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, can largely be accounted for 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 is problematic 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 are applicable to any conditional generation task. We show this by performing thorough experiments on image-to-image translation, super-resolution, and image inpainting tasks using Cityscapes, and CelebA dataset. Quantitative evaluation also confirms that our methods achieve a great diversity in outputs while retaining or even improving the quality of images.

1 INTRODUCTION

Recently, active research has led to 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 $\ell_1/\ell_2$ 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. $\ell_1/\ell_2$ loss) aggravates the mode collapse, a notorious problem of GANs. In conditional generation tasks, which are intrinsically learning one-to-many mappings, 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 the output, and surprisingly many previous methods omit a 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. $\ell_1/\ell_2$ loss) to predict conditional statistics of the 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 the 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 are applicable to any conditional generation task. 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


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. 2018). 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. (2018) 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 $\mathcal{L}_{GAN}$ and the reconstruction loss $\mathcal{L}_{Rec}$.

$$\mathcal{L} = \mathcal{L}_{GAN} + \lambda_{Rec}\mathcal{L}_{Rec}$$  (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. $\ell_1$/$\ell_2$ loss), we consider the perceptual loss as a branch of the reconstruction loss.

The loss $\mathcal{L}_{GAN}$ 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 $G$.

$$\min_{G} \max_{D} \mathbb{E}_{x,y}[\log D(x,y)] + \mathbb{E}_{x,z}[\log (1 - D(x,G(x,z)))]$$  (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 $\ell_1$, (Isola et al., 2017; Wang et al., 2017) and $\ell_2$ loss (Pathak et al., 2016; Mathieu et al., 2016). Both losses can be formulated as follows with $p = 1, 2$ respectively.

$$\mathcal{L}_{Rec} = \mathcal{L}_p = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_p].$$  (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 $(\mathcal{X}, \mathcal{Y})$ assuming each distribution is defined as follows.

$$p_L(\mathcal{Y} | \mathcal{X}; \theta) = \prod_{i=1}^{N} \frac{1}{2b} \exp(-\frac{\|y_i - f_{\theta}(x_i)\|}{b}), \quad p_G(\mathcal{Y} | \mathcal{X}; \theta) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma^2} \exp(-\frac{(y_i - f_{\theta}(x_i))^2}{2\sigma^2})$$

where $N$ is the size of the dataset, and $f$ is the model parameterized by $\theta$. The central and dispersion measure for Gaussian are the mean and variance $\sigma^2$, and the correspondences for Laplace are the median and mean absolute deviation (MAD) $b$. Therefore, using $\ell_2$ loss leads the model output $f_{\theta}(x)$ to become an estimate of the conditional average of $y$ given $x$, while using $\ell_1$ loss is equivalent to estimating the conditional median of $y$ given $x$ (Bishop, 2006). Note that the model $f_{\theta}$ is trained to predict the mean (or median) of the data distribution, not to generate 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 can be problematic, because it may worsen the mode collapse. Below, we discuss this argument both mathematically and empirically.

We analyze the mismatch between GAN loss and $\ell_2$ loss in terms of the set of optimal generators. Refer to appendix $\text{F}$ for the case of $\ell_1$ loss. The sets of optimal generators for the GAN loss and the $\ell_2$ loss, denoted by $\mathcal{G}$ and $\mathcal{R}$ respectively, can be formulated as follows:

$$\mathcal{G} = \{G | p_{data}(y|x) = p_G(y|x)\}, \quad \mathcal{R} = \{G | \arg \min_G \mathbb{E}_{x,y,z}[\mathcal{L}_2(G(x,z), y)]\}. $$  (4)

The problem of $\ell_2$ loss is that it forces the model to predict only the mean of $p(y|x)$, while pushing the conditional variance to zero. According to James (2003), the $\ell_2$ loss can be decomposed into one irreducible term, the inherent noise of data, and two reducible terms: the squared bias and the variance of outputs. Thus, $\ell_2$ loss is minimized when both bias and variance are zero, which implies that $\mathcal{R}$ is a subset of $\mathcal{V}$ where $G$ has no conditional variance.

