (or median) of data distribution, not to be individual samples from the distribution.

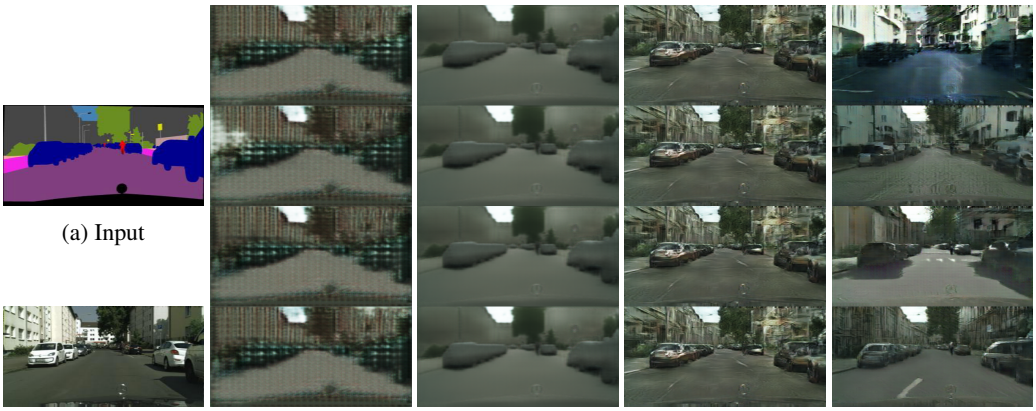
3.2 Loss of Modality by the Reconstruction Loss

We argue that the joint use of the reconstruction loss with the GAN loss could be problematic, because it can worsen the mode collapse, a notorious problem of GAN training. We discuss this argument both mathematically and empirically.

In conditional generation tasks, there are many outputs that can be paired with the input. For translating a semantic segmentation map to realistic photos (Isola et al., 2017), for instance, there are many possible outputs with different color, brightness and shapes for a single segmentation map. That is, the conditional variance of $P(y|x)$ should be non-zero. Although the conditional GAN is supposed to learn a one-to-many mapping, the ℓ_2 (or ℓ_1) loss forces the model to predict only the mean (or median) of $P(y|x)$, while pushing the conditional variance to zero. It eventually refrains from generating diverse samples indistinguishable from real data.

Specifically, we can show that the ℓ_2 loss minimizes the conditional variance of prediction to zero. We defer the detailed derivation to the appendix. According to James (2003), for any symmetric loss function $\mathcal{L}_s$ and an estimator $\hat{y}$ for y, the loss is decomposed into one irreducible term $\mathrm{Var}(y)$ and two reducible terms: $\mathrm{SE}(\hat{y}, y)$ and $\mathrm{VE}(\hat{y}, y)$, where SE refers to *systematic effect*, the change in error caused by bias, while VE refers to *variance effect*, the change in error caused by variance. For the ℓ_2 loss, SE and VE correspond to the squared bias and the variance, respectively. As a result, in the context of conditional generation tasks, the ℓ_2 loss minimizes the conditional variance of $\hat{y}$ given x to zero, which eventually gets rid of diversity of output samples.

Figure 1 shows the actual examples of the Pix2Pix model (Isola et al. (2017)) applied to translate segmentation labels to realistic photos in Cityscapes dataset. We slightly modify the model so that



(a) Input

(b) Ground truth (c) GAN (d) ℓ_1 (e) GAN + ℓ_1 (f) Ours

Figure 1: The loss mismatch in conditional GANs with examples of a Pix2Pix variant (Isola et al. (2017)) trained on Cityscapes dataset. Taking additional noise input, the model translates a segmentation label map to realistic images. We compare the results according to different loss functions (columns) with different noise input (rows). (c) Using the GAN loss alone is not enough to learn the complex data distribution due to its training instability. (d) Using the ℓ_1 loss only, the model generates images fairly close to the ground-truth, but they are far from being realistic. (e) The combination of the two losses enables the model to generate visually appealing samples. However, the model completely ignores noise input and always generates almost identical images. (f) In contrast, the model trained with our loss term generates a diverse set of images fully utilizing noise input.

it takes additional noise input. We use the ℓ_1 loss as the reconstruction loss as done in the original paper. We train four models that use different combinations of loss terms, and generate four samples with different noise input. As shown in Figure 1(c), the model trained with only the GAN loss fails to generate realistic images, since the signal from discriminator is too unstable to learn the translation task. In Figure 1(d), the model with only the ℓ_1 loss is trained more stable but produces results far from being realistic. The combination of the two losses in Figure 1(e) helps not only reliably train the model but also generate visually appealing images; yet, it results in a lack of variation. The model completely ignores noise input and always outputs the same image for the segmentation map. This phenomenon is also reported in (Mathieu et al., 2016; Isola et al., 2017), although the cause is unknown. Pathak et al. (2016) and Iizuka et al. (2017) even state that better results are obtained without noise in the models. Finally, Figure 1(f) shows that our new objective enables the model to generate not only visually appealing but also diverse output samples.

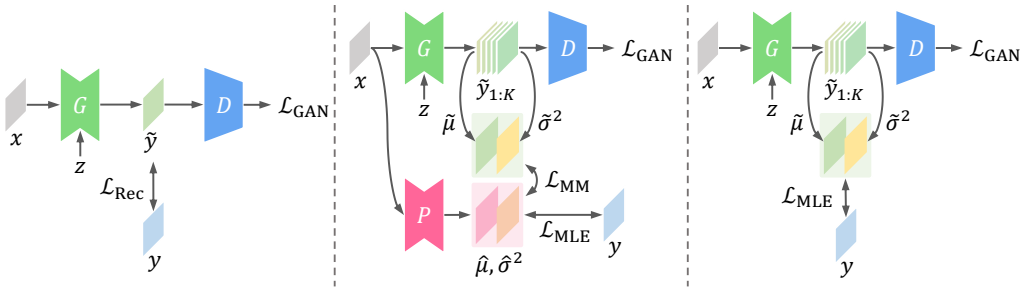
4 Approach

We propose novel alternatives of the reconstruction loss that are applicable to virtually any conditional generation tasks. Trained with our new loss terms, the model can accomplish both the stability of training and multimodal generation as already seen in Figure 1(d). Figure 2 illustrates architectural comparison between conventional conditional GANs and our models. In the conventional conditional GAN, the MLE losses are applied to the generator's objective to make sure that it generates an output sample well matched to its ground-truth. On the other hand, our key idea is to apply MLE losses to make sure that the *statistics*, such as mean or variance, of the conditional distribution $p(y|x)$ are similar between the generator's sample distribution and actual data distribution.

In section 4.1, we extend the ℓ_1/ℓ_2 loss to estimate all the parameters of Laplace and Gaussian distribution. In section 4.2 and 4.3, we present two novel training schemes for conditional GANs. Finally, we discuss our methods' novelties over existing algorithms in section 4.4.

4.1 The MLE for Mean and Variance

The ℓ_2 loss encourages the model to perform the MLE of the conditional mean of y given x while the variance σ^2, the other parameter of Gaussian, is assumed to be fixed. If we allow the model to



(a) Conditional GAN (b) MLMM (c) MCMLE

Figure 2: Architecture comparison of the proposed MLMM and MCMLE with conventional conditional GANs. (a) In a conditional GAN, G generates a sample $\tilde{y}$ (*e.g.* a realistic image) from an input x (*e.g.* a segmentation map), and D determines whether $\tilde{y}$ is real or fake. The reconstruction loss $\mathcal{L}_{\text{Rec}}$ enforces a sample $\tilde{y}$ to be similar to a ground-truth image y. (b) In our first method MLMM, G generates a set of samples $\tilde{y}_{1:K}$. The predictor P, the twin network of G, is trained with an MLE loss to estimate the conditional mean and variances of y given x. Then, we match the sample mean $\tilde{\mu}$ to predicted $\hat{\mu}$ and the sample variance $\tilde{\sigma}^2$ to predicted $\hat{\sigma}^2$. (c) The MCMLE is simpler than MLMM with no prediction, and directly calculates the MLE loss between $\tilde{\mu}$ and y.

estimate the conditional variance as well, the MLE loss corresponds to

$$\mathcal{L}_{\text{MLE,Gaussian}} = \mathbb{E}_{x,y\sim p_{\text{data}}}\left[\frac{(y-\hat{\mu})^2}{2\hat{\sigma}^2} + \frac{1}{2}\log\hat{\sigma}^2\right] \quad \text{where } \hat{\mu}, \hat{\sigma}^2 = f_{\boldsymbol{\theta}}(x). \tag{4}$$

where the model $f_{\boldsymbol{\theta}}$ now estimates both estimated $\hat{\mu}$ and $\hat{\sigma}^2$ for x. Estimating the conditional variance along with the mean can be interpreted as estimating the heteroscedastic aleatoric uncertainty in (Kendall & Gal, 2017) where the variance is the measure of aleatoric uncertainty.

