Generate High Fidelity Images With Generative Variational Autoencoder

Abhinav Sagar
Vellore Institute of Technology, Vellore, Tamil Nadu, India

Abstract

In this work, we address the problem of blurred images which are often generated using Variational Autoencoders and the problem of mode collapse in Generative Adversarial Networks using a single model architecture. We use the encoder of VAE as it is while replacing the decoder with a discriminator. The encoder is fed data from a normal distribution while the generator is fed that from a gaussian distribution. The combination from both is then fed to a discriminator which tells whether the generated images are correct or not. We tested the model on 3 different datasets increasing in complexity MNIST, fashion MNIST and TCIA Pancreas CT dataset. On training the model for 300 iterations, it was able to generate much sharper images as compared to those of VAEs. This work is potentially very exciting as we are able to combine the advantages of generative models and inference models in a bayesian approach. As there is a shortage of medical data, this approach could be revolutionary given that we are able to reason from the bayesian approach taking into account the uncertainty of generation.

Keywords: Variational Autoencoder, Deep Learning, Neural Network, Computer Vision, Generative Model

1. Introduction

Recently, convolutional neural networks have achieved great success in computer vision problems including image classification, object detection, pose estimation, semantic segmentation etc. However, the training of deep neural networks requires hundreds or even thousands of images. In many real world problems, lack of datasets often hinders the progress. Hence it becomes imperative to create additional training data. One way which is often used is using data augmentation techniques in which the images already present in the dataset are rotated, shifted, scaled, adding noise etc is done. Using this, the size of the dataset can be increased considerably to multiple times the size of the original data.

Another area which is actively researched is using generative adversarial networks for image generation. Using this technique, new images can be generated by training on the existing images present in the dataset. The new images are realistic but different from the original data. There are two main approaches of using data augmentation using GANs: image to image translation and sampling from random distribution.

Using the first approach, training is relatively easy as it is done with the guidance of another dataset and the quality of generated images is comparable to that of real images. However, the drawback is that it requires extensive training data, and the generated outputs are very similar in shape to those which are

already present in the dataset thus defeating the purpose to some extent. The second method can generate completely new images with more variability by learning the data distribution itself. However, the drawback with this training is that the training is often unstable and requires much more time tuning many of the parameters involved. With recent advances in hyperparameter optimization using bayesian approaches like the gaussian process, however this approach has been used to some degree of success.

Another approach for image generation uses variational autoencoders. This architecture contains an encoder which is also known as generative network which takes a latent encoding as input and outputs the parameters for a conditional distribution of the observation. The decoder is also known as an inference network which takes as input an observation and outputs a set of parameters for the conditional distribution of the latent representation. During training VAEs use a concept known as reparameterization trick in which sampling is done from a gaussian distribution. This sample is multiplied by standard deviation of the distribution and added to the mean of the distribution. Using VAEs for image generation is an active area of research lately.

2. Related Work

Lately there has been a surge of paper published on Generative models . Many models including Pixel RNNs (Van den Oord et al, 2016), Pixel CNNs (Van den Oord et al, 2016), Plug and Play generative networks (Nguyen et al, 2016) have been worked on along with their variants. However the main two generative model architecture revolves around Generative Adversarial Networks (GANs) (Goodfellow et al, 2014) and Variational Autoencoders (VAEs) (Kingma & Welling et al, 2013).

Both GANs and VAEs have their own advantages and disadvantages. GANs tend to produce sharper images which look more realistic i.e. closer to the images present in the dataset. On the other hand VAEs have both the properties i.e. can be used both as a generative model and an inference model. Also there have been actively work done to add inference capability to GANs (Kingma & Welling et al, 2013) but still it is in its infancy stage compared to VAEs which have been using the bayesian approach in gaussian process, variational inference, bayesian deep learning etc. The other advantage of using VAEs is that they produce better log likelihoods (Wu et al, 2016) which is important considering the measure which evaluates the variety of image quality generated.

The problem with VAEs is that they can't produce sharp images like GANs comes down to the fact that inference models during training don't capture true posterior distribution. Some of the recent works using more expressive priors (Kingma et al, 2016) have been researched actively to some degree of success. Some of the work have also tried to combine both GANs and VAEs architectures by using the best of both worlds (Makhzani et al, 2015; Larsen et al, 2015). To learn the loss function, this architecture changed the decoder from VAEs to a GAN discriminator. The blurriness comes from the reconstruction loss in decoder while training VAEs. Since the loss term, it now uses one from a discriminator hence the model is able to create sharp images similar to that produced by GANs.