$$\mathcal{R} \subset \mathcal{V} = \{G | \text{Var}_{x}(G(x,z)|x) = 0\}.$$  (5)

In conditional generation tasks, however, many outputs can be paired with the input, i.e. the conditional variance $\text{Var}_{x}(y|x)$ is non-zero. Thereby we conclude $\mathcal{G} \cap \mathcal{V} = \emptyset$, which reduces to $\mathcal{G} \cap \mathcal{R} = \emptyset$. It means that any generator $G$ cannot be optimal to both GAN and L2 loss simultaneously. Therefore, it is hard to anticipate what solution is attained by training and how the model behaves when the two losses are combined. One thing for sure is that the mode collapse is an inevitable consequence.
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 (Cordts et al., 2016). We slightly modify the model so that it takes additional noise input. We use the $\ell_1$ loss as the reconstruction loss as it was 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 the discriminator is too unstable to learn the translation task. In Figure 1(d), the model with only the $\ell_1$ loss is more stable but produces results far from being realistic. The combination of the two losses in Figure 1(e) helps not only to reliably train the model but also to generate visually appealing images; yet, its results lack variation. 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 task. Trained with our new loss terms, the model can accomplish both the stability of training and multimodal generation as already seen in Figure 1(f). 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 $\ell_1/\ell_2$ loss to estimate all the parameters of the Laplace and Gaussian distribution. In section 4.2 and 4.3, we present two novel training schemes for conditional GANs. Finally, in section 4.4, we discuss our methods’ novelties over existing algorithms.

4.1 THE MLE FOR MEAN AND VARIANCE

The $\ell_2$ loss encourages the model to perform the MLE of the conditional mean of $y$ given $x$ while the variance $\sigma^2$, the other parameter of Gaussian, is assumed to be fixed. If we allow the model to
estimate the conditional variance as well, the MLE loss corresponds to
\[
\mathcal{L}_{\text{MLE, Gaussian}} = \mathbb{E}_{x,y} \left[ \frac{(y - \hat{\mu})^2}{2\hat{\sigma}^2} + \frac{1}{2} \log \hat{\sigma}^2 \right] \quad \text{where} \quad \hat{\mu}, \hat{\sigma}^2 = f_\theta(x) \tag{6}
\]
where the model \( f_\theta \) now estimates both \( \hat{\mu} \) 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
\[
\mathcal{L}_{\text{MLE, Laplace}} = \mathbb{E}_{x,y} \left[ \frac{|y - \hat{\mu}|}{\hat{b}} + \log \hat{b} \right] \quad \text{where} \quad \hat{\mu}, \hat{b} = f_\theta(x) \tag{7}
\]
where \( \hat{\mu} \) is the predicted median and \( \hat{b} \) is the predicted MAD ([Bloesch et al., 2018]). 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 \( \ell_2 \) loss under Gaussian assumption. It is straightforward to obtain Laplace versions of our methods for the \( \ell_1 \) 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 \( y_{1:K} \) for each input \( x \) by varying noise input \( z \). Second, we introduce a separate component predictor, which is a clone of the 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. (6) 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 the generator. Specifically, we match the predicted mean/variance and the mean/variance of the generator’s distribution, which is computed by the sample mean \( \hat{\mu} \) and variance \( \hat{\sigma}^2 \) from \( y_{1:K} \). 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,
\[
\mathcal{L} = \mathcal{L}_{\text{GAN}} + \lambda \mathcal{L}_{\text{MLMM}} \quad \text{where} \quad \mathcal{L}_{\text{MLMM}} = (\hat{\mu} - \mu)^2 + (\hat{\sigma}^2 - \sigma^2)^2, \tag{8}
\]
\[
\hat{\mu}, \hat{\sigma}^2 = P(x); \quad \hat{\mu} = \frac{1}{K} \sum_{i=1}^{K} \hat{y}_i, \quad \hat{\sigma}^2 = \frac{1}{K-1} \sum_{i=1}^{K} (\hat{y}_i - \hat{\mu})^2, \quad \text{where} \quad \hat{y}_{1:K} = G(x, z_{1:K}). \tag{9}
\]
One possible variant is to match only the first moment $\mu$. We denote the original method 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 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 to match them 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, we can consider the statistics of the generated distribution as the estimators for the statistics of the real distribution. Like MLMM, we can use the sample statistics $\hat{\mu}$ and $\hat{\sigma}^2$ as approximated statistics of the generated distribution (in this case, it corresponds to Monte Carlo estimation). Then we replace $\mu$ and $\sigma^2$ in Eq.(6) with $\hat{\mu}$ and $\hat{\sigma}^2$ to obtain the Monte Carlo MLE (MCMLE) loss:

$$
\mathcal{L}_{\text{MCMLE}} = E_{x,y} \left[ \frac{(y - \hat{\mu})^2}{2\hat{\sigma}^2} + \frac{1}{2} \log \hat{\sigma}^2 \right],
$$

where $\hat{\mu}$ and $\hat{\sigma}^2$ are defined as Eq.(9). The MCMLE architecture is illustrated in Figure 2(c). Similarly to MLMM, we can build a variant MCMLE$_1$, where only the $\ell_2$ loss between $y$ and $\hat{y}$ is used. Also, Laplace assumption yields the Laplace versions of 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 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 bias caused by the predictor. On the other hand, the use of predictor in MLMM provides less variance. 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 suffer from a loss of diversity in the output samples unlike existing models using the reconstruction loss in terms of the set of optimal generators, as in section 3.2. The sets of optimal mappings for our loss terms are formulated as follows:

$$
\mathcal{M}_1 = \{ G | E_g[(E_y[y|x] - E_z[G(x,z)|x])^2] = 0 \} \quad (11)
$$

$$
\mathcal{M}_2 = \{ G | E_g[E_y(y|x) - E_z[G(x,z)|x]]^2 + (\text{Var}_y(y|x) - \text{Var}_z(G(x,z)|x))^2 = 0 \} \quad (12)
$$

where $\mathcal{M}_1$ corresponds to MLMM$_1$ and MCMLE$_1$ while $\mathcal{M}_2$ corresponds to MLMM$_{1/2}$ and MCMLE$_{1/2}$. It is straightforward to show that $G \subset \mathcal{M}_2 \subset \mathcal{M}_1$; 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:

$$
[E_y[y|x] = E_z[G(x,z)|x]] \land [\text{Var}_y(y|x) = \text{Var}_z(G(x,z)|x)].
$$

To summarize, there is no generator that is both optimal for the GAN loss and the $\ell_2$ loss since $G \cap \mathcal{R} = \emptyset$. Moreover, as the $\ell_2$ 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 loss is also optimal w.r.t. our loss terms since $G \subset \mathcal{M}_2 \subset \mathcal{M}_1$.

This proof may not fully demonstrate why our model does not give up multimodality, which could be an interesting future work in line with the mode collapsing issue of GANs. Nonetheless, we can at least assert that our loss functions do not suffer from the same side effect as the reconstruction loss does. Another remark is that this proof is confined to conditional GAN models without explicit encoding of latent variable. It may not be applicable to the models that explicitly encode latent variables such as Bao et al. (2017), Zhu et al. (2017), Lee et al. (2018a), Huang et al. (2018), and Lee et al. (2018b), since the latent variables provide meaningful information about the target.
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 the 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 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 as well as reconstruction loss in all three tasks. On the other hand, we could not find any working configuration of MCMLE in the image inpainting task. Also, MCMLE is more sensitive to the number of samples generated for each 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 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 the 300 different inputs using the trained 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, similarly 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.
not improve the performance. Moreover, additional errors may arise from predicting more statistics, means could be enough to guide GAN training in many cases. It implies that adding more statistics or covariance. Third, using the statistics of high-level features may capture correlations which are unreachable with pixel-level statistics.