For the Laplace distribution, we can derive the similar MLE loss as

$$\mathcal{L}_{\text{MLE,Laplace}} = \mathbb{E}_{x,y\sim p_{\text{data}}}\left[\frac{|y-\hat{m}|}{\hat{b}} + \log\hat{b}\right] \quad \text{where } \hat{m}, \hat{b} = f_{\boldsymbol{\theta}}(x) \tag{5}$$

where $\hat{m}$ is the predicted median and $\hat{b}$ is the predicted MAD. In practice, it is more numerically stable to predict the logarithm of variance or MAD (Kendall & Gal, 2017).

In the following, we will describe our methods mainly with the ℓ_2 loss under Gaussian assumption. It is straightforward to obtain Laplace versions of our methods for the ℓ_1 loss by simply replacing the mean and variance with the median and MAD.

4.2 Maximum Likelihood Moment Matching (MLMM)

Our first model is named as *Maximum Likelihood Moment Matching* (MLMM) whose overall architecture is depicted in Figure 2(b). Its architecture follows that of a conditional GAN, but there are two updates. First, the generator produces K different samples $\tilde{y}_{1:K}$ for each input x by varying noise input z. Second, we introduce a separate component *predictor*, which is a clone of generator with some minor differences: (i) no noise source as input, and (ii) both mean and variance prediction as output. The predictor uses the MLE loss in Eq.(4) with ground-truth y to obtain the predictions of conditional mean and variance, *i.e.* $\hat{\mu}$ and $\hat{\sigma}^2$. When training the generator, we utilize the predicted statistics of real distribution to guide the outputs of generator. Specifically, we match the predicted mean/variance and the mean/variance of the generator's distribution, which is computed by the sample mean $\tilde{\mu}$ and variance $\tilde{\sigma}^2$ from $\tilde{y}_{1:K}$. Then, we define the MLMM loss $\mathcal{L}_{\text{MLMM}}$ as the sum of squared errors between predicted statistics and sample statistics. The final loss becomes the weighted sum of GAN loss and MLMM loss. In summary,

$$\mathcal{L} = \mathcal{L}_{\text{GAN}} + \lambda_{\text{M}}\mathcal{L}_{\text{MLMM}} \quad \text{where } \mathcal{L}_{\text{MLMM}} = (\tilde{\mu}-\hat{\mu})^2 + (\tilde{\sigma}^2-\tilde{\sigma}^2)^2, \tag{6}$$

$$\hat{\mu}, \hat{\sigma}^2 = P(x); \quad \tilde{\mu} = \frac{1}{K}\sum_{i=1}^{K}\tilde{y}_i, \ \tilde{\sigma}^2 = \frac{1}{K-1}\sum_{i=1}^{K}(\tilde{y}_i-\tilde{\mu})^2, \quad \text{where } \tilde{y}_{1:K} = G(x, z_{1:K}). \tag{7}$$

One possible variant is to match only the first moment μ. We denote the original method $\text{MLMM}_{1/2}$ and the variant MLMM_1 where the number indicates the order of matched moments. In addition, we can easily derive the Laplace version of $\text{MLMM}_{1/2}$ that uses median and MAD.

4.3 Monte Carlo MLE

In MLMM, we use a surrogate entity, predictor, to predict the conditional moments and match with the sample moments. Another approach may be directly applying MLE losses to the sample statistics; since the generator is supposed to approximate the real distribution if optimal, we can consider the statistics of generated distribution as the estimators for the statistics of real distribution. Like MLMM, we can use the sample statistics $\tilde{\mu}$ and $\tilde{\sigma}^2$ as approximated statistics of generated distribution (in this case, it corresponds to Monte Carlo estimation). Then we replace $\hat{\mu}$ and $\hat{\sigma}^2$ in Eq.(4) with $\tilde{\mu}$ and $\tilde{\sigma}^2$ to obtain Monte Carlo MLE (MCMLE) loss:

$$\mathcal{L}_{\text{MCMLE}} = \mathbb{E}_{x,y\sim p_{\text{data}}}\left[\frac{(y-\tilde{\mu})^2}{2\tilde{\sigma}^2} + \frac{1}{2}\log\tilde{\sigma}^2\right], \tag{8}$$

where $\tilde{\mu}$ and $\tilde{\sigma}^2$ are defined as Eq.(7). The MCMLE architecture is illustrated in Figure 2(c). Similarly to MLMM, we can build a variant MCMLE_1, where the ℓ_2 loss only between y and $\tilde{\mu}$ is used. Also, Laplace assumption yields the Laplace versions of $\text{MCMLE}_{1/2}$. The detailed algorithms of all eight variants are presented in appendix A.

Compared to MLMM, MCMLE allows the generator to access real data y directly, thus there is no bias caused by the predictor. On the other hand, the use of predictor in MLMM provides less variance in target values, and leads more stable training especially when a batch or sample size is small. Another important aspect worth comparison is overfitting, which should be carefully considered when using MLE with finite training data. In MLMM, we can choose the predictor with the smallest validation loss to avoid overfitting, and freeze it while training the generator. Therefore, the generator trained with MLMM suffers relatively less from overfitting, compared to the generator trained with MCMLE which directly observes training data. To sum up, the two methods have their own pros and cons. We will empirically compare the behaviors of these two approaches in section 5.2.

4.4 Analyses on Our Approach

We discuss why our approach does not lose variability of output samples unlike existing models with reconstruction losses. Given that the mode collapse in GANs are not fully understood yet, we here prove that the joint use of the GAN loss with the ℓ_2 loss always leads a suboptimal generator, while our loss does not. We defer to appendix F for the ℓ_1 loss case.

In conditional generation tasks, we can assume that there always exists x such that the conditional variance of real data $\text{Var}(y|x) > 0$ (*i.e.* different outputs are possible for a given input). Training conditional GAN can be regarded as finding a generator with a mapping $G : (x, z) \rightarrow y$ that minimizes a given loss function. The sets of optimal generators for the GAN loss and the ℓ_2 loss, denoted by $\mathbb{G}$ and $\mathbb{R}$ respectively, can be formulated as follows:

$$\mathbb{G} = \{G|p_{\text{data}}(y|x) = p_G(y|x)\}, \quad \mathbb{R} = \{G|\arg\min_G \mathbb{E}_{x,y,z}[\mathcal{L}_2(G(x,z),y)]\} \tag{9}$$

According to the decomposition in section 3.2, ℓ_2 loss is minimized when both bias and variance are zero. Therefore, $\mathbb{R}$ is a subset of $\mathbb{V}$ where G has no variance as

$$\mathbb{R} \subset \mathbb{V} = \{G|\text{Var}_z(G(x,z)|x) = 0\}. \tag{10}$$

Since we initially assume that $\text{Var}(y|x) > 0$ for some x, we can conclude $\mathbb{G} \cap \mathbb{V} = \emptyset$, and thus $\mathbb{G} \cap \mathbb{R} = \emptyset$.

