

A Generative Adversarial Density Estimator

M. Ehsan Abbasnejad, Javen Shi, Anton van den Hengel, Lingqiao Liu

{ehsan.abbasnejad, javen.shi, anton.vandenhengel, lingqiao.liu}@adelaide.edu.au

Australian Institute of Machine Learning & The University of Adelaide, Australia

Abstract

Density estimation is a challenging unsupervised learning problem. Current maximum likelihood approaches for density estimation are either restrictive or incapable of producing high-quality samples. On the other hand, likelihood-free models such as generative adversarial networks, produce sharp samples without a density model. The lack of a density estimate limits the applications to which the sampled data can be put, however. We propose a Generative Adversarial Density Estimator, a density estimation approach that bridges the gap between the two. Allowing for a prior on the parameters of the model, we extend our density estimator to a Bayesian model where we can leverage the predictive variance to measure our confidence in the likelihood. Our experiments on challenging applications such as visual dialog where the density and the confidence in predictions are crucial shows the effectiveness of our approach.

1. Introduction

Generative modelling is amongst the longest-standing problems in machine learning, and one that has been drawing increasing attention. This is at least partly due to the shortcomings of the predominant discriminative deep learning-based models. These shortcomings include the failure to generalise, a lack of robustness to data distribution changes, and the need for large volumes of training data.

Deep generative models have been successful in addressing some of these shortcomings. In particular, deep maximum likelihood models such as deep Boltzmann machines [40], variational autoencoders (VAEs) [23], autoregressive models [17, 36], real non-volume preserving transformations [13] etc. have demonstrated an impressive ability to model complex densities.

Likelihood-free approaches [20, 16] such as generative adversarial networks (GANs) [16] have outperformed previous deep generative models in their ability to model complex distributions. In image-based problems they have shown a particular ability to consistently generate sharp and

realistic looking samples [22, 48, 34]. GANs are one of the implicit generative models wherein density is not explicitly defined [11]. Unfortunately, GANs are incapable of evaluating densities by-design.

Maximum likelihood models have the advantage that they are able to represent probability density explicitly, but either perform poorly in terms of the quality of samples they generate or are relatively inefficient to train.

As modern datasets are typically large, high-dimensional and highly structured, the challenge is building models that are powerful enough to capture the underlying complexity yet still efficient to train in estimating the density of instances.

Density estimates are essential in a wide range of practical problems in Computer Vision, particularly when likelihoods over the generated samples are critical. This occurs, for example, when it is important to both explore and optimise over a search space, when confidence estimates about a hypothesis are required, or when control over the level of generalisation is important. The typical sample quality metrics are inadequate since the generative model may simply memorize the data or miss important modes [45]. Moreover, in an application such as one where the virtual agent is engaged in a dialog with the user about an image, the estimate of how likely an answer is in addition to the agent's confidence improves both dialogue fluidity and method performance. Alternatively, density estimates such as autoregressive models [17], flow-based methods [12, 13] or non-parameteric methods such as kernel density estimation (KDE) [16] are either overly computationally demanding or rely on heavy engineering of the neural networks involved.

In this paper, we introduce a Generative Adversarial Density Estimator that is both easy to train and expressive enough to model high-dimensional data. In particular, we bridge the gap between the maximum likelihood and likelihood-free models by explicitly modeling the likelihood while using adversarial samples to compute the normalizer. As a by-product of this model, we show the local properties of the density function are captured and allow for diverse samples to be generated. Further, we show our

approach is capable of modeling the likelihood of complex data, such as images, from which realistic looking samples can be taken.

In addition, our approach easily extends to Bayesian estimation where the distribution of the parameters are incorporated in the generative model. This allows us to compute the predictive variance which uncovers the “uncertainty” in the prediction. This uncertainty can be used in an applications such as Visual Dialog [10] where an agent is trained to respond to questions when confident or reply “I don’t know” for uncertain answers. This is essential if such agents are to be of practical utility.

This paper makes the following contributions:

1. We propose a generative adversarial density estimator network able to perform efficient sampling as well as likelihood evaluation without the need for an explicit invertible generator.
2. We propose a learning objective, based on sound theoretical grounds, for our adversarial density estimator that attains good log-likelihoods and generates high-quality samples on multiple datasets as well as estimating uncertainty in a vision-and-language problem.
3. Our approach is naturally extended to a Bayesian setting. In particular we investigate a conjugate prior and an efficient gradient based Monte Carlo method to estimate the posterior.
4. We propose a method to efficiently regularize the Jacobian of the generator function to evaluate the likelihoods of the model.
5. We provide a theoretical apparatus for future research in density estimation, in particular, utilizing adversarial training.

2. Related Work

Deep Boltzmann machines [40, 3] represent one of the earlier approaches to maximum likelihood modelling, but due to their intractable nature, they are hard to train. Recently, autoregressive approaches [36, 41] have been used to model the distribution of each image pixel value directly. As such, the ordering of the dimensions can be critical to the training of the model. The sequential nature of these models significantly limits their computational efficiency. VAEs [23, 15, 1] have shown to be successful in modeling a latent variable from which instances from the distribution can be reconstructed. While these approaches have been very successful, image samples generated by VAEs are generally blurry and exhibit unrealistic artefacts.