Second, in terms of statistics matching, our methods can be extended to explore other higher order prediction, for which our methods can be directly applied to generate diverse, high-quality samples. 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.

### 6 Conclusion

In this work, we pointed out that there is a significant mismatch between conventional reconstruction losses and GAN loss. As alternatives, we proposed a set of novel loss functions named MLMM and MCMLE that enable conditional GAN models to accomplish both stability of training and multi-modal 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Realism</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random real data</td>
<td>1.00</td>
<td>0.559</td>
</tr>
<tr>
<td>Pix2Pix+noise</td>
<td>0.22</td>
<td>0.004</td>
</tr>
<tr>
<td>BicycleGAN</td>
<td>0.16</td>
<td>0.191</td>
</tr>
<tr>
<td>MLMM</td>
<td>0.49</td>
<td>0.519</td>
</tr>
<tr>
<td>MLMM1/2</td>
<td>0.44</td>
<td>0.388</td>
</tr>
<tr>
<td>MCMLE1</td>
<td>0.54</td>
<td>0.299</td>
</tr>
<tr>
<td>MCMLE1/2</td>
<td>0.14</td>
<td>0.453</td>
</tr>
<tr>
<td>Laplace</td>
<td>0.19</td>
<td>0.368</td>
</tr>
<tr>
<td>MLMM</td>
<td>0.69</td>
<td>0.380</td>
</tr>
<tr>
<td>MLMM1/2</td>
<td>0.17</td>
<td>0.393</td>
</tr>
<tr>
<td>MCMLE1</td>
<td>0.19</td>
<td>0.384</td>
</tr>
<tr>
<td>MCMLE1/2</td>
<td>0.20</td>
<td>0.453</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Realism</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random real data</td>
<td>1.00</td>
<td>0.290</td>
</tr>
<tr>
<td>SRGAN+noise</td>
<td>0.60</td>
<td>0.005</td>
</tr>
<tr>
<td>Gaussian MLMM1/2</td>
<td>0.52</td>
<td>0.050</td>
</tr>
<tr>
<td>GLCIC+noise</td>
<td>0.28</td>
<td>0.004</td>
</tr>
<tr>
<td>Gaussian MLMM1/2</td>
<td>0.33</td>
<td>0.060</td>
</tr>
</tbody>
</table>

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.
REFERENCES


Michael Bloesch, Jan Czarnowski, Ronald Clark, Stefan Leutenegger, and Andrew Davison. CodeSLAM - learning a compact, optimisable representation for dense visual SLAM. *CVPR*, 2018.


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.


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.


A ALGORITHMS

We elaborate on 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 only calculate the gradient 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

Require: Generator \(G\), discriminator \(D\), pre-trained predictor \(P\), MLMM loss coefficient \(\lambda_{MLMM}\)

Require: input \(x\), ground truth \(y\), the number of samples \(K\)

1: for \(i = 1\) to \(K\) do
2: \(z_i \leftarrow \text{GenerateNoise()}\)
3: \(\tilde{y}_i \leftarrow G(x, z_i)\) \{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: \(L_{MLMM} \leftarrow (\hat{\mu} - \tilde{\mu})^2\)
8: \(L_{GAN} \leftarrow \frac{1}{K} \sum_{i=1}^{K} - \log D(x, \tilde{y}_i)\)
9: \(\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G} L_{GAN} + \lambda_{MLMM} L_{MLMM})\)

Algorithm 2 Generator update in Gaussian MLMM_{1/2}

Require: Generator \(G\), discriminator \(D\), pre-trained predictor \(P\), MLMM loss coefficient \(\lambda_{MLMM}\)

Require: input \(x\), ground truth \(y\), the number of samples \(K\)