On the other hand, the sets of optimal mappings for our loss terms are formulated as follows:

$$\mathbb{M}_1 = \{G|\mathbb{E}_x[(\mathbb{E}_y[y|x] - \mathbb{E}_z[G(x,z)|x])^2] = 0\} \tag{11}$$

$$\mathbb{M}_2 = \{G|\mathbb{E}_x[(\mathbb{E}_y[y|x] - \mathbb{E}_z[G(x,z)|x])^2 + (\text{Var}_y(y|x) - \text{Var}_z(G(x,z)|x))^2] = 0\} \tag{12}$$

where $\mathbb{M}_1$ corresponds to MLMM_1 and MCMLE_1 while $\mathbb{M}_2$ corresponds to $\text{MLMM}_{1/2}$ and $\text{MCMLE}_{1/2}$. It is straightforward to show that $\mathbb{G} \subset \mathbb{M}_2 \subset \mathbb{M}_1$; if $p_{\text{data}}(y|x) = p_G(y|x)$ is satisfied at optimum, the conditional expectations and variations of both sides should be the same too:

$$\big[\mathbb{E}_y[y|x] = \mathbb{E}_z[G(x,z)|x]\big] \wedge \big[\text{Var}_y(y|x) = \text{Var}_z(G(x,z)|x)\big]. \tag{13}$$

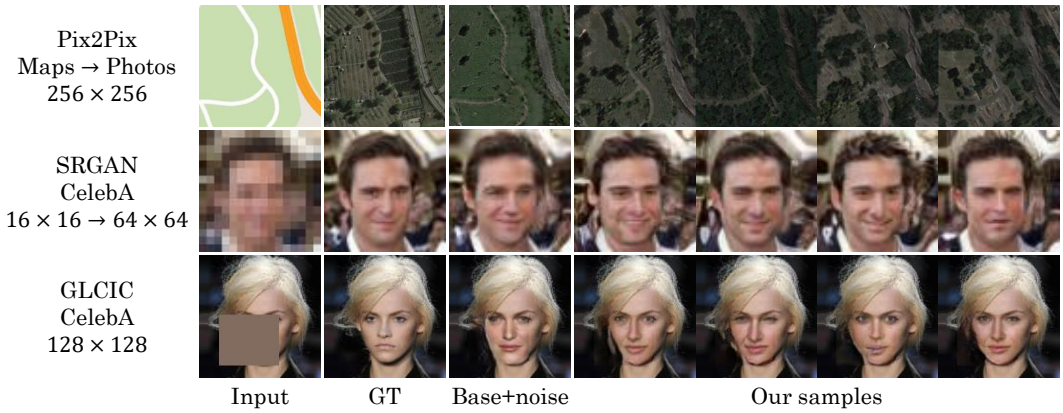


Figure 3: Comparison between the results of our method $\text{MMLM}_{1/2}$ and the state-of-the-art methods on image-to-image translation, super-resolution and image inpainting tasks. In every task, our model generates diverse images of high quality, while existing methods with the reconstruction loss do not.

To summarize, there is no generator that is both optimal for the GAN loss and the ℓ_2 loss since $\mathbb{G} \cap \mathbb{R} = \emptyset$. Thus, if both losses are jointly used as done in most existing algorithms, the final solution is sub-optimal. Moreover, as the ℓ_2 loss pushes the conditional variance to zero, the final solution is likely to lose multimodality. On the other hand, the optimal generator w.r.t. the GAN is also optimal w.r.t. our loss term since $\mathbb{G} \subset \mathbb{M}_2 \subset \mathbb{M}_1$.

This proof may not fully demonstrate why our model does not give up multimodality, which could be an interesting future work connected with the mode collapsing issue of GANs. Nonetheless, we can at least assert that our loss functions do not suffer from the side effect of the reconstruction loss.

5 Experiments

In order to show the generality of our methods, we apply them to three conditional generation tasks: image-to-image translation, super-resolution and image inpainting, for each of which we select as base models Pix2Pix, SRGAN (Ledig et al., 2017), and GLCIC (Iizuka et al., 2017), respectively. We use the Maps dataset (Isola et al., 2017) and Cityscapes dataset (Cordts et al., 2016) for image translation and the CelebA dataset (Liu et al., 2015) for the other tasks. We minimally modify the base models to include random noise source and train them with the $\text{MLMM}_{1/2}$ objective. We do not use any other loss terms such as perceptual loss, and the models are trained from scratch. We present all the training and implementation details and more thorough results in the appendix.

5.1 Qualitative Evaluation

In every task, most of our methods successfully generate diverse images as presented in Figure 3. Our methods are generally applicable to a wide variety of conditional generation tasks. From qualitative aspects, the most noticeable difference between MLMM and MCMLE lies in the training stability. We find that MLMM works well in all three tasks, while we cannot find any working configuration of MCMLE in the image inpainting task. Also, MCMLE is more sensitive to the number of output samples by the generator per input. In SRGAN experiments, for example, both methods converge reliably with the sampling number of 24, while MCMLE often diverges at the sampling number of 12. Although MCMLE is simpler and can be trained in an end-to-end manner, its applicability is rather limited compared to MLMM. We observe that the optimal setting varies greatly from task to task. We provide generated samples for each configuration in the appendix.

5.2 Quantitative Evaluation

As done in Zhu et al. (2017), we quantitatively measure diversity and realism of generated images. We evaluate our methods on Pix2Pix–Cityscapes, SRGAN–CelebA, and GLCIC–CelebA tasks. For each (method, task) pair, we generate 20 images from each of 300 different inputs using the trained

Method		Realism	Diversity
Random real data		1.00	0.559
Pix2Pix+noise		$0.22^{\pm 0.04}$	0.004
BicycleGAN		$0.16^{\pm 0.03}$	0.191
Gaussian	MLMM_1	$0.49^{\pm 0.05}$	**0.519**
	$\text{MLMM}_{1/2}$	$0.44^{\pm 0.05}$	0.388
	MCMLE_1	$\mathbf{0.54}^{\pm 0.05}$	0.299
	$\text{MCMLE}_{1/2}$	$0.14^{\pm 0.02}$	0.453
Laplace	MLMM_1	$0.19^{\pm 0.03}$	0.368
	$\text{MLMM}_{1/2}$	$0.17^{\pm 0.02}$	0.380
	MCMLE_1	$0.19^{\pm 0.03}$	0.393
	$\text{MCMLE}_{1/2}$	$0.20^{\pm 0.04}$	0.384

(a) Pix2Pix–Cityscapes

Method	Realism	Diversity
Random real data	1.00	0.290
SRGAN+noise	$\mathbf{0.60}^{\pm 0.06}$	0.005
Gaussian $\text{MLMM}_{1/2}$	$0.52^{\pm 0.05}$	**0.050**

(b) SRGAN–CelebA

Method	Realism	Diversity
Random real data	1.00	0.426
GLCIC+noise	$0.28^{\pm 0.04}$	0.004
Gaussian $\text{MLMM}_{1/2}$	$\mathbf{0.33}^{\pm 0.04}$	**0.060**

(c) GLCIC–CelebA

Table 1: Quantitative evaluation on three (method, dataset) pairs. Realism is measured by $2(1-F)$ where F is the average of F-measures of identifying fake by human evaluators. Diversity is scored by the average of conditional LPIPS values. In both metrics, higher are better. In all three tasks, our methods generate highly diverse images with competitive or even better realism.

model. As a result, the test sample set size is 6,000 in total. For diversity, we measure the average LPIPS score (Zhang et al., 2018). Among 20 generated images per input, we randomly choose 10 pairs of images and compute conditional LPIPS values. We then average the scores over the test set. For realism, we conduct a human evaluation experiment from 33 participants. We present a real or fake image one at a time and ask participants to tag whether it is real or fake. The images are presented for 1 second for SRGAN/GLCIC and 0.5 second for Pix2Pix, which are similar to (Zhu et al., 2017). We calculate the accuracy of identifying fake images with averaged F-measure F and use $2(1-F)$ as the realism score. The score is assigned to 1.0 when all samples are completely indistinguishable from real images and 0 when evaluators make no misclassification.

There are eight configurations of our methods in total, depending on the MLE (Gaussian and Laplace), the number of statistics (one and two) and the method type (MLMM and MCMLE). We test all variants for the Pix2Pix task and only Gaussian $\text{MLMM}_{1/2}$ for the others. We compare with base models in every task and additionally BicycleGAN for the Pix2Pix task. We use the official BicycleGAN implementation from the authors with minimal modification. Table 1 summarizes the results. In terms of both realism and diversity, our methods achieve competitive performance compared to the base models, sometimes even better. For instance, in Pix2Pix–Cityscapes setting, three of our methods, Gaussian MLMM_1, $\text{MLMM}_{1/2}$, and MCMLE_1, significantly outperform BicycleGAN and Pix2Pix+noise in both measures. Without exception, the diversity scores of our methods are far greater than those of the baselines while maintaining competitive realism scores. These results confirm that our methods generate a broad spectrum of quality images from a single input.

6 Conclusion

In this work, we pointed out that there is a significant mismatch between conventional reconstruction losses and GAN loss which leads to suboptimal generators. As alternatives, we proposed a set of novel loss functions named MLMM and MCMLE that enable conditional GAN models to accomplish both the stability of training and multimodal generation. Empirically, we showed that our loss functions were successfully integrated with multiple state-of-the-art models for image translation, super-resolution, and image inpainting tasks, for which our method generated realistic image samples of high visual fidelity and variability on Cityscapes and CelebA datasets.