Generative Adversarial Networks (GANs) [16] avoid the maximum likelihood principle altogether. Instead, the generative network is associated with a discriminator network whose task is to distinguish between samples and real data. Successfully trained GAN models consistently outperform their counterparts in producing high-quality, sharp and

realistic image samples [46, 22]. However, GANs cannot estimate the density function over the sampled data instances, even though approaches such as Wasserstein distance (WGAN) [6, 19, 2] and f-GAN [35] are similar to ours. Recently, [38, 18] considered likelihood maximisation with a change of variable. However, these approaches require designing a function that is both invertible and that has an easily computable Jacobian. These designs typically produce low quality samples, or are inefficient.

3. Generative Adversarial Density Estimator

3.1. Adversarial Formulation

We consider the problem of estimating the density for an observation sample x (e.g. an image) defined by

$$p(x|w) = \exp(-\phi(w, x) - A(w)),$$

$$A(w) = \log \int \exp(-\phi(w, x)) dx. \quad (1)$$

Here, $\phi(w, x)$ is the energy function for the Boltzmann distribution and the parameter w fully specifies the density function. We use a deep neural network with linear output to model this function, i.e. $\phi(w, x) = w_1 \cdot \phi_2(w_2, x)$, where $w = \{w_1, w_2\}$ and feature vector $\phi_2(w_2, x)$ is the sufficient statistics that we learn. Alternatively ϕ can be seen as a function that maps the input x to a feature space where the density is easy to formulate. In addition, $A(w)$ is the normalizer that ensures $\int p(x|w) dx = 1$.

Given a dataset $X = \{x_1 \dots, x_n\}$ where samples are drawn i.i.d from the underlying distribution p , we are interested in finding parameter w in Equation 1. To that end, we maximize the expected log-likelihood of these observations in the dataset with respect to this density model to obtain the parameter w , that is, $\max_w E_p[\log p(x|w)]$, where

$$E_p[\log p(x|w)] = \int \log(p(x|w)) p(x|w) dx$$

$$= E_p[-\phi(w, x)] - A(w) \quad (2)$$

Computing the normalizer $A(w)$ is intractable, and thus we resort to approximations. In particular, we utilize the variational principle to obtain an upper bound on $A(w)$ using an alternative distribution q parameterized by θ as

$$A_q(w) = \sup_{\theta} \left[\frac{E_q[-\phi(w, x)]}{\int \phi(w, x) q(x) dx} - \int -q(x) \log(q(x)) dx \right] \quad (3)$$

$H_x(q)$

Note that direct maximization of $A_q(w)$ yields the optimal value when $p = q$, i.e. when q takes the same form as p . Classical variational inference methods (see e.g. [23])

define a class of distributions from which a q is selected (or rather parameter θ is obtained). Here, motivated by the approach used in GANs, we specify q through a likelihood-free deterministic function which we use to generate samples (hence called generator). Assuming this generator function $x = g_\theta(z)$, $z \sim p_z$ is invertible¹, by the calculus of variable change [7] $E_q[\phi(w, x)]$ in Equation 3 is:

$$\int \phi(w, x) q(x) dx = \int \phi(w, g_\theta(z)) p_z(g_\theta^{-1}(x) | z, \theta) \det J_x J_z^{-1/2} dz \quad (4)$$

$$\text{and } p_z(g_\theta^{-1}(x) | z, \theta) \det J_x J_z^{-1/2} dz = p_z(z | x) dz,$$

where p_z is the prior distribution with parameter θ and J_x is the Jacobian of $g_\theta(x)^{-1}$. Here we use the convention that θ is composed of two parts: θ_g, θ_z for the parameters of the transformation function $g_\theta(\cdot)$ and the distribution of z , respectively. In the second line of Equation 4, we specify that g_θ preserves the volume with respect to (w, x) . Note in practice we use (w_2, x) as a surrogate for g^{-1} and regularize it to have a relatively constant determinant (by adding a regularization term for the gradients of (w_2, x)).

To compute the entropy of q in $H_x(q)$ in Equation 3, again using variable change, we have $\log p_z(z) = \log q(g(z)) + \log(\det J_z J_z^{-1/2})$ and $q(g(z)) = q(x)$ for which the expectation yields,

$$H_x(q) = \underbrace{H(\bar{p}_z)}_{\text{entropy of prior noise}} + \underbrace{H_g}_{\text{local geometry of the transformed space}}, \quad (5)$$

where $H_g = -E_{p_z} \log \det(J_z J_z)^{1/2}$

where J_z is the Jacobian of g_θ (the generator). Intuitively, this equality states that the entropy with respect to the distribution of the generated samples is the sum of two terms (1) the entropy of the noise prior which captures the information present in the noise distribution (specified by parameter θ_z) and (2) the expected local behavior in the transformed space (specified by parameter θ_g). From the information geometric perspective, the latter term captures the volume of the geometric space that the transformation function g_θ produces.

Remark 1. Explicitly constraining the generator to be invertible is infeasible, however, in practice we can use networks with inverted architecture to mediate this. Furthermore, when the matrix $J_z J_z$ is full-rank, it indicates the mapping is one-to-one, i.e. there is an invertible function that maps samples from z to observation samples x .