1: for \(i = 1\) to \(K\) do
2: \(z_i \leftarrow \text{GenerateNoise()}\)
3: \(\tilde{y}_i \leftarrow G(x, z_i)\) \{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: \(\hat{\sigma}^2 \leftarrow \frac{1}{K} \sum_{i=1}^{K} (\hat{y}_i - \hat{\mu})^2\) \{Sample variance\}
8: \(L_{MLMM} \leftarrow (\hat{\mu} - \tilde{\mu})^2 + \hat{\sigma}^2\)
9: \(L_{GAN} \leftarrow \frac{1}{K} \sum_{i=1}^{K} - \log D(x, \tilde{y}_i)\)
10: \(\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G} L_{GAN} + \lambda_{MLMM} L_{MLMM})\)
Algorithm 3 Generator update in Laplace MLMM$_1$

Require: Generator $G$, discriminator $D$, pre-trained predictor $P$, MLMM loss coefficient $\lambda_{\text{MLMM}}$

Require: input $x$, ground truth $y$, the number of samples $K$

1: for $i = 1$ to $K$ do
2: $z_i \leftarrow \text{GenerateNoise}()$
3: $\tilde{y}_i \leftarrow G(x, z_i)$ \{Generate $K$ samples\}
4: end for
5: $\hat{m} \leftarrow P(x)$ \{Predicted median\}
6: $\tilde{m} \leftarrow \text{med}(\tilde{y}_1:K)$ \{Sample median\}
7: $t_{1:K} = \text{detach}(\tilde{y}_1:K + (\hat{m} - \tilde{m}))$
8: $\mathcal{L}_{\text{MLMM}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} (t_i - \tilde{y}_i)^2$
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 4 Generator update in Laplace MLMM$_{1/2}$

Require: Generator $G$, discriminator $D$, pre-trained predictor $P$, MLMM loss coefficient $\lambda_{\text{MLMM}}$

Require: input $x$, ground truth $y$, the number of samples $K$

1: for $i = 1$ to $K$ do
2: $z_i \leftarrow \text{GenerateNoise}()$
3: $\tilde{y}_i \leftarrow G(x, z_i)$ \{Generate $K$ samples\}
4: end for
5: $\hat{m}, \hat{b} \leftarrow P(x)$ \{Predicted median and MAD\}
6: $\tilde{m} \leftarrow \text{med}(\tilde{y}_1:K)$ \{Sample median\}
7: $\hat{b} \leftarrow \frac{1}{K} \sum_{i=1}^{K} |\tilde{y}_i - \hat{m}|$ \{Sample MAD\}
8: $t_{1:K} = \text{detach}(\tilde{y}_1:K + (\hat{m} - \tilde{m}))$
9: $\mathcal{L}_{\text{MLMM}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} (t_i - \tilde{y}_i)^2 + (\hat{b} - \hat{b})^2$
10: $\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} - \log D(x, y_i)$
11: $\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G} \mathcal{L}_{\text{GAN}} + \lambda_{\text{MLMM}} \mathcal{L}_{\text{MLMM}})$

Algorithm 5 Generator update in Gaussian MCMLE$_1$

Require: Generator $G$, discriminator $D$, MCMLE loss coefficient $\lambda_{\text{MCMLE}}$

Require: input $x$, ground truth $y$, the number of samples $K$

1: for $i = 1$ to $K$ do
2: $z_i \leftarrow \text{GenerateNoise}()$
3: $\tilde{y}_i \leftarrow G(x, z_i)$ \{Generate $K$ samples\}
4: end for
5: $\hat{m} \leftarrow \frac{1}{K} \sum_{i=1}^{K} \tilde{y}_i$ \{Sample mean\}
6: $\mathcal{L}_{\text{MCMLE}} \leftarrow (y - \hat{m})^2$
7: $\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} - \log D(x, y_i)$
8: $\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G} \mathcal{L}_{\text{GAN}} + \lambda_{\text{MCMLE}} \mathcal{L}_{\text{MCMLE}})$