There are numerous possible directions beyond this work. First, there are other conditional generation tasks that we did not cover, such as text-to-image synthesis, text-to-speech synthesis, and video prediction, for which our methods can be directly applied to generate diverse, high-quality samples. Second, in terms of statistics matching, our methods can be extended to explore other higher order statistics or covariance. Third, using the statistics of high-level features may capture correlations which are unreachable with pixel-level statistics.

REFERENCES

Jianmin Bao, Dong Chen, Fang Wen, Houqiang Li, and Gang Hua. CVAE-GAN: Fine-Grained image generation through asymmetric training. *ICCV*, 2017.

Prateep Bhattacharjee and Sukhendu Das. Temporal coherency based criteria for predicting video frames using deep multi-stage generative adversarial networks. *NIPS*, 2017.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Joan Bruna, Pablo Sprechmann, and LeCun, Yann. Super-Resolution with deep convolutional sufficient statistics. *ICLR*, 2016.

Adrian Bulat, Jing Yang, and Georgios Tzimiropoulos. To learn image super-resolution, use a GAN to learn how to do image degradation first. *ECCV*, 2018.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CVPR*, 2016.

Ian Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *arXiv*, 2016.

Ian Goodfellow, Pouget-Abadie, Jean, Mehdi Mirza, Bing Xu, Warde-Farley, David, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *NIPS*, 2014.

Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. Multimodal unsupervised Image-to-Image translation. *ECCV*, 2018.

Satoshi Iizuka, Simo-Serra, Edgar, and Hiroshi Ishikawa. Globally and locally consistent image completion. *SIGGRAPH*, 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-Image translation with conditional adversarial networks. *CVPR*, 2017.

Gareth James. Variance and bias for general loss functions. *Machine Learning*, 2003.

Yunseok Jang, Gunhee Kim, and Yale Song. Video prediction with appearance and motion conditions. *ICML*, 2018.

Justin Johnson, Alexandre Alahi, and Fei-Fei, Li. Perceptual losses for Real-Time style transfer and Super-Resolution. *ECCV*, 2016.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *NIPS*, 2017.

Diederik P. Kingma and Max Welling. Auto-Encoding variational bayes. *ICLR*, 2014.

Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. *ICML*, 2016.

Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic single image Super-Resolution using a generative adversarial network. *CVPR*, 2017.

Alex Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv*, 2018a.

Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse Image-to-Image translation via disentangled representations. *ECCV*, 2018b.

Yijun Li, Sifei Liu, Jimei Yang, and Ming-Hsuan Yang. Generative face completion. *CVPR*, 2017.

Yifan Liu, Zengchang Qin, Zhenbo Luo, and Hua Wang. Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks. *arXiv*, 2017.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. *ICCV*, 2015.

Chaochao Lu, Michael Hirsch, and Bernhard Schölkopf. Flexible Spatio-Temporal networks for video prediction. *CVPR*, 2017.

Pengyuan Lyu, Xiang Bai, Cong Yao, Zhen Zhu, Tengteng Huang, and Wenyu Liu. Auto-Encoder guided GAN for chinese calligraphy synthesis. *ICDAR*, 2017.

Dwarikanath Mahapatra, Behzad Bozorgtabar, Sajini Hewavitharanage, and Rahil Garnavi. Image super resolution using generative adversarial networks and local saliency maps for retinal image analysis. *MICCAI*, 2017.

Michael Mathieu, Camille Couprie, and LeCun, Yann. Deep multi-scale video prediction beyond mean square error. *ICLR*, 2016.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv*, 2014.

Kyle Olszewski, Zimo Li, Chao Yang, Yi Zhou, Ronald Yu, Zeng Huang, Sitao Xiang, Shunsuke Saito, Pushmeet Kohli, and Hao Li. Realistic dynamic facial textures from a single image using GANs. *ICCV*, 2017.

Xi Ouyang, Yu Cheng, Yifan Jiang, Chun-Liang Li, and Pan Zhou. Pedestrian-Synthesis-GAN: generating pedestrian data in real scene and beyond. *arXiv preprint arXiv* , 2018.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. *CVPR*, 2016.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *ICLR*, 2018.

Mark Sabini and Gili Rusak. Painting outside the box: Image outpainting with GANs. *arXiv*, 2018.

Mehdi S. M. Sajjadi, Bernhard B Sch, and Michael Hirsch. EnhanceNet: single image Super-Resolution through automated texture synthesis. *ICCV*, 2017.

Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Scribbler: Controlling deep image synthesis with sketch and color. *CVPR*, 2017.

Yuhang Song, Chao Yang, Yejin Shen, Peng Wang, Qin Huang, and C.-C. Jay Kuo. SPG-Net: segmentation prediction and guidance network for image inpainting. *arXiv preprint arXiv* , 2018.

Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *ICLR*, 2017.

Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. *CVPR*, 2017.

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. *arXiv*, 2017.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Guilin Liu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. Video-to-Video synthesis. *arXiv*, 2018.

Wenqi Xian, Patsorn Sangkloy, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. TextureGAN: controlling deep image synthesis with texture patches. *arXiv* , 2017.

You Xie, Erik Franz, Mengyu Chu, and Nils Thuerey. tempoGAN: a temporally coherent, volumetric GAN for super-resolution fluid flow. *SIGGRAPH*, 2018.

Xiangyu Xu, Deqing Sun, Jinshan Pan, Yujin Zhang, Hanspeter Pfister, and Ming-Hsuan Yang. Learning to Super-Resolve blurry face and text images. *ICCV*, 2017.

Chao Yang, Yuhang Song, Xiaofeng Liu, Qingming Tang, and C.-C. Jay Kuo. Image inpainting using block-wise procedural training with annealed adversarial counterpart. *arXiv preprint arXiv* , 2018.

Raymond A. Yeh, Chen Chen, Teck Yian Lim, Alexander G. Schwing, Hasegawa-Johnson, Mark, and Minh N. Do. Semantic image inpainting with deep generative models. *CVPR*, 2017.

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas Huang. Free-Form image inpainting with gated convolution. *arXiv*, 2018.

He Zhang, Vishwanath Sindagi, and Vishal Patel. Image de-raining using a conditional generative adversarial network. *arXiv*, 2017a.

Richard Zhang, Phillip Isola, Alexei Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *CVPR*, 2018.

Shu Zhang, Ran He, and Tieniu Tan. DeMeshNet: blind face inpainting for deep MeshFace verification. *TIFS*, 2017b.

Yipin Zhou and Tamara L. Berg. Learning temporal transformations from Time-Lapse videos. *ECCV*, 2016.

Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A. Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. *NIPS*, 2017.

A ALGORITHMS

We elaborate the algorithms of all eight variants of our methods in detail from Algorithm 1 to Algorithm 8. The presented algorithms assume a single input per update, although we use mini-batch training in practice. Also we use non-saturating GAN loss, $-\log D(x, \tilde{y})$ (Goodfellow, 2016).

For Laplace MLEs, the statistics that we compute are median and MAD. Unlike mean, however, the gradient of median is defined only in terms of the single median sample. Therefore, a naive implementation would simply calculate the gradient only for the median sample, which is not effective for training. Therefore, we use a special trick to distribute the gradients to every sample. In MLMM, we first calculate the difference between the predicted median and the sample median, and then add it to samples $\tilde{y}_{1:K}$ to set the target values $t_{1:K}$. We consider $t_{1:K}$ as constants so that the gradient is not calculated for the target values (this is equivalent to `Tensor.detach()` in PyTorch and `tf.stop_gradient()` in TensorFlow). Finally, we calculate the loss between the target values and samples, not the medians. We use the similar trick for MCMLE.