Since computing the $\log \det$ term in Equation 5 is computationally expensive, we resort to an approximation that

¹This is merely for theoretical analysis. In practice, we don't need to explicitly define an invertible function

is easier to optimize with stochastic gradient descent methods. We utilize the following Lemma²:

Lemma 2. For a matrix $A \in \mathbb{R}^{d \times d}$ and I_d the identity matrix of size d where $\lambda_1(A)$ denotes the largest eigenvalue and $\lambda_1(A) < \infty$ we have

$$\log \det(A) = d \log(\lambda_1(A)) + \frac{(-1)^{k+1} \text{tr}(A/\lambda_1(A)^k)}{k}. \quad (6)$$

Proof. Refer to the supplementary material. \square

Employing Lemma 2, we are able to substitute the computation of the determinant for a more efficient alternative approximation. This approximation is arbitrarily close to the true value depending on the number of steps in the series and the value chosen for k .

3.2. Main Algorithm

Taking the derivative of the normalizer w.r.t. the parameter w_1 in Equation 1 yields the first moment of this distribution, i.e. $E[(w_2, x)] = \frac{A(w)}{w_1}$. Furthermore, the second derivative is the covariance (and in this case Fisher information specifying the sensitivity of the parameter to the input). As such, it matches the moments from the true distribution and its estimate with respect to the sufficient statistics (w_2, x) when using maximum likelihood.

Putting Equation 1, 3, 4 and 5 together, we have a max-min problem to solve for two sets of parameters: w and θ . Finding the saddle point solution for this problem is challenging, as such, we employ an alternating optimization using stochastic gradient descent to update the parameters w and θ iteratively, as

$$w \leftarrow w + \eta_w \nabla_w E_p(w, x) - E_{p_z}(w, g_\theta(z)) - \eta_g E_{p_z}(w, g_\theta(z)) + H(p_z) + H_g, \quad (7)$$

In general, we need to optimize w at each step until convergence using the θ found in the previous iteration. In practice, we use a smaller learning rate to update θ (or fewer rounds of update) to guarantee convergence [25].

The transformation function acts similar to the generator in GANs is mode-seeking and it is complemented by the ability of maximum likelihood to cover the space of X . In particular, maximizing the entropy in the input noise and the output ensures a transformation function g_θ that does not collapse.

Based on the definition we provided in Equation 4, the ideal distribution for p_z is one that follows a similar distribution. In particular, z is more likely for the most frequent samples in the true distribution for which $w(x)$ is the highest. However, as a prior we can choose a simple distribution such as a Gaussian whose entropy $H(p_z)$ can

²A randomized version of this lemma is proposed in [8]

be analytically computed in closed-form (e.g. for Gaussian $\log(\frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2\sigma^2}})$ for standard deviation σ).

Remark 3. It is interesting to note that if $g_g(\cdot) = \cdot^{-1}(\cdot)$ then from Equation 4, $E_q[w_1(w_2, x)] = E_{p_z}[w_1(w_2, g_g(z))] = E_{p_z}[w_1 z]$, hence $E_q[w_2(x)] = E_{p_z}[z]$ and $E_p[w_2(x)] = E_{p_z}[z]$. This entails distribution of prior z should match that of w_2 in expectation. In practice, constraining g_g to be an inverse of w_2 (even in expectation) is too restrictive and hard to implement³. However, it is fruitful to know when constructing neural networks for this task, an inverse architecture for the networks learning g_g and w_2 has the best chance of success. Similarly, it is based on this argument and the entropy in Equation 5 that it is better to learn the distribution of z . However, it should be noted that since g_g is a deterministic function, changes to the distribution of z can negatively impact our generator or decrease the convergence speed of g_g network. In practice, we make sure the learning rate of z is smaller than g_g , its counterpart for g_g .

To estimate the integrals in the normalizer $A_q(w)$ we employ Monte Carlo method using samples the generator produces and Lemma 2 to obtain $A(w)$, i.e.

$$\hat{A}(w) = \frac{1}{m} \sum_{j=1}^m (w, x_j) + H(p_z) + \hat{H}_g$$

where $x_j = g_g(z_j)$, $z_j \sim q(z_j | z)$, $j = 1, \dots, m$ (8)

Here, \hat{H}_g is the empirical estimate of H_g using samples z_j for the Jacobian of the transformation function g_g .

The summary of the algorithm is shown in Algorithm 1 where we employ this alternating optimization. In our approach density estimator maximizes the log-likelihood of the given data with respect to the model in Equation 1. Generated samples from the transformation function g_g are the byproduct of this maximization in the normalizer.

We note that our approach in maximizing the H_g in Equation 5 resembles the ones in volume preserving or non-volume preserving generative models [13, 12] and normalized flows [18, 38] that have been recently investigated. These approaches maximize the likelihood of the data (corresponding to only maximizing H_g) directly by carefully designing a function that is both invertible and its Jacobian is easy to compute. These designs typically lead to approximations that either produce subpar samples or are inefficient. Alternatively, our approach does not require an

³We observed in practice that if the generator function g_z shares weights with that of w_2 (i.e. second layer of g_z with last layer of w_2 and so forth) the approach does not perform well. In addition, even if we constrain each layer of the g_z network to match in distribution to layers of the quality of the generated samples deteriorate.