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

Require: Generator $G$, discriminator $D$, MCMLE loss coefficient $\lambda_{\text{MCMLE}}$

Require: input $x$, ground truth $y$, the number of samples $K$

1: for $i = 1$ to $K$ do
2: $z_i \leftarrow \text{GenerateNoise}()$
3: $\tilde{y}_i \leftarrow G(x, z_i)$ \{Generate $K$ samples\}
4: end for
5: $\hat{m} \leftarrow \frac{1}{K} \sum_{i=1}^{K} \tilde{y}_i$ \{Sample mean\}
6: $\hat{\sigma}^2 \leftarrow \frac{1}{K} \sum_{i=1}^{K} (\tilde{y}_i - \hat{m})^2$ \{Sample variance\}
7: $\mathcal{L}_{\text{MCMLE}} \leftarrow \frac{(y - \hat{m})^2}{\hat{\sigma}^2} + \frac{1}{2} \log \hat{\sigma}^2$
8: $\mathcal{L}_{\text{GAN}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} - \log D(x, y_i)$
9: $\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G} \mathcal{L}_{\text{GAN}} + \lambda_{\text{MCMLE}} \mathcal{L}_{\text{MCMLE}})$
Algorithm 7 Generator update in Laplace MCMLE

Require: Generator $G$, discriminator $D$, MCMLE loss coefficient $\lambda_{\text{MCMLE}}$

Require: input $x$, ground truth $y$, the number of samples $K$

1: for $i = 1$ to $K$ do
2: \hspace{1em} $z_i \leftarrow \text{GenerateNoise}()$
3: \hspace{1em} $\tilde{y}_i \leftarrow G(x, z_i)$ \{Generate $K$ samples\}
4: end for
5: $\tilde{m} \leftarrow \text{med}(\tilde{y}_{1:K})$ \{Sample median\}
6: $t_{1:K} = \text{detach}(\tilde{y}_{1:K} + (y - \tilde{m}))$
7: $L_{\text{MCMLE}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} |t_i - \tilde{y}_i|$
8: $L_{\text{GAN}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} -\log D(x, y_i)$
9: $\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G} L_{\text{GAN}} + \lambda_{\text{MCMLE}} L_{\text{MCMLE}})$

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

Require: Generator $G$, discriminator $D$, MCMLE loss coefficient $\lambda_{\text{MCMLE}}$

Require: input $x$, ground truth $y$, the number of samples $K$

1: for $i = 1$ to $K$ do
2: \hspace{1em} $z_i \leftarrow \text{GenerateNoise}()$
3: \hspace{1em} $\tilde{y}_i \leftarrow G(x, z_i)$ \{Generate $K$ samples\}
4: end for
5: $\tilde{m} \leftarrow \text{med}(\tilde{y}_{1:K})$ \{Sample median\}
6: $\tilde{b} \leftarrow \frac{1}{K} \sum_{i=1}^{K} |\tilde{y}_i - \tilde{m}|$ \{Sample MAD\}
7: $t_{1:K} = \text{detach}(\tilde{y}_{1:K} + (y - \tilde{m}))$
8: $L_{\text{MCMLE}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} \frac{|t_i - \tilde{y}_i|}{\tilde{b}} + \log \tilde{b}$
9: $L_{\text{GAN}} \leftarrow \frac{1}{K} \sum_{i=1}^{K} -\log D(x, y_i)$
10: $\theta_G \leftarrow \text{Optimize}(\theta_G, \nabla_{\theta_G} L_{\text{GAN}} + \lambda_{\text{MCMLE}} L_{\text{MCMLE}})$

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 train 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.

Convergence speed. Our methods need more training steps (about $1.5 \times$) to generate high-quality images compared to those with the reconstruction loss. This is an expectable behavior because our methods train the model to generate a much wider range of outputs.