Algorithm 1 Generator update in Gaussian MLMM_1

Require: Generator G, discriminator D, pre-trained predictor P, MLMM loss coefficient λ_{MLMM}
Require: input x, ground truth y, the number of samples K
1: **for** $i = 1$ **to** K **do**
2: $\quad z_i \leftarrow \text{GenerateNoise}()$
3: $\quad \tilde{y}_i \leftarrow G(x, z_i)$ {Generate K samples}
4: **end for**
5: $\hat{\mu} \leftarrow P(x)$ {Predicted mean}
6: $\tilde{\mu} \leftarrow \frac{1}{K}\sum_{i=1}^{K} \tilde{y}_i$ {Sample mean}
7: $\mathcal{L}_{\text{MLMM}} \leftarrow (\hat{\mu} - \tilde{\mu})^2$
8: $\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} -\log D(x, y_i)$
9: $\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G}\mathcal{L}_{\text{GAN}} + \lambda_{\text{MLMM}}\mathcal{L}_{\text{MLMM}})$

Algorithm 2 Generator update in Gaussian $\text{MLMM}_{1/2}$

Require: Generator G, discriminator D, pre-trained predictor P, MLMM loss coefficient λ_{MLMM}
Require: input x, ground truth y, the number of samples K
1: **for** $i = 1$ **to** K **do**
2: $\quad z_i \leftarrow \text{GenerateNoise}()$
3: $\quad \tilde{y}_i \leftarrow G(x, z_i)$ {Generate K samples}
4: **end for**
5: $\hat{\mu}, \hat{\sigma}^2 \leftarrow P(x)$ {Predicted mean and variance}
6: $\tilde{\mu} \leftarrow \frac{1}{K}\sum_{i=1}^{K} \tilde{y}_i$ {Sample mean}
7: $\tilde{\sigma}^2 \leftarrow \frac{1}{K-1}\sum_{i=1}^{K} (\tilde{y}_i - \tilde{\mu})^2$ {Sample variance}
8: $\mathcal{L}_{\text{MLMM}} \leftarrow (\hat{\mu} - \tilde{\mu})^2 + (\hat{\sigma}^2 - \tilde{\sigma}^2)^2$
9: $\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} -\log D(x, y_i)$
10: $\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G}\mathcal{L}_{\text{GAN}} + \lambda_{\text{MLMM}}\mathcal{L}_{\text{MLMM}})$

Algorithm 3 Generator update in Laplace MLMM_1

Require: Generator G, discriminator D, pre-trained predictor P, MLMM loss coefficient λ_{MLMM}
Require: input x, ground truth y, the number of samples K
for $i = 1$ **to** K **do**
  $z_i \leftarrow \text{GenerateNoise}()$
  $\tilde{y}_i \leftarrow G(x, z_i)$ {Generate K samples}
end for
$\hat{m} \leftarrow P(x)$ {Predicted median}
$\tilde{m} \leftarrow \text{med}(\tilde{y}_{1:K})$ {Sample median}
$t_{1:K} = \text{detach}(\tilde{y}_{1:K} + (\hat{m} - \tilde{m}))$
$\mathcal{L}_{\text{MLMM}} \leftarrow \frac{1}{K}\sum_{i=1}^{K}(t_i - \tilde{y}_i)^2$
$\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} -\log D(x, y_i)$
$\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G}\mathcal{L}_{\text{GAN}} + \lambda_{\text{MLMM}}\mathcal{L}_{\text{MLMM}})$

Algorithm 4 Generator update in Laplace $\text{MLMM}_{1/2}$

Require: Generator G, discriminator D, pre-trained predictor P, MLMM loss coefficient λ_{MLMM}
Require: input x, ground truth y, the number of samples K
for $i = 1$ **to** K **do**
  $z_i \leftarrow \text{GenerateNoise}()$
  $\tilde{y}_i \leftarrow G(x, z_i)$ {Generate K samples}
end for
$\hat{m}, \hat{b} \leftarrow P(x)$ {Predicted median and MAD}
$\tilde{m} \leftarrow \text{med}(\tilde{y}_{1:K})$ {Sample median}
$\tilde{b} \leftarrow \frac{1}{K}\sum_{i=1}^{K}|\tilde{y}_i - \tilde{m}|$ {Sample MAD}
$t_{1:K} = \text{detach}(\tilde{y}_{1:K} + (\hat{m} - \tilde{m}))$
$\mathcal{L}_{\text{MLMM}} \leftarrow \frac{1}{K}\sum_{i=1}^{K}(t_i - \tilde{y}_i)^2 + (\hat{b} - \tilde{b})^2$
$\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} -\log D(x, y_i)$
$\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G}\mathcal{L}_{\text{GAN}} + \lambda_{\text{MLMM}}\mathcal{L}_{\text{MLMM}})$

Algorithm 5 Generator update in Gaussian MCMLE_1

Require: Generator G, discriminator D, MCMLE loss coefficient λ_{MCMLE}
Require: input x, ground truth y, the number of samples K
for $i = 1$ **to** K **do**
  $z_i \leftarrow \text{GenerateNoise}()$
  $\tilde{y}_i \leftarrow G(x, z_i)$ {Generate K samples}
end for
$\tilde{\mu} \leftarrow \frac{1}{K}\sum_{i=1}^{K}\tilde{y}_i$ {Sample mean}
$\mathcal{L}_{\text{MCMLE}} \leftarrow (y - \tilde{\mu})^2$
$\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} -\log D(x, y_i)$
$\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G}\mathcal{L}_{\text{GAN}} + \lambda\mathcal{L}_{\text{MCMLE}})$

Algorithm 6 Generator update in Gaussian $\text{MCMLE}_{1/2}$

Require: Generator G, discriminator D, MCMLE loss coefficient λ_{MCMLE}
Require: input x, ground truth y, the number of samples K
for $i = 1$ **to** K **do**
  $z_i \leftarrow \text{GenerateNoise}()$
  $\tilde{y}_i \leftarrow G(x, z_i)$ {Generate K samples}
end for
$\tilde{\mu} \leftarrow \frac{1}{K}\sum_{i=1}^{K}\tilde{y}_i$ {Sample mean}
$\tilde{\sigma}^2 \leftarrow \frac{1}{K-1}\sum_{i=1}^{K}(\tilde{y}_i - \tilde{\mu})^2$ {Sample variance}
$\mathcal{L}_{\text{MCMLE}} \leftarrow \frac{(y-\tilde{\mu})^2}{2\tilde{\sigma}^2} + \frac{1}{2}\log\tilde{\sigma}^2$
$\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} -\log D(x, y_i)$
$\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G}\mathcal{L}_{\text{GAN}} + \lambda\mathcal{L}_{\text{MCMLE}})$

Algorithm 7 Generator update in Laplace MCMLE$_1$

Require: Generator G, discriminator D, MCMLE loss coefficient λ_{MCMLE}
Require: input x, ground truth y, the number of samples K
for $i = 1$ **to** K **do**
  $z_i \leftarrow \mathrm{GenerateNoise}()$
  $\tilde{y}_i \leftarrow G(x, z_i)$ {Generate K samples}
end for
$\tilde{m} \leftarrow \mathrm{med}(\tilde{y}_{1:K})$ {Sample median}
$t_{1:K} = \mathrm{detach}(\tilde{y}_{1:K} + (y - \tilde{m}))$
$\mathcal{L}_{\mathrm{MCMLE}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} |t_i - \tilde{y}_i|$
$\mathcal{L}_{\mathrm{GAN}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} -\log D(x, y_i)$
$\theta_G \leftarrow \mathrm{Optimize}(\theta_G, \nabla_{\theta_G}\mathcal{L}_{\mathrm{GAN}} + \lambda\mathcal{L}_{\mathrm{MCMLE}})$

Algorithm 8 Generator update in Laplace MCMLE$_{1/2}$

Require: Generator G, discriminator D, MCMLE loss coefficient λ_{MCMLE}
Require: input x, ground truth y, the number of samples K
for $i = 1$ **to** K **do**
  $z_i \leftarrow \mathrm{GenerateNoise}()$
  $\tilde{y}_i \leftarrow G(x, z_i)$ {Generate K samples}
end for
$\tilde{m} \leftarrow \mathrm{med}(\tilde{y}_{1:K})$ {Sample median}
$\tilde{b} \leftarrow \frac{1}{K}\sum_{i=1}^{K} |\tilde{y}_i - \tilde{m}|$ {Sample MAD}
$t_{1:K} = \mathrm{detach}(\tilde{y}_{1:K} + (y - \tilde{m}))$
$\mathcal{L}_{\mathrm{MCMLE}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} \frac{|t_i - \tilde{y}_i|}{\tilde{b}} + \log \tilde{b}$
$\mathcal{L}_{\mathrm{GAN}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} -\log D(x, y_i)$
$\theta_G \leftarrow \mathrm{Optimize}(\theta_G, \nabla_{\theta_G}\mathcal{L}_{\mathrm{GAN}} + \lambda\mathcal{L}_{\mathrm{MCMLE}})$

B Training Details

B.1 Common Configurations

We use PyTorch for the implementation of our methods. In every experiment, we use AMSGrad optimizer (Reddi et al., 2018) with LR $= 10^{-4}, \beta_1 = 0.5, \beta_2 = 0.999$. We use the weight decay of a rate 10^{-4} and the gradient clipping by a value 0.5. In case of MLMM, we traine the predictor until it is overfitted, and use the checkpoint with the lowest validation loss. The weight of GAN loss is fixed to 1 in all cases.