Another work on adversarial autoencoders to generate images) (Makhzani et al, 2015) uses the concept of replacing the Kullback-Leibler regularization term that appears in the training objective for VAEs with an adversarial loss that encourages the posterior to be close to the prior over the latent variables. In this manner adversarial autoencoders work in a similar way to the past approaches by learning to maximize the maximum-likelihood objective.

In this paper, we present generative variational autoencoders, a technique for training Variational Autoencoders to create high fidelity images similar to that produced using GANs. We trained and tested our model on three different datasets increasing in complexity in order MNIST, fashion MNIST and TCIA Pancreas CT dataset.

3. Background

Our model is a variant of original VAE architecture. VAEs are a class of generative models which are parametric in nature. They are specified by a prior over the latent variables and our goal is to compute the posterior distribution given the likelihood. VAEs are represented mathematically using the below equation where the first term denotes the KL divergence between the original and the posterior distributions (Kingma & Welling et al, 2013). The second term denotes the reconstruction error of obtaining the sample from the latent distribution.

$$\log p_\theta(x) \geq -\mathrm{KL}(q_\phi(z|x), p(z)) + \mathrm{E}_{q_\phi(z|x)} \log p_\theta(x|z)$$

The right hand side in the equation above denoted Evidence Lower Bound (ELBO) which needs to be maximized. The goal is to optimize the max likelihood. The problem is that it requires solving the integral which is intractable in nature. Hence there is a need to convert the above problem to an optimization problem. This is done using variational inference techniques as shown in the equation below.

$$\max_\theta \max_\phi \mathrm{E}_{p_\mathcal{D}(x)}[-\mathrm{KL}(q_\phi(z|x), p(z)) + \mathrm{E}_{q_\phi(z|x)} \log p_\theta(x|z)]$$

4. Method

4.1 Dataset

In this work, we have used 3 publicly available datasets:

- MNIST - This is a large dataset of handwritten digits which has been used successfully for training image classification and image processing algorithms. It contains 60,000 training images and 10,000 test images. A sample of the dataset is shown in Fig 1.

Fig 1: MNIST dataset

- Fashion MNIST - This dataset is also similar to MNIST with 60,000 training images and 10,000 test images. Each example is a 28x28 grayscale image which is labelled into one of the 10 classes of fashion wear like trouser, top, sandal etc. A sample of the dataset is shown in Fig 2.

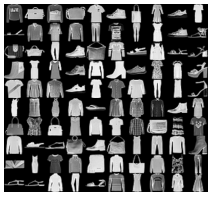


Fig 2: Fashion MNIST dataset

- TCIA Pancreas CT - The National Institutes of Health Clinical Center performed 82 abdominal contrast enhanced 3D CT scans. The CT scans have resolutions of 512x512 pixels with varying pixel sizes and slice thickness between 1.5 − 2.5 mm. A sample of the dataset is shown in Fig 3.

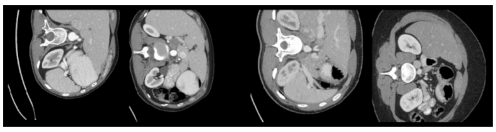


Fig 3: TCIA Pancreas CT dataset

4.2 Model Architecture

The main challenge with GANs is the mode collapse problem i.e. the generated images are quite similar to each other and there is not enough variety in the images generated. On the other hand, the main challenge with VAEs is that they are not able to generate sharp images. However since VAEs don't have the mode collapse problem, hence we reasoned and combined both the architectures ie using the encoder while replacing the decoder with a discriminator. The architecture which we propose handles both of the cases and gets the best of both worlds.

In this work, we show how instead of inference made in the way shown in original VAE architecture, we can use add the error vector to the original data and multiply by standard distribution. The new term goes to the encoder and gets converted to the latent space. In the decoder, similarly the error vector gets added to the latent vector and multiplied by standard deviation. In this manner we use the encoder of VAE in a manner similar to that in the original VAE. While we replace the decoder with a discriminator and hence change the loss function accordingly.