Training stability. MLMM is similar to the reconstruction loss in terms of training stability. Our methods work stably with a large range of hyperparameter $\lambda$. For example, the loss coefficient of MLMM can be set across several orders of magnitude (from tens to thousands) with similar results. However, as noted, MCMLE is unstable compared to MLMM.

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 $1 \times 1, 2 \times 2, 4 \times 4$ 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 $\lambda_{MLMM} = \lambda_{MCMLE} = 10$. For the baseline, we use $\ell_1$ 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 $\lambda_{MLMM} = 2400$ and $\lambda_{MCMLE} = 20$. For the baseline, we use $\ell_2$ 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 $128 \times 128$. 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 MLMM$_{1/2}$ and MCMLE methods. We successfully trained GLCIC with Gaussian MLMM$_{1/2}$ using $\lambda_{MLMM} = 1000$. However, we could not find any working setting for MCMLEs. For the baseline model, we use $\ell_2$ 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 $\ell_2$ 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.

Figure 4: Experiments on a synthetic 2D dataset. (a) The data distribution. (b)(c) Using GAN loss alone or with $\ell_2$ loss results in mode collapse. (d) Training a predictor with the MLE loss in Eq.(6) 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 MLMM_{1/2} (success cases).

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

Figure 10: GLCIC–CelebA Gaussian MLMM$_{1/2}$ (failure cases).
E Decomposition of Reconstruction Loss

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 \( \text{Var}(y) \) and two reducible terms: \( \text{SE}(\hat{y}, y) \) and \( \text{VE}(\hat{y}, y) \), 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.

\[
\mathbb{E}_{y, \hat{y}}[\mathcal{L}_s(y, \hat{y})] = \mathbb{E}_{y} [\mathcal{L}_s(y, \hat{y})] + \mathbb{E}_{y, \hat{y}} [\mathcal{L}_s(y, \hat{y}) - \mathcal{L}_s(y, \hat{y})] + \mathbb{E}_{y} [\mathcal{L}_s(y, \hat{y}) - \mathcal{L}_s(y, \hat{y})]
\]

\[\text{SE}(\hat{y}, y) = \mathbb{E}_{y, \hat{y}} [\mathcal{L}_s(y, \hat{y}) - \mathcal{L}_s(y, \hat{y})] \]

\[\text{VE}(\hat{y}, y) = \mathbb{E}_{y, \hat{y}} [\mathcal{L}_s(y, \hat{y}) - \mathcal{L}_s(y, \hat{y})] \]

\( S \) 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 \( \hat{y} = Sy = \hat{y} \) reducing both SE and VE to 0. For \( \ell_2 \) loss, \( \hat{y} \) and \( Sy \) are the expectations of \( y \) and \( \hat{y} \).

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

Also, SE and VE correspond to the squared bias and the variance, respectively.

\[
\text{SE}(\hat{y}, y) = \mathbb{E}_{y} [(\mathcal{L}_2(y, \hat{y}) - \mathcal{L}_2(y, \hat{y})^2) - (y - Sy)^2]
\]

\[
\text{VE}(\hat{y}, y) = \mathbb{E}_{y, \hat{y}} [(\mathcal{L}_2(y, \hat{y}) - \mathcal{L}_2(y, \hat{y}))^2 - (y - Sy)^2]
\]

This decomposition offers another interpretation of the \( \ell_2 \) loss: minimizing the bias and the variance of prediction.

F Mismatch between \( \ell_1 \) Loss and GAN Loss

In section 4.4 we showed that if there exists some \( x \) such that \( \text{Var}(y|x) > 0 \), there is no generator optimal for both \( \ell_2 \) loss and GAN loss. On the other hand, \( \ell_1 \) loss has some exceptional cases where the generator can minimize GAN loss and \( \ell_1 \) loss at the same time. We identify what the cases are and explain why such cases are rare.