B.2 Pix2Pix

Our Pix2Pix variant is based on the U-net generator from
`https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix`.

- Noise input: We concatenate Gaussian noise tensors of size $H \times W \times 32$ at the 1×1, 2×2, 4×4 feature map of the decoder. Each element in the noise tensors are independently sampled from $\mathcal{N}(0, 1)$.
- Input normalization: We normalize the inputs so that each channel has a zero-mean and a unit variance.
- Batch sizes: We use 16 for the discriminator and the predictor and 8 for the generator. When training the generator, we generate 10 samples for each input, therefore its total batch size is 80.
- Loss weights: We set $\lambda_{\mathrm{MLMM}} = \lambda_{\mathrm{MCMLE}} = 10$. For the baseline, we use ℓ_1 loss as the reconstruction loss and set $\lambda_{\ell_1} = 100$.
- Update ratio: We update generator once per every discriminator update.

B.3 SRGAN

Our SRGAN variant is based on the PyTorch implementation of SRGAN from `https://github.com/zijundeng/SRGAN`.

- Noise input: We concatenate Gaussian noise tensor of size $H \times W \times 16$ at each input of the residual blocks of the generator, except for the first and last convolution layers. Each element in the noise tensors are independently sampled from $\mathcal{N}(0, 1)$.
- Input normalization: We make $16 \times 16 \times 3$ input images' pixel values lie between -1 and 1. We do not further normalize them with their mean and standard deviation.
- Batch sizes: We use 32 for the discriminator and the predictor and 8 for the generator. When training the generator, we generate 24 samples for each input, and thus its total batch size is 192.
- Loss weights: We set $\lambda_{\mathrm{MLMM}} = 2400$ and $\lambda_{\mathrm{MCMLE}} = 20$. For the baseline, we use ℓ_2 loss as the reconstruction loss and set $\lambda_{\ell_2} = 1000$.
- Update ratio: We update generator five times per every discriminator update.

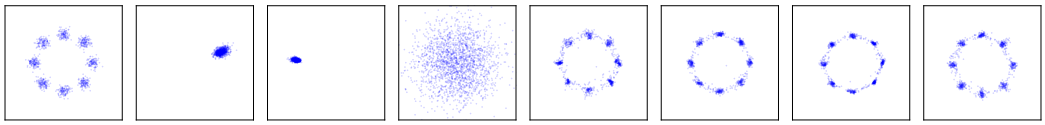
B.4 GLCIC

We built our own PyTorch implementation of the GLCIC model.

- Noise input: We concatenate Gaussian noise tensor of size $H \times W \times 32$ at each input of the first and second dilated convolution layers. We also inject the noise to the convolution layer before the first dilated convolution layer. Each element in the noise tensors are independently sampled from $\mathcal{N}(0, 1)$.
- Input resizing and masking: We use square-cropped CelebA images and resize them to 128×128. For masking, we randomly generate a hole of size between 48 and 64 and fill it with the average pixel value of the entire training dataset.
- Input normalization: We make $128 \times 128 \times 3$ input images' pixel values lie between 0 and 1. We do not further normalize them with their mean and standard deviation.
- Batch sizes: We use 16 for the discriminator and the predictor and 8 for the generator. When training the generator, we generate 12 samples for each input, therefore its total batch size is 96.
- Loss weights: For GLCIC, we tested Gaussian $\mathrm{MLMM}_{1/2}$ and MCMLE methods. We successfully trained GLCIC with Gaussian $\mathrm{MLMM}_{1/2}$ using $\lambda_{\mathrm{MLMM}} = 1000$. However, we could not find any working setting for MCMLEs. For the baseline model, we use ℓ_2 loss for the reconstruction loss and set $\lambda_{\ell_2} = 100$.
- Update ratio: We update generator three times per every discriminator update.

C PREVENTIVE EFFECTS FOR MODE COLLAPSE

Our MLMM and MCMLE methods have preventive effect on mode collapse. Figure 4 shows toy experiments of unconditional generation on synthetic 2D data which is hard to learn with GANs due to mode collapse. We train a simple 3-layer MLP with different objectives. When trained only with GAN loss, the model captures only one mode as shown in figure 4b. Adding ℓ_2 loss cannot fix this issue either as in figure 4c. In contrast, all four of our methods (figure 4e-4h) prevent mode collapse and successfully capture all eight modes. Notice that even the simpler variants MLMM_1 and MCMLE_1 effectively keep the model from mode collapse. Intuitively, if the generated samples are biased toward a single mode, their statistics, *e.g.* mean or variance, deviate from real statistics. Our methods penalize such deviations, thereby reducing mode collapse significantly. Although we restrict the scope of this paper to conditional generation tasks, these toy experiments shows that our methods has a potential to mitigate mode collapse and stabilize training even for unconditional generation tasks.



(a) Data (b) GAN (c) GAN+ℓ_2 (d) MLE (e) GAN + MLMM_1 (f) GAN + $\text{MLMM}_{1/2}$ (g) GAN + MCMLE_1 (h) GAN + $\text{MCMLE}_{1/2}$

Figure 4: Experiments on a synthetic 2D dataset. (a) The data distribution. (b)(c) Using GAN loss alone or with ℓ_2 loss results in mode collapse. (d) Training a predictor with the MLE loss in Eq.(4) produces a Gaussian distribution with the mean and variance close to real distribution. The dots are samples from the Gaussian distribution parameterized by the outputs of the predictor. (e)-(h) Generators trained with our methods successfully capture all eight modes.

D GENERATED SAMPLES

D.1 IMAGE-TO-IMAGE TRANSLATION (PIX2PIX)