The comparison between model architectures of VAE and our architecture is shown in Fig 4. Our architecture can be seen both as an extension of VAE as well as that of GAN. Reasoning it as the former is easy as this requires a change in loss function for decoder, while the latter can be made by recalling the fact that GAN essentially works on the concept of zero sum game maintaining Nash Equilibrium between the generator and discriminator. In our case, both the encoder from VAE and discriminator from GAN are playing zero sum game and are competing with each other. As the training proceeds, the loss is decreasing in both the cases until it stabilizes.

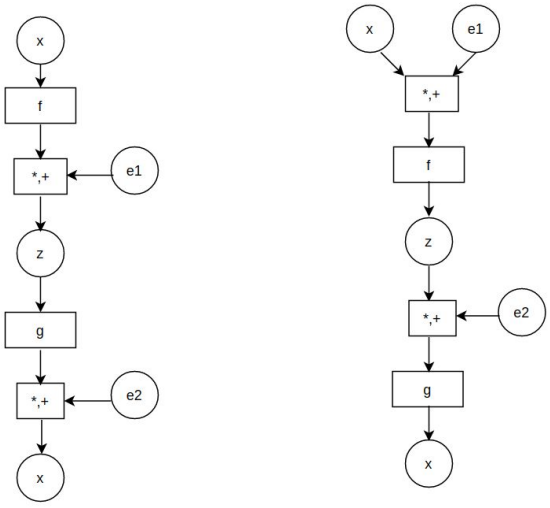


Fig 4: Standard VAE vs our Model

Our discriminator and encoder networks have four 3D convolution layers, each of which uses 3×3 filters. Since the output from the last layer has to be a single value for the discriminator and a vector for the encoder, output channel sizes are set accordingly. We use Batch Normalization and Leaky Rectified Linear Unit (LeakyReLU) layers after each layer. In training, we found that our architecture suffers from instability during training. Hence we tried different loss functions and finally settled with WGAN loss function which measures Wasserstein distance between both distributions. Also we used the gradient penalty term to stabilize the training.

Our loss function has a total for 3 loss terms. While training, the encoder and the generator are considered as one network. Thus, we sum up the loss functions of the two networks in the order encoder-generator, discriminator as one and train the networks.

In our model architecture, two latent vectors are sampled one from normal distribution and the other from gaussian distribution. The one from normal distribution is fed to the encoder while the one from gaussian distribution is fed to the generator. The outputs from both the vectors are in turn fed to the discriminator to tell whether the generated image is real or not. Hence our architecture can be separated into 3 different parts, generator, encoder and discriminator. The model architecture is shown in Fig 5.

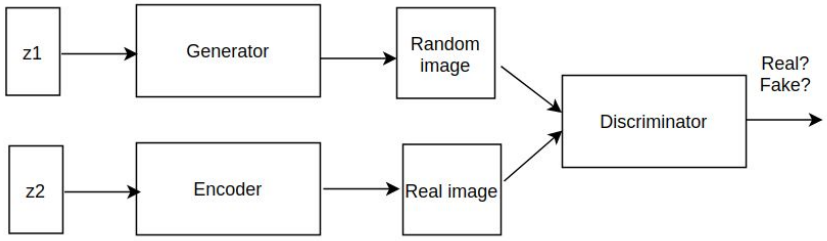


Fig 5: Our model architecture

4.3 Algorithm

Next we present our algorithm which is trained using Stochastic Gradient Descent (SGD):

Algorithm: Generative Variational Autoencoder (GVA)

---

i ← 0
while not converged do
$Sample\{x^{(1)}, \ldots, x^{(m)}\}$ from data distrib. $p_{\mathcal{D}}(x)$
$Sample\{z^{(1)}, \ldots, z^{(m)}\}$ from prior $p(z)$
Sample $\{\epsilon^{(1)}, \ldots, \epsilon^{(m)}\}$ from $\mathcal{N}(0, 1)$

$$g_\theta \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\theta \log p_\theta \left(x^{(k)} | z_\phi \left(x^{(k)}, \epsilon^{(k)}\right)\right)$$

$$g_\phi \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\phi \log p_\phi \left(x^{(k)} | z_\theta \left(x^{(k)}, \epsilon^{(k)}\right)\right)$$

$PerformSGD - updatesfor\theta, \phi$
i ← i + 1
End while loop

The generator and discriminator layerwise architecture details are shown in Table 1 and Table 2 respectively. We have denoted ResNet block as consisting of 3 layers - convolutional, max pooling layer, 30% dropouts in between the layers and batch normalization layers.