To begin with, \( \ell_1 \) loss is decomposed as follows:

\[
\mathcal{L}_1 = \mathbb{E}_{x,y,z} [y - G(x, z)]
\]

\[
= \mathbb{E}_{x,z} [\mathbb{E}_{y} [y - G(x, z)]]
\]

\[
= \mathbb{E}_{x,z} \left[ \int p(y|x) y - G(x, z) dy \right]
\]

\[
= \mathbb{E}_{x,z} \left[ \int_{-\infty}^{G(x,z)} p(y|x) (G(x, z) - y) dy + \int_{G(x,z)}^{\infty} p(y|x) (y - G(x, z)) dy \right]
\]

To minimize \( \ell_1 \) 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 \( \ell_1 \) loss is convex.

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

\[
\int_{-\infty}^{G(x,z)} p(y|x) dy = \frac{1}{2}
\]
Therefore, $G(x, z)$ should be the conditional median to minimize the loss. Unlike $\ell_2$ loss, there can be an interval of $G(x, z)$ that satisfies Eq. (17). Specifically, any value between interval $[a, b]$ is a conditional median if \[ \int_{-\infty}^{a} p(y|x)\,dy = \int_{b}^{\infty} p(y|x)\,dy = \frac{1}{2}. \]
If every real data belongs to the interval of conditional median, then the generator can be optimal for both GAN loss and $\ell_1$ 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 $\ell_1$ 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 $\ell_1$ 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 $\ell_1$ loss is highly likely to be disjoint with the optimal set for GAN loss.

G EXPERIMENTS ON MORE COMBINATIONS OF LOSS FUNCTIONS

We present some more experiments on different combinations of loss functions. The following results are obtained in the same manner as section D.2 and section D.3.

Figure 11 shows the results when we train the Pix2Pix model only with our loss term (without the GAN loss). The samples of MLMM$_1$ and MCMLE$_1$ are almost similar to those with the reconstruction loss only. Since there is no reason to generate diverse images, the variances of the samples are near zero while the samples are hardly distinguishable from the output of the predictor. On the other hand, MLMM$_{1/2}$ and MCMLE$_{1/2}$ do generate diverse samples, although the variation styles are different from one another. In MLMM$_{1/2}$, where the attraction for diversity is relatively mild, the samples show low-frequency variation. Intriguingly, MCMLE$_{1/2}$ incurs high-frequency variation patterns as the sample variance of MCMLE$_{1/2}$ is much closer than MLMM$_{1/2}$ to the predicted variance.

(a) MLMM$_1$ only
(b) MLMM$_{1/2}$ only
(c) MCMLE$_1$ only
(d) MCMLE$_{1/2}$ only

Figure 11: Ablation test using our loss terms only. The first row of each image is arranged in the order of input, ground truth, predicted mean, sample mean, predicted variance, and sample variance. The other two rows are generated samples. Without GAN loss, the results of MLMM$_1$ and MCMLE$_1$ are almost identical to the result of $\ell_2$ loss (predicted mean), while MLMM$_{1/2}$ and MCMLE$_{1/2}$ show variations with different styles.
Another experiment that we carry out is about the joint use of all loss terms: GAN loss, our losses and reconstruction loss. Specifically, we use the following objective with varying $\lambda_{Rec}$ from 0 to 100 to train the Pix2Pix model on Cityscapes dataset.

$$\mathcal{L} = \mathcal{L}_{GAN} + 10\mathcal{L}_{MLMM} + \lambda_{Rec}\mathcal{L}_{Rec}$$

Figure 12 shows the results of the experiments. As $\lambda_{Rec}$ increases, the sample variance reduces. This confirms again that the reconstruction is the major cause of loss of variability. However, we find one interesting phenomenon regarding the quality of the samples. The sample quality deteriorates up to a certain value of $\lambda_{Rec}$, but gets back to normal as $\lambda_{Rec}$ further increases. It implies that either MLMM or the reconstruction loss can find some high-quality local optima, but the joint use of them is not desirable.