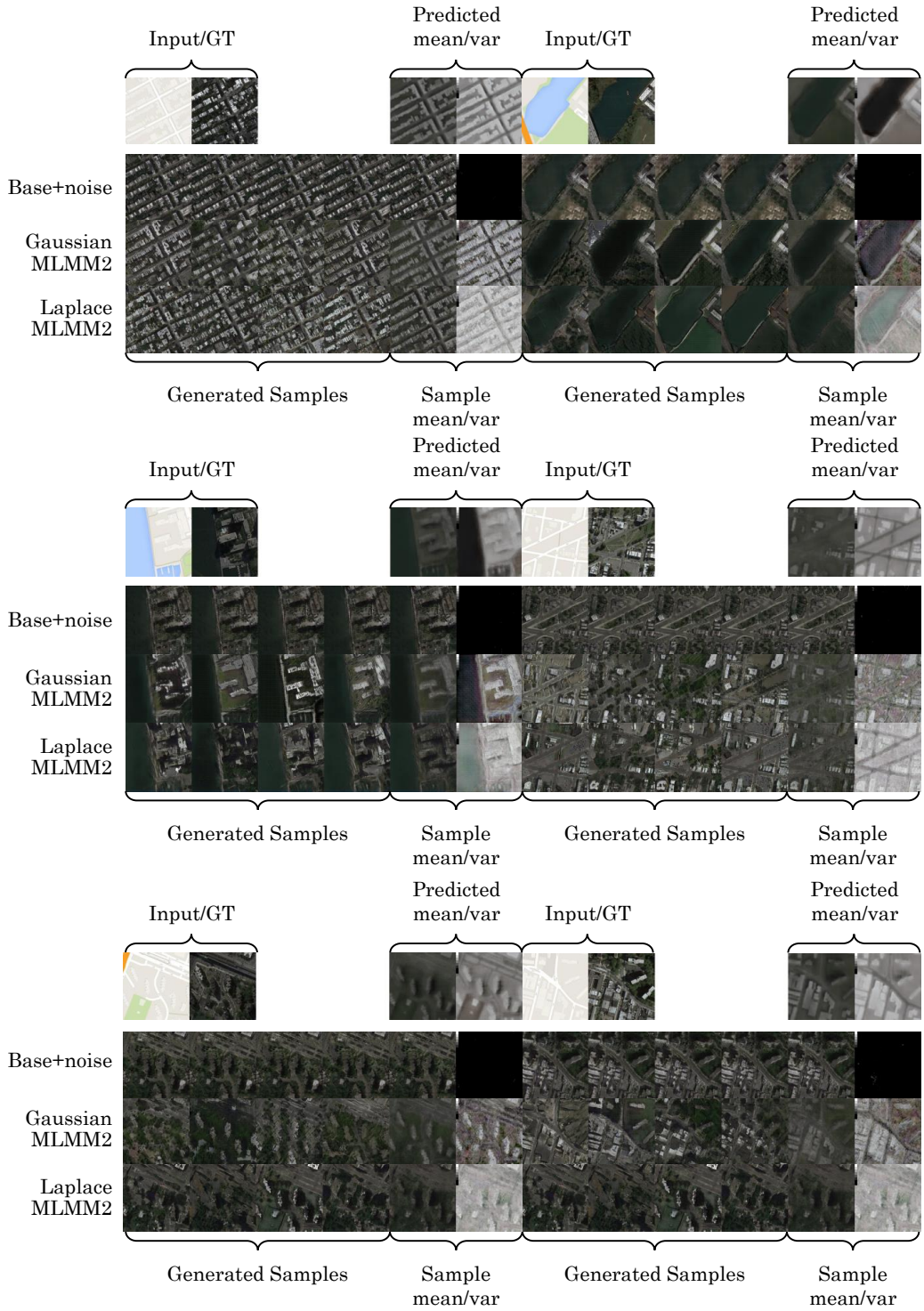


Figure 5: Comparison of our methods in Pix2Pix–Maps

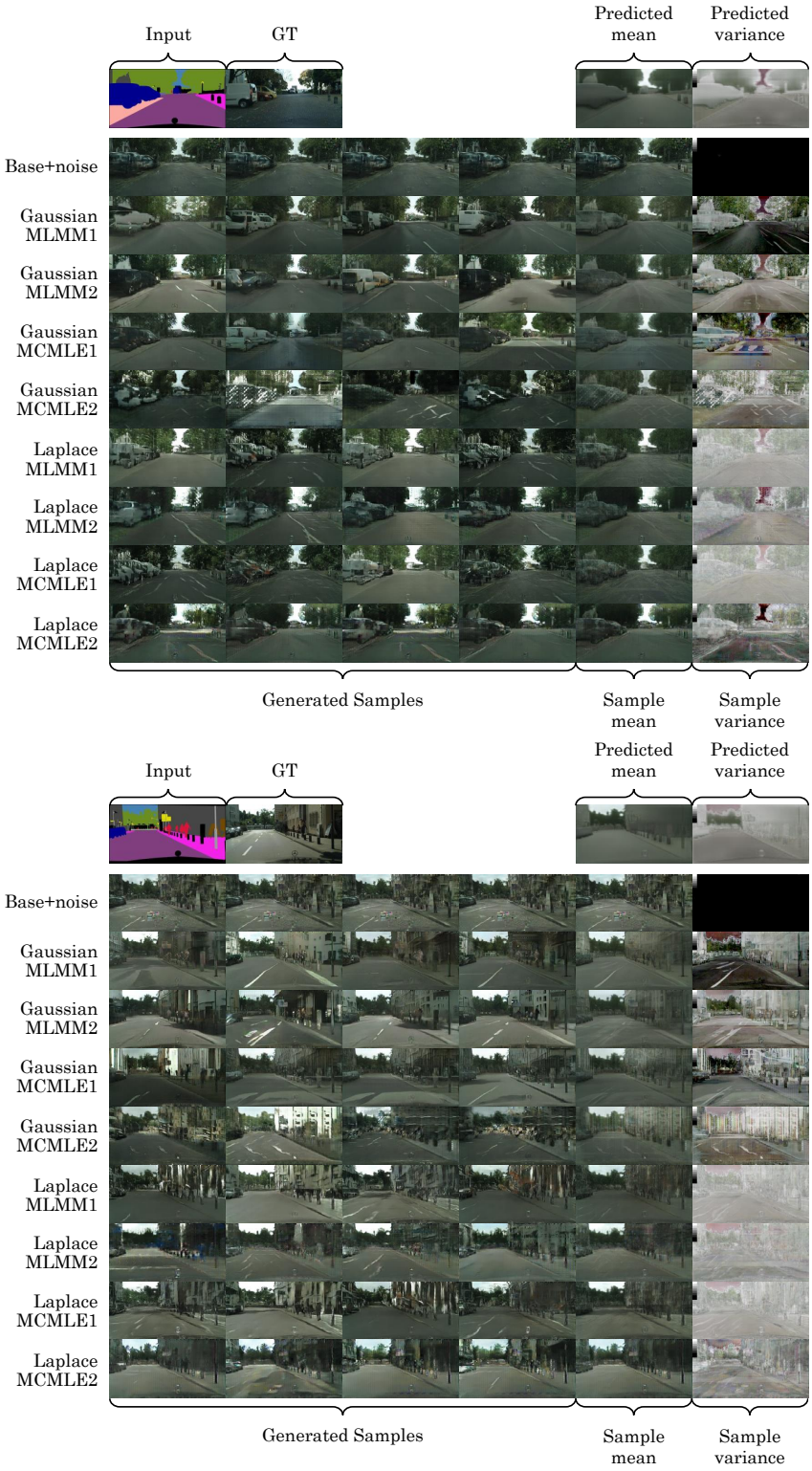


Figure 6: Comparison of our methods in Pix2Pix–Cityscapes

D.2 SUPER-RESOLUTION (SRGAN)

The first rows of following images are composed of input, ground-truth, predicted mean, sample mean, predicted variance, and sample variance. The other rows are generated samples.

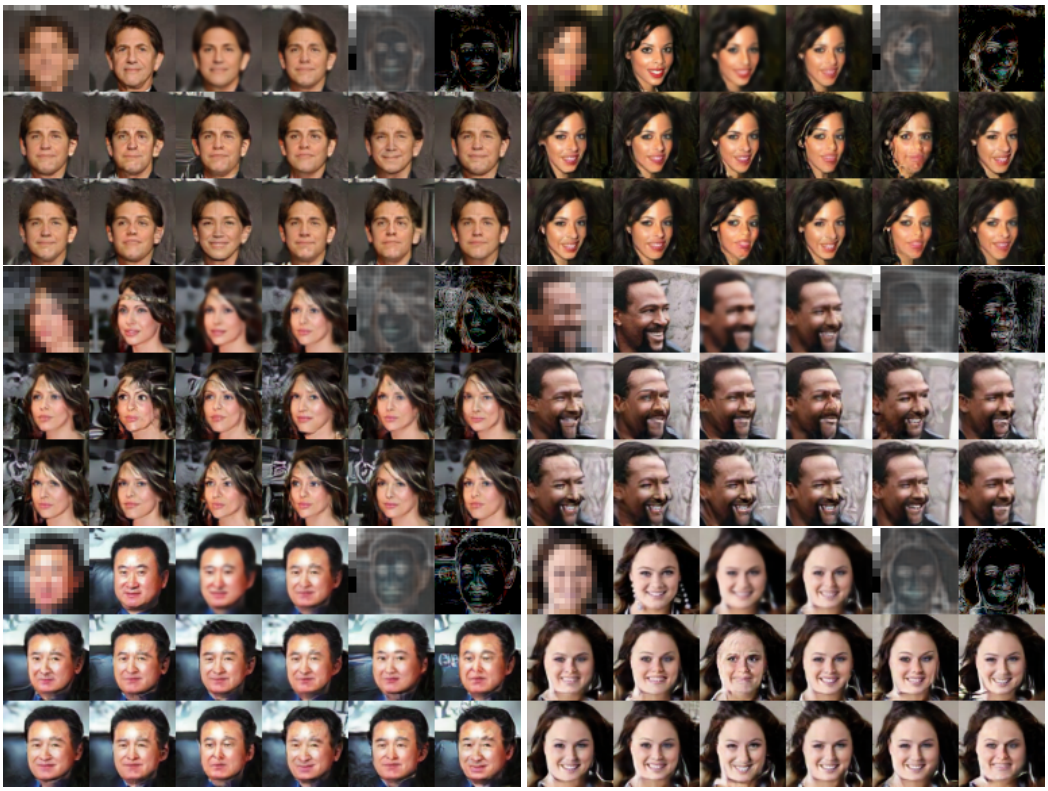


Figure 7: SRGAN–CelebA Gaussian $\text{MLMM}_{1/2}$ (success cases).