Algorithm 1 Our density estimation algorithm.

```

1: Input:  $X$ , learning rates  $w_1, g, z, z < g < w$ ,
   regularization parameter
2: while is not converged do
3:   Sample uniformly  $x_1, \dots, x_n$  from  $X$ 
4:   Sample  $z_1, \dots, z_m \sim p_z(z | z)$ 
5:    $w = w - \frac{1}{n} \sum_{i=1}^n (w, x_i) - \frac{1}{m} \sum_{i=1}^m (w, g(z_i))$ 
6:    $R = -\frac{1}{w_2} (w_2, x)^2$ 
7:    $w = w + w \cdot \text{Adam}(w, w + wR)$ 
8:   Sample  $z_1, \dots, z_m \sim p_z(z | z)$ 
9:    $\hat{H}_g = \frac{1}{2} \log \det(J_z J_z)$  Using Lemma 2
10:   $F_{g,zG} = F_{g,zG} + \frac{1}{m} \sum_{i=1}^m (w, g_{F_{g,zG}}(z_i)) + \hat{H}_g$ 
11:   $g = g - g \cdot \text{Adam}(g, g)$ 
12:   $z = z + z \cdot \text{Adam}(z, z H(p_z) + F_{g,zG})$ 
13: end while

```

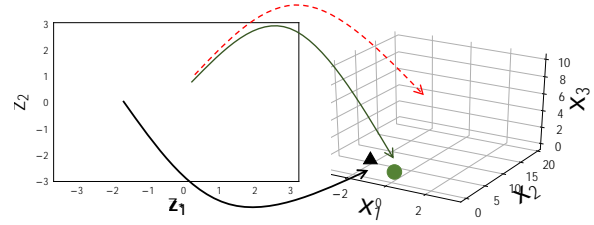


Figure 1. Samples from z are transformed by $g(z)$. Red dashed-line indicate the transformation function that has higher $\log \det(J_z J_z)$. This transformation induces a higher entropy in the output space and hence better spread out.

invertible function. In addition, we apply the transformation in the dual space rather than directly on the input data. On the other hand, we know $E_p[w_1(w_2, x)] = A(w)$, then for a given distribution q we have, $E_p[(w, x)] = E_z[p_z(w, g_g(z)) H(p_z) + H_g]$ where the entropies provide the bound on the difference between the moments under p and q .

3.3 Bayesian Extension

One benefit of such density estimation is that we can easily extend the model to a Bayesian one where a distribution over parameters given the observations is taken rather than the point estimate. As such, we employ prior distribution over the parameters and the Bayes rule to derive the posterior distribution over the parameters. For the exponential distribution in this paper we use a conjugate prior for parameter w_1 : $p(w_1) = \exp(w_1 - \theta_0 A(w_1))$. The posterior is then computed in a closed-form: then

$$p(w_1 | x) = \exp(w_1 + (w_2, x) - (1 + \theta_0) A(w_1))$$

For w_2 we use the same prior, however, the posterior does not have a closed-form solution. As such, we employ Stochastic Gradient Langevin Dynamics (SGLD) [47]

to sample from this posterior. SGLD is an efficient algorithm that is easy to implement when the gradient of the log of the posterior is easy to compute (as it is in our case where the density is estimated using a deep neural network). SGLD performs Monte Carlo sampling by adding a Gaussian noise with a variance proportionate to the learning rate to the gradient updates of the log posterior. Therefore the predictive distribution for a test instance x is

$$p(x|X) = \int p(x|w)p(w|X)dw \quad (9)$$

where we employ Monte Carlo estimate of this integral along with the samples of the posterior to estimate. Predicting the output in Equation 9 needs to be weighted by the step size in the Langevin dynamics to adjust for the decrease in the step size which in turn decreases the mixing rate as explained in [47].

Note in practice, we found it helps to speed up the “burn-in” process where we use Adam for the first few iterations (in larger datasets), after which we revert to SGD and sample according to SGLD update rule in Algorithm 1. As was shown in [39, 5, 4] such Bayesian approaches can significantly improve the performance due to the ability of the model in exploring the space and averaging parameter values.

3.4. Mode Collapse

Mode collapse is a phenomenon in which function g_g maps all z s to a small number of points in the space of X . This can happen in similar algorithms such as GANs where mode collapse is a major problem. Our approach is less likely to suffer from this phenomenon since we are maximizing the H_g for the generated samples. As shown in Figure 1 we transform the input z to the output space as the local geometry of the output space better spread out because we are enforcing the local geometry of a point mapped in the output space x to encourage higher volume.

The interplay between the entropy of the distribution of z and the local characteristics of the transformation function heavily contribute to the performance of the quality of the samples generated. In particular, in case the distribution of z is Gaussian, maximizing the entropy amount to increasing its corresponding variance. Starting with a large variance and transforming to a spread-out space in p , leads to an estimate of the likelihood that covers all the space with relatively similar value. Hence, the samples generated this way exhibit poor quality and as such under-perform in estimating the correct likelihood. On the other hand, when the initial variance of the distribution of z is small and gradually this variance is increased, the samples generated by g_g cover the mode of the distribution p first and subsequently learns the less frequent samples. In practice, we take the latter approach and start with a small variance for the p_z so that the mode of p is learnt first.