Table 1: Generator architecture details

Layer	Output Size	Filter
Fully Connected	256 × 32 × 32	512 → 256 × 32 × 32
ResNet Block	256 × 32 × 32	512 → 256 → 256
ResNet Block	256 × 32 × 32	256 → 256 → 256
Upsampling	256 × 64 × 64	-
ResNet Block	128 × 64 × 64	256 → 128 → 128
ResNet Block	128 × 64 × 64	128 → 128 → 128
Upsampling	128 × 128 × 128	-
ResNet Block	64 × 128 × 128	128 → 64 → 64
ResNet Block	64 × 128 × 128	64 → 64 → 64

Conv2D	3 × 128 × 128	64 → 3

Table 2: Discriminator architecture details

Layer	Output Size	Filter
Conv2D	64 × 128 × 128	3 → 64
ResNet Block	64 × 128 × 128	64 → 64 → 64
ResNet Block	128 × 128 × 128	64 → 64 → 128
AvgPool2D	128 × 64 × 64	-
ResNet Block	128 × 64 × 64	128 → 128 → 128
ResNet Block	256 × 64 × 64	128 → 128 → 256
AvgPool2D	256 × 32 × 32	-
Fully Connected	256 × 32 × 32	256 × 32 × 32 → 1000

5. Experiments

Our experiments are conducted on an NVIDIA Titan GPU. Code is implemented in Python programming language using the Pytorch deep learning library. For training the model, the Adam optimizer is used with a learning rate of 0.0001 for all three networks, and the size of mini-batch is set to 32. All the generated samples are generator outputs from random latent vectors. We normalize all data into the range [-1, 1].

We have used two evaluation metrics to measure the performance of our model. First of them measures the distribution distance between the real and generated samples with maximum mean discrepancy (MMD) scores . The second metric evaluates the generation diversity with multi-scale structural similarity metric (MS-SSIM). Table 3. compares MMD and MS-SSIM scores with previous architectures.

Table 3: Quantitative results on MNIST

	MMD×10−4	MS-SSIM
WGAN-GP	0.327	0.996
VAE-GAN	0.075	0.972
α-GAN	0.131	0.843
Ours	0.068	0.818

We also tried varying the latent variable size and to see if any correlation is present and we found that the latent variable is indeed very important in getting the best results. We noticed that the model with a small latent vector size of 100 suffers from severe mode collapse. The best results can be obtained using a

moderately large latent vector size. Table 4 compares the effect of different latent variable sizes on the MMD and MS-SSIM scores respectively.

As can be seen, latent variable size with value 1000 produces the best results of those being compared. Both at low and high latent variable size mode collapse is seen which is one of the main challenges faced with GANs. The results are consistent with both of the evaluation metrics ie MMD and SSIM.

Table 4: Effect of latent vector on MMD and SSIM on MNIST

Latent variable size	MMD×10−4	MS-SSIM
z 100	0.104	0.856
z 500	0.085	0.821
z 1000	0.068	0.818
z 2000	0.074	0.844

There are four evaluation metrics which have been used in the literature for testing how good the model is performing. These are log-likelihood, reconstruction error, ELBO, KL divergence.

The log-likelihood is calculated by finding the parameter that maximizes the log-likelihood of the observed sample. The reconstruction error is the distance between the original data point and its projection onto a lower-dimensional subspace. The optimization problem used in our model uses KL divergence error which is intractable hence we maximize ELBO instead of minimizing the KL divergence. KL divergence is a measure of how similar the generated probability distribution is to the true probability distribution.

All of the above metrics are useful but can be misleading at times specially when used in isolation. Hence it is important to use them together to get a true picture of the results. The comparison using these evaluation metrics of our model on MNIST dataset with the original VAE architecture is shown in Table 5.

Table 5: Comparison of results in original VAE vs our architecture on MNIST

	VAE	Ours
log-likelihood	-1.568	-1.353
reconstruction error	88.5 × 10−3	4.27 × 10−3
ELBO	-1.697	-1.404
KL divergence	0.165	0.046

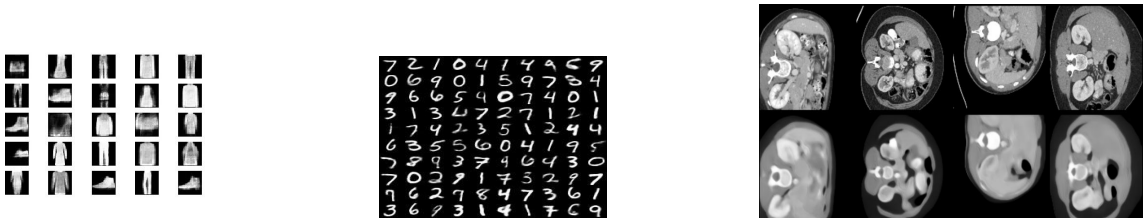
Next we also compare our log probability distribution value with those obtained by others which is shown in Table 6. The log probability distribution is an important evaluation metric in the sense that it shows the diversity of the samples generated.