Figure 8: SRGAN–CelebA Gaussian $\text{MLMM}_{1/2}$ (failure cases).

D.3 IMAGE INPAINTING (GLCIC)

This section shows the results of GLCIC–CelebA task. The images are shown in the same manner as the previous section.

Figure 9: GLCIC–CelebA Gaussian $\text{MLMM}_{1/2}$ (success cases).

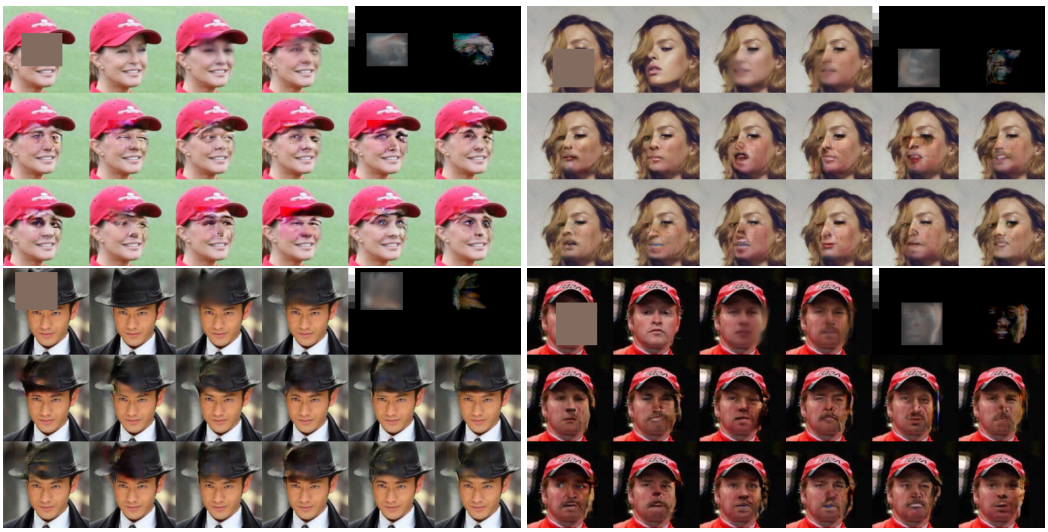


Figure 10: GLCIC–CelebA Gaussian $\text{MLMM}_{1/2}$ (failure cases).

E DECOMPOSITION OF RECONSTRUCTION LOSS

According to James (2003), ℓ_1 and ℓ_2 loss can be decomposed into following form.

$$\mathbb{E}_{y,\hat{y}}[\mathcal{L}_s(y,\hat{y})] = \underbrace{\mathbb{E}_y\big[\mathcal{L}_s(y,Sy])\big]}_{\mathrm{Var}(y)} + \underbrace{\mathbb{E}_y\big[\mathcal{L}_s(y,S\hat{y}) - \mathcal{L}_s(y,Sy)\big]}_{\mathrm{SE}(\hat{y},y)} \tag{14}$$

$$+ \underbrace{\mathbb{E}_{y,\hat{y}}\big[\mathcal{L}_s(y,\hat{y}) - \mathcal{L}_s(y,S\hat{y})\big]}_{\mathrm{VE}(\hat{y},y)}, \tag{15}$$

S is an operator that is defined to be

$$Sy = \arg\min_{\mu} \mathbb{E}_y[\mathcal{L}_s(y,\mu)].$$

Notice that the total loss is minimized when $\hat{y} = S\hat{y} = Sy$ reducing both SE and VE to 0. For ℓ_2 loss, Sy and $S\hat{y}$ are the expectations of y and $\hat{y}$.

$$Sy = \arg\min_{\mu} \mathcal{L}_2(y,\mu) = \mathbb{E}_y[y], S\hat{y} = \arg\min_{\mu} \mathcal{L}_2(\hat{y},\mu) = \mathbb{E}_{\hat{y}}[\hat{y}]$$

$$\begin{aligned}\mathrm{SE}(\hat{y},y) &= \mathbb{E}_y\big[\mathcal{L}_2(y,S\hat{y}) - \mathcal{L}_2(y,Sy)\big] = \mathbb{E}_y\big[(y-S\hat{y})^2 - (y-Sy)^2\big] \\ &= \mathbb{E}_y\big[(S\hat{y}+Sy-2y)(S\hat{y}-Sy)\big] = (S\hat{y}-Sy)^2,\end{aligned} \tag{16}$$

$$\begin{aligned}\mathrm{VE}(\hat{y},y) &= \mathbb{E}_{y,\hat{y}}\big[\mathcal{L}_2(y,\hat{y}) - \mathcal{L}_2(y,s\hat{y})\big] = \mathbb{E}_{y,\hat{y}}\big[(y-\hat{y})^2 - (y-S\hat{y})^2\big] \\ &= \mathbb{E}_{y,\hat{y}}\big[2(S\hat{y}-y)(\hat{y}-S\hat{y}) + (\hat{y}-S\hat{y})^2\big] = \mathbb{E}_{\hat{y}}\big[(\hat{y}-S\hat{y})^2\big]\end{aligned} \tag{17}$$

This decomposition offers another interpretation of the ℓ_2 loss: minimizing the bias and the variance of prediction.

F MISMATCH BETWEEN ℓ_1 LOSS AND GAN LOSS

In section 4.4, we showed that if there exists some x such that $\mathrm{Var}(y|x) > 0$, there is no generator optimal for both ℓ_2 loss and GAN loss. On the other hand, ℓ_1 loss has some exceptional cases where the generator can minimize GAN loss and ℓ_1 loss at the same time. We identify what the cases are and explain why such cases are rare.

To begin with, ℓ_1 loss is decomposed as follows:

$$\begin{aligned}\mathcal{L}_1 &= \mathbb{E}_{x,y,z}\big[|y-G(x,z)|\big] \\ &= \mathbb{E}_{x,z}\Big[\mathbb{E}_y\big[|y-G(x,z)|\big]\Big] \\ &= \mathbb{E}_{x,z}\Big[\int p(y|x)|y-G(x,z)|dy\Big] \\ &= \mathbb{E}_{x,z}\Big[\int_{-\infty}^{G(x,z)} p(y|x)(G(x,z)-y)dy + \int_{G(x,z)}^{\infty} p(y|x)(y-G(x,z))dy\Big]\end{aligned}$$

To minimize ℓ_1 loss, we need the gradient w.r.t. $G(x,z)$ to be zero for all x and z that $p(x) > 0$ and $p(z) > 0$. Note that this is a sufficient condition for minimum since the ℓ_1 loss is convex.

$$\frac{\partial \mathcal{L}_1}{\partial G(x,z)} = \int_{-\infty}^{G(x,z)} p(y|x)dy - \int_{G(x,z)}^{\infty} p(y|x)dy = 2\int_{-\infty}^{G(x,z)} p(y|x)dy - 1 = 0 \tag{18}$$

$$\int_{-\infty}^{G(x,z)} p(y|x)dy = \frac{1}{2} \tag{19}$$

Therefore, $G(x,z)$ should be the conditional median to minimize the loss. Unlike ℓ_2 loss, there can be an *interval* of $G(x,z)$ that satisfies Eq.(19). Specifically, any value between interval $[a,b]$

is a conditional median if $\int_{-\infty}^{a} p(y|x)dy = \int_{-\infty}^{b} p(y|x)dy = \frac{1}{2}$. If every real data belongs to the interval of conditional median, then the generator can be optimal both GAN loss and ℓ_1 loss.

For instance, assume that there are only two discrete values of y possible for any given x, say -1 and 1, with probability 0.5 for each. Then the interval of median becomes $[-1, 1]$, thus any $G(x, z)$ in the interval $[-1, 1]$ minimizes the ℓ_1 loss to 1. If the generated distribution is identical to the real distribution, *i.e.* generating -1 and 1 with the probability of 0.5, the generator is optimal w.r.t. both GAN loss and ℓ_1 loss.

However, we note that such cases hardly occur. In order for such cases to happen, for any x, every y with $p(y|x) > 0$ should be the conditional median, which is unlikely to happen in natural data such as images. Therefore, the set of optimal generators for ℓ_1 loss is highly likely to be disjoint with the optimal set for GAN loss.