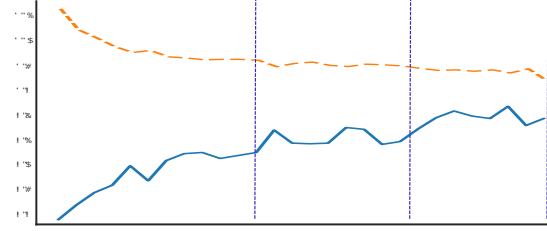


Figure 2. A simple density estimation: the first row shows a multimodal distribution and the distribution of samples from the generator at various stages; and, in the second diagram, the convergence of the estimated density to its true value is shown.

	Modes
DCGAN [37]	99
ALI [14]	16
Unrolled GAN [33]	48.7
Ours	125
Ours Bayesian	142

Table 1. Degree of mode collapse on Stacked-MNIST.

4. Experiments

Our approach provides a path for density estimation while we have the capacity to draw samples from the distribution learned from the dataset. As such, we direct our focus towards various applications where various aspects of our approach is evaluated. We use Algorithm 1 to train our density model in which a generator is also learnt that we can sample from. When the distribution of the output for the generator is too large, computing the Jacobian is expensive. As such, during training we sample a set of dimensions from the output and compute the gradients. To estimate the Jacobian we use Lemma 2 and found the value of ϵ set to the dimension of the Jacobian performs well in practice. In addition, we update the parameter w twice as often as that of θ while regularize the gradients to ensure stable training that converges to an optimum solution.

4.1. Simulation

For the first experiment, we use samples from a Gaussian mixture model to simulate a dataset we are interested in its density model. Each component is a Gaussian distribution with standard deviation 0.1. The generated dataset is shown in Figure 2 on top. We then use a simple two fully connected layer neural network for the generator and (w, x) . In the top row of Figure 2 the generated samples are plotted and as shown over time our approach is able to capture the density model and generate samples from all the modes. Furthermore, at the bottom of Figure 2 the convergence of

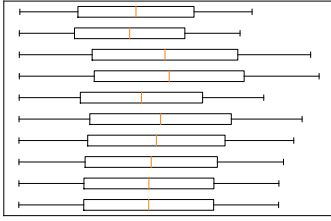


Figure 3. Average posterior probability for each digit sorted from the highest variance to the lowest.

Figure 4. Samples from MNIST sorted based on their density from high to low.

the density estimate to its true value is shown. As observed, as the distribution of the samples from the generator look more similar to the real ones, the estimate of the density (as measured by the expected log-likelihood with respect to the true distribution) approaches the true value.

4.2. Image Generation and Density Estimation

In this section, we examine the quality of the generated images using our approach as well as its density estimation ability. We use DCGAN’s [37] architecture for image generation.

MNIST: To evaluate the effect of the Jacobian on the mode collapse, we have conducted an experiment similar to that of [33] where we select 3 digits at random and build a 3-channel “stacked-MNIST” by concatenating them. Using a pre-trained classifier, we find the number of modes for which the generator produced at least one sample. The results of running this experiment is shown in Table 4. As observed our approach outperforms its counterparts indicating the ability of our approach in avoiding mode-collapse.

We use MNIST, a dataset of handwritten characters, to generate samples from q . We show samples both from maximum log-likelihood and Bayesian. We use the predictive probability obtained from Equation 9 is shown in Figure 5. It is interesting to observe that 1 and 7 are the least likely digits because they have straight lines that have structural differences with other digits. In addition, we compare the likelihood estimated by our approach compared to the-state-of-the-art. The evaluation of log-likelihood (LL) in nats is reported in Table 2. All log-likelihood values were estimated as the mean of 5000 instances drawn randomly from the test set. Since we use the predictive distribution in Equation 9, the likelihood is not as low as when we compute the likelihood directly. Hence, the reported values are stochastic lower bounds on the true value. Figure 4 depicts samples drawn from the generator sorted by the likelihood using

MNIST	$-\log p(x)$
VAE [23]	87.88
IWAE (IW = 50) [9]	86.10
VAE+NF [38]	85.10
ASY-IWAE [49]	85.76
Auxiliary VAE [31]	84.59
DARN [17]	84.13
IWAE+ConvFlow [50]	79.78
Ours	82.39
Ours-Bayesian	83.12

Table 2. MNIST test set negative log-likelihood with generative models (lower is better). These values are estimates since the true density is intractable.

(a) Max LL (b) Bayesian (c) Max LL (d) Bayesian

Figure 5. Samples drawn from the generator using MNIST: 5(a) and 5(b); and, CIFAR-10 5(c) and 5(d). Alternatives (e.g. [24, 41]) can generate visually better quality images.

our Bayesian approach.

CIFAR-10: We have trained our approach on CIFAR-10 dataset [26] composed of 50,000 RGB images of natural scenes. Samples from this dataset is shown in Figure 5. We achieve the inception score of 7.84 using a modified ResNet.