Table 6: Results fo independent samples for a model trained on MNIST

Method	log p(x) ≥
VAE + NF (T=80)	−85.1
VAE + HVI (T=16)	−88.3
conv VAE + HVI (T=16)	−84.1
VAE + VGP (2hl)	−81.3
DRAW + VGP	−79.9
VAE + IAF	−80.8
Ours	-82.2

6. Results

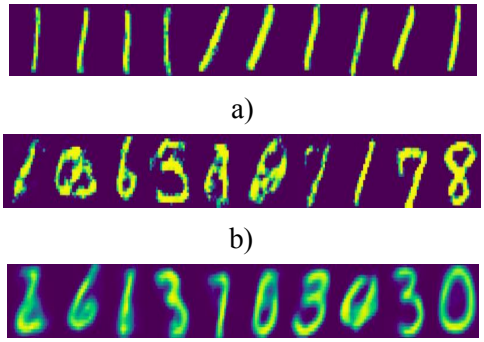
In this section, we present the generated images on all the 3 datasets used for validation. The images were trained for 1000 iterations both in the cases of MNIST and fashion MNIST while was trained for 300 iterations on TCIA Pancreas CT dataset. The generated images are shown in Fig 6.



a) b) c)

Fig 6: Generated images a) MNIST b) Fashion MNIST c) TCIA Pancreas CT

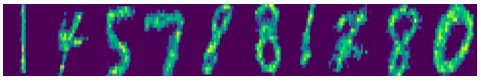
Next we compare our results with previous state of the art models on MNIST dataset in Fig 7.



a)

b)

c)



d)

Fig 7: Generated MNIST images a) GAN b) WGAN c) VAE d) GVAE

7. Conclusions

In this paper, we presented a new training procedure for Variational Autoencoders based on generative modelling. This allows us to make the inference model much more flexible, effectively allowing it to represent almost any posterior distributions over the latent variables. The architecture was trained and tested on 3 different publicly available datasets and was able to generate handwritten digits, fashion clothes and brain tumour images with much higher fidelity when compared to standard VAEs. Using generative model approaches to generate additional training data especially in fields like biomedical imaging could be revolutionary as there is a shortage of medical data for training deep convolutional neural network architectures. Artificially creating additional data of high fidelity could be used for training more robust deep learning algorithms.

References

1. Yu, B., et al.: 3d cgan based cross-modality mr image synthesis for brain tumor segmentation. In: 2018 IEEE 15th International Symposium on Biomedical Imaging. pp. 626–630 (2018).

2. Chen, Xi, Kingma, Diederik P, Salimans, Tim, Duan, Yan, Dhariwal, Prafulla, Schulman, John, Sutskever, Ilya, and Abbeel, Pieter. Variational lossy autoencoder. arXiv preprint arXiv:1611.02731, 2016.

3. Holger R. Roth, Amal Farag, Evrim B. Turkbey, Le Lu, Jiamin Liu, and Ronald M. Summers. (2016). Data From Pancreas-CT. The Cancer Imaging Archive.

4. Mart´ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In USENIX Symposium on Operating Systems Design and Implementation, volume 16, pp. 265–283, 2016.

5. Forest Agostinelli, Matthew Hoffman, Peter Sadowski, and Pierre Baldi. Learning activation functions to improve deep neural networks. arXiv preprint arXiv:1412.6830, 2014.

6. Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International Conference on Machine Learning, pp. 448–456, 2015.

7. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

8. . Shin, H.C., et al.: Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In: Simulation and Synthesis in Medical Imaging, pp. 1–11 (2018)

9. Rosca, M., et al.: Variational Approaches for Auto-Encoding Generative Adversarial Networks.
10. P Kingma, D., Welling, M.: Auto-encoding variational bayes. In: International Conference on Learning Representations (2014)

11. Odena, A., Olah, C., Shlens, J.: Conditional image synthesis with auxiliary classifier GANs. In: International Conference on Machine Learning. pp. 2642–2651 (2017).