CelebA: While MNIST and CIFAR-10 are smaller dimensions, we evaluate our approach on CelebA [28] with size 256×256 . While our approach performs density estimation, we can generate high-quality images to estimate the normalizer. In this experiment, we utilize the Progressive-GAN [22] architecture and learning procedure to produce images and evaluate their density estimation. We compare the output of our density estimator versus WGAN-GP’s [19] discriminator output as shown in Figure 6. The histogram distribution of the fake samples in WGAN-GP looks very similar to the Normal distribution similar to the prior noise. In addition, as expected with most GAN-based methods, the discriminator value overlaps between real and fake samples indicating the discriminator’s confusion. Our approach on the other hand, shows a better spread histograms. Furthermore, the images in the graph shows samples of generated images with their probability as computed by our approach. It is observed that more typical looks that are more representative of the dataset concentrated in a high density area. In the progressive training, we freeze the update of the noise

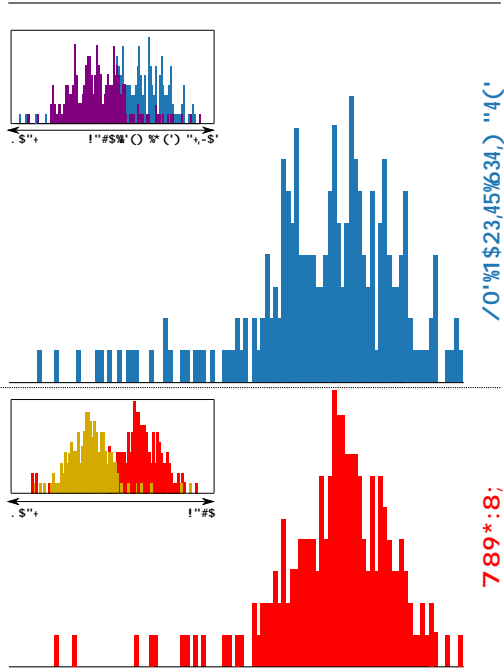


Figure 6. Histogram of the evaluation of $(w, g_g(z))$ for 500 samples generated from the CelebA dataset with 256×256 dimensions (on top) compared to the discriminator value of WGAN-GP (on bottom). On the left corner of each plot, the distribution of the $(w, g_g(z))$ is shown vs. the output of the discriminator in the WGAN-GP. As shown, discriminator in WGAN-GP finds a distribution that has around half overlap between reals and the fakes.

entropy for the higher quality images to avoid changing the networks significantly. The Frechet Inception Distance (FID) computed from 50K images is 5.96 for this dataset.

4.3. Uncertainty in Visual Question Answering

We utilize our approach in the visual dialog problem introduced in [10]. The task is to design an agent capable of replying to questions based on image content and history of a dialog with a human user. The objective is to generate responses to questions that are most likely to be given by a human. The transformation function g_g is responsible for generating a suitable answer for the user’s question, given the previous history of agent-user conversations, the user question and the visual content (rather than just a noise distribution in the previous experiment). Then, the likelihood of the human responses is higher.

Our approach is well embedded into a larger dialog system built upon that of [42, 29]. We use a hierarchical setup where a neural network for sequence models (LSTM in our

case) encodes the word sequence and another one captures the context through historical dependencies of the conversations. A common practice is to use an attention mechanism with the encoder to implicitly identifying parts of dialog history relevant to the current question [42, 10]. It is expected that a similar mechanism is able to localize the relevant regions in the image consistent with the attended history. Since the output is discrete, we use Gumbel-Max trick [21, 32] which illustrates the ability of our approach in maximizing likelihood for discrete data.

This task is specifically suitable for Bayesian formulation of our approach since the space in which the function g_g formulates the responses is large and as such this causes uncertainty in the prediction. Hence, our approach leverages this uncertainty in the predictive distribution when responding by providing “I don’t know” statements when the predictive variance is high.

The visual dialog [10] dataset we use was collected by pairing two Amazon Mechanical Turk users conversing about an image. The “questioner” user is shown the captions of the image (from COCO dataset [27]) while the image itself remains hidden. The “answerer” user sees the image and provides responses about the image to the questioner. This dialog is carried out for 10 rounds on the COCO dataset leading to 83,000 training dialogs and 40,000 validation dialog. Following the baseline, the likelihood score is used in ranking amongst 100 candidate answers.

We set the parameter for the Gumbel temperature to 0.1. Our LSTMs are single layer with 512-dimensional hidden states. Further, we use VGG-19 [44] to represent images and Adam optimization with the learning-rate of $4e-4$. For the Bayesian case, we use SGD and set the learning rate to decrease from $1e-3$ to $1e-4$. Following the problem setup in [29], we pre-train our transformation function g_g by maximizing the score it generates compared to the ground-truth. Similarly, we pre-train the density estimator using samples from 100 rounds of the question-answers such that the score of the ground-truth is higher than non-matching responses in the training set. We pre-train both for 20 epochs. Moreover, we regularize the latent space from which the answers are generated with its entropy (we assumed the latent space is a zero-mean Gaussian). We draw 5 samples from which we pick the most likely.

In Table 3 we provide the results of running our approach on the visual dialog dataset compared to the state-of-the-art GAN methods and other baselines in [43, 29]. Models are evaluated on standard retrieval metrics: mean rank, recall @k, and mean reciprocal rank (MRR) of human response. The results from the pre-training stage is indicated as HCAIE in the table and as is shown GAN training improves the performance. In addition, our approach outperforms the baseline with a good margin and is more stable in training due to its regularization in the output. Moreover,

Model	Discriminative					Generative				
	MRR	R@1	R@5	R@10	Mean	MRR	R@1	R@5	R@10	Mean
HieCoAtt [30]	0.5788	43.51	74.49	83.96	5.84	-	-	-	-	-
LF [10]	0.5807	43.82	74.68	84.07	5.78	0.5199	41.83	61.78	67.59	17.07
HRE [10]	0.5868	44.82	74.81	84.36	5.66	0.5242	42.28	62.33	68.17	16.79
MN [10]	0.5965	45.55	76.22	85.37	5.46	0.5259	42.29	62.85	68.88	17.06
HCIAE [29]	0.6140	47.73	77.50	86.35	5.15	0.5386	44.06	63.55	69.24	16.01
GAN1 [29]	0.2177	8.82	32.97	52.14	18.53	0.5298	43.12	62.74	68.58	16.25
GAN2 [29]	0.6050	46.20	77.92	87.20	4.97	0.5459	44.33	65.05	71.40	14.34
WGAN	0.5524	43.77	67.75	72.32	13.18	0.5306	43.49	62.38	67.73	17.15
Ours	0.6125	47.17	77.25	87.65	4.82	0.5448	44.37	64.34	71.81	14.93
Ours Bayesian	0.6204	47.65	78.91	88.30	4.61	0.5513	45.03	65.25	71.93	14.01

Table 3. Results on Visual Dialog dataset with Late Fusion (LF), Hierarchical Recurrent Encoder (HRE), Memory Network (MN). GAN1 is devised by using generated fake samples in the discriminator and GAN2, in addition, includes negative answers. Higher values are better except for the mean. Best numbers are boldfaced.

Q: is the train old? A: i think so Sample A: yes (0.03 ± 0.2)	Q: is this in color ? A: yes Sample A: yes (0.1 ± 0.05)	Q: what vegetables are there? A: carrots, cauliflower, broccoli Sample A: carrots , carrots , and cucumbers (0.1 ± 0.01)
Q: is the train moving? A: no but it does have some steam coming Sample A: no (0.02 ± 0.1)	Q: how old does the boy look? A: he is not facing me, but maybe Sample A: 3 (0.1 ± 0.4)	Q: what color is the table ? A: dark brown Sample A: white (0.1 ± 0.02)

Table 4. Qualitative comparison from the Visual Dialog dataset. (Q) is the question, (A) is the ground-truth human response, (Sample A): is the generated answer from the generator. **Red** response indicates lack of confidence, which will be substituted by “I don’t know”.

we devised the Bayesian variant (as in Section) to compute the predictive distribution and pick the one with the highest confidence according to Equation 9 (highest mean plus standard deviation). Since the space of answers generated is typically large even with pre-training, performance of the generator and discriminator can deteriorate. This is because the noisy samples from the generator lead to decreased the performance in the discriminator.

In Table 4 we show samples from the visual dialog dataset. We evaluate the generated answers by sampling the discriminator function and evaluating the variance of the prediction (likelihood of the generated answer being human-response in the log space). As is shown in the table, we observe the variance of the prediction is generally higher for the wrong responses. We can use a simple thresholding on the predictive variance to determine the answers to be substituted by “I don’t know”. For instance, in Table 4 we show an example where a highly uncertain answer is substituted. Even though for frequent answers such as yes/no responses the variances are less reliable. In addition, for longer sentences, the variances are generally higher. This is

expected due to the potential diversity.

5. Conclusion

In this paper, we introduced generative adversarial density estimator. Our approach estimates the density of data using a lower bound on its normalizer. We formalized the problem as a maximum likelihood for the density estimator in which a transformation function generates samples for approximating the normalizer. We showed that having a density model and the prior on the parameters, we are able to build a Bayesian alternative using which we can measure our confidence in the likelihood. Our experiments on challenging applications such as image density or visual dialog where both the likelihood and confidence in predictions are crucial shows the effectiveness of our approach. Moreover, the samples drawn from the estimated density is less susceptible to mode collapse which is a desired property for generative models.

We believe our approach opens new avenues for further research in density estimation and its marriage with adversarial models.