12. Larsen, A.B.L., et al.: Autoencoding beyond pixels using a learned similarity metric. In: International Conference on Machine Learning. pp. 1558–1566 (2016).

13. Han, C., et al.: Gan-based synthetic brain mr image generation. In: 2018 IEEE 15th International Symposium on Biomedical Imaging. pp. 734–738 (2018).

14. Goodfellow, I.J., et al.: Generative adversarial nets. In: Advanced in Neural Information Processing Systems. pp. 2672–2680 (2014)

15. Dar, S.U., et al.: Image synthesis in multi-contrast MRI with conditional generative adversarial networks. IEEE Transactions on Medical Imaging pp. 1–1 (2019)

16. Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. The elements of statistical learning, volume 1. Springer series in statistics Springer, Berlin, 2001.

17. He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

18. Huszar, Ferenc. Variational inference using implicit distri- ´ butions. arXiv preprint arXiv:1702.08235, 2017.

19. Kingma, Diederik P, Salimans, Tim, and Welling, Max. Improving variational inference with inverse autoregressive flow. arXiv preprint arXiv:1606.04934, 2016.

20. Kucukelbir, Alp, Ranganath, Rajesh, Gelman, Andrew, and Blei, David. Automatic variational inference in stan. In Advances in neural information processing systems, pp. 568–576, 2015.

21. Larsen, Anders Boesen Lindbo, Sønderby, Søren Kaae, and Winther, Ole. Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300, 2015.

22. Li, Yingzhen and Liu, Qiang. Wild variational approximations. In NIPS workshop on advances in approximate Bayesian inference, 2016.

23. Liu, Qiang and Feng, Yihao. Two methods for wild variational inference. arXiv preprint arXiv:1612.00081, 2016.

24. Maaløe, Lars, Sønderby, Casper Kaae, Sønderby, Søren Kaae, and Winther, Ole. Auxiliary deep generative models. arXiv preprint arXiv:1602.05473, 2016.

25. Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, and Goodfellow, Ian. Adversarial autoencoders. arXiv preprint arXiv:1511.05644, 2015.

26. Nguyen, Anh, Yosinski, Jason, Bengio, Yoshua, Dosovitskiy, Alexey, and Clune, Jeff. Plug & play generative networks: Conditional iterative generation of images in latent space. arXiv preprint arXiv:1612.00005, 2016.

27. Nowozin, Sebastian, Cseke, Botond, and Tomioka, Ryota. f-gan: Training generative neural samplers using variational divergence minimization. arXiv preprint arXiv:1606.00709, 2016.

28. Poole, Ben, Alemi, Alexander A, Sohl-Dickstein, Jascha, and Angelova, Anelia. Improved generator objectives for gans. arXiv preprint arXiv:1612.02780, 2016.

29. Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015.

30. Ranganath, Rajesh, Tran, Dustin, Altosaar, Jaan, and Blei, David. Operator variational inference. In Advances in Neural Information Processing Systems, pp. 496–504, 2016.

31. Rezende, Danilo Jimenez and Mohamed, Shakir. Variational inference with normalizing flows. arXiv preprint arXiv:1505.05770, 2015.

32. Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. arXiv preprint arXiv:1401.4082, 2014.

33. Salimans, Tim, Kingma, Diederik P, Welling, Max, et al. Markov chain monte carlo and variational inference: Bridging the gap. In ICML, volume 37, pp. 1218–1226, 2015.

34. Tran, Dustin, Ranganath, Rajesh, and Blei, David M. The variational gaussian process. arXiv preprint arXiv:1511.06499, 2015.

35. Van den Oord, Aaron, Kalchbrenner, Nal, Espeholt, Lasse, Vinyals, Oriol, Graves, Alex, et al. Conditional image generation with pixelcnn decoders. In Advances In Neural Information Processing Systems, pp. 4790–4798, 2016.

36. Van den Oord, Aaron van den, Kalchbrenner, Nal, and Kavukcuoglu, Koray. Pixel recurrent neural networks. arXiv preprint arXiv:1601.06759, 2016.

37. Wu, Yuhuai, Burda, Yuri, Salakhutdinov, Ruslan, and Grosse, Roger. On the quantitative analysis of decoder-based generative models. arXiv preprint arXiv:1611.04273, 2016.