References

- [1] E. Abbasnejad, S. Sanner, E. V. Bonilla, and P. Poupart. Learning community-based preferences via dirichlet process mixtures of gaussian processes. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13*, pages 1213–1219, 2013. **2**
- [2] E. Abbasnejad, J. Shi, and A. van den Hengel. Deep lipschitz networks and dudley GANs, 2018. **2**
- [3] I. Abbasnejad, S. Sridharan, D. Nguyen, S. Denman, C. Fookes, and S. Lucey. Using synthetic data to improve facial expression analysis with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1609–1618, 2017. **2**
- [4] M. E. Abbasnejad, A. Dick, Q. Shi, and A. van den Hengel. Active learning from noisy tagged images. 2018. **5**
- [5] M. E. Abbasnejad, Q. Shi, I. Abbasnejad, A. van den Hengel, and A. R. Dick. Bayesian conditional generative adversarial networks. *CoRR*, abs/1706.05477, 2017. **5**
- [6] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *ArXiv e-prints*, Jan. 2017. **2**
- [7] A. Ben-Israel. The change-of-variables formula using matrix volume. *SIAM Journal on Matrix Analysis and Applications*, 21(1):300–312, 1999. **3**
- [8] C. Boutsidis, P. Drineas, P. Kambadur, E.-M. Kontopoulou, and A. Zouzias. A randomized algorithm for approximating the log determinant of a symmetric positive definite matrix. *Linear Algebra and its Applications*, 533, 2017. **3**
- [9] Y. Burda, R. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. 09 2015. **6**
- [10] A. Das, S. Kottur, K. Gupta, A. Singh, D. Yadav, J. M. Moura, D. Parikh, and D. Batra. Visual Dialog. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. **2, 7, 8**
- [11] P. J. Diggle and R. J. Gratton. Monte Carlo methods of inference for implicit statistical models. *Journal of the Royal Statistical Society.*, pages 193–227, 1984. **1**
- [12] L. Dinh, D. Krueger, and Y. Bengio. Nice: Non-linear independent components estimation. *CoRR*, abs/1410.8516, 2014. **1, 4**
- [13] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real nvp. 2016. **1, 4**
- [14] V. Dumoulin, I. Belghazi, B. Poole, O. Mastropietro, A. Lamb, M. Arjovsky, and A. Courville. Adversarially learned inference. In *International Conference on Learning Representation*, 2017. **5**
- [15] M. Ehsan Abbasnejad, A. Dick, and A. van den Hengel. Infinite variational autoencoder for semi-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5888–5897, 2017. **2**
- [16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *International Conference on Neural Information Processing Systems*, 2014. **1, 2**
- [17] K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra. Deep autoregressive networks. In *The International Conference on Machine Learning (ICML)*, 2014. **1, 6**
- [18] A. Grover, M. Dhar, and S. Ermon. Flow-gan: Combining maximum likelihood and adversarial learning in generative models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18)*, the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018, pages 3069–3076, 2018. **2, 4**
- [19] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017. **2, 6**
- [20] M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Likelihood-free inference via classification. *Statistics and Computing*, 2018. **1**
- [21] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *CoRR*, 11 2016. **7**
- [22] T. Karras, T. Aila, S. Laine, and J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. In *The International Conference on Learning Representations (ICLR)*, 2017. **1, 2, 6**
- [23] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *The International Conference on Learning Representations (ICLR)*, 2014. **1, 2, 6**
- [24] A. Kolesnikov and C. H. Lampert. PixelCNN models with auxiliary variables for natural image modeling. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1905–1914, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. **6**
- [25] V. R. Konda and J. N. Tsitsiklis. Convergence rate of linear two-time-scale stochastic approximation. *Ann. Appl. Probab.*, 14(2):796–819, 05 2004. **3**
- [26] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009. **6**

- [27] T.-Y. Lin, M. Michael, B. Serge, H. James, P. Pietro, R. Deva, D. Piotr, and Z. C. Lawrence. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*. Springer International Publishing, 2014. 7
- [28] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015. 6
- [29] J. Lu, A. Kannan, , J. Yang, D. Parikh, and D. Batra. Best of both worlds: Transferring knowledge from discriminative learning to a generative visual dialog model. *NIPS*, 2017. 7, 8
- [30] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16*, pages 289–297, USA, 2016. Curran Associates Inc. 8
- [31] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML’16*, pages 1445–1454, 2016. 6
- [32] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *CoRR*, 2016. 7
- [33] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein. Unrolled generative adversarial networks. *International Conference on Learning Representations*, 2016. 5, 6
- [34] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE*, 2017. 1
- [35] S. Nowozin, B. Cseke, and R. Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization. *ArXiv e-prints*, 2016. 2
- [36] A. v. d. Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu. Conditional image generation with pixelcnn decoders. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16*, pages 4797–4805, 2016. 1, 2
- [37] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, 2015. 5, 6
- [38] D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning, ICML’15*, pages 1530–1538. JMLR.org, 2015. 2, 4, 6
- [39] Y. Saatchi and A. G. Wilson. Bayesian GAN. In *Advances in Neural Information Processing Systems*, 2017. 5
- [40] R. Salakhutdinov and G. E. Hinton. Deep boltzmann machines. In *International conference on artificial intelligence and statistics, AISTATS’09*, 2009. 1, 2
- [41] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *The International Conference on Learning Representations (ICLR)*, 2017. 2, 6
- [42] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings AAAI*, pages 3776–3783, 2016. 7
- [43] R. Shetty, M. Rohrbach, L. A. Hendricks, M. Fritz, and B. Schiele. Speaking the same language: Matching machine to human captions by adversarial training. *CoRR*, abs/1703.10476, 2017. 7
- [44] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv*, 2014. 7
- [45] L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. *ArXiv e-prints*, Nov. 2015. 1
- [46] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. 2017. 2
- [47] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 681–688, 2011. 4, 5
- [48] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017. 1
- [49] G. Zheng, Y. Yang, and J. Carbonell. Likelihood almost free inference networks. 11 2017. 6
- [50] G. Zheng, Y. Yang, and J. Carbonell. Convolutional normalizing flows. In *ICML Workshop on Theoretical Foundations and Applications of Deep Learning*, 2018. 6