Generate High Resolution Images With Generative Variational Autoencoder

Abhinav Sagar[*]
Vellore Institute of Technology
Vellore, Tamil Nadu, India
`abhinavsagar4@gmail.com`

Abstract

In this work, we present a novel neural network to generate high resolution images. We replace the decoder of VAE with a discriminator while using the encoder as it is. The encoder uses data from a normal distribution while the generator from a gaussian distribution. The combination from both is given to a discriminator which tells whether the generated images are correct or not. We evaluate our network on 3 different datasets: MNIST, LSUN and CelebA-HQ dataset. Our network beats the previous state of the art using MMD, SSIM, log likelihood, reconstruction error, ELBO and KL divergence as the evaluation metrics while generating much sharper images. This work is potentially very exciting as we are able to combine the advantages of generative models and inference models in a principled bayesian manner.

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 is 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 successful.

---

[*]Website of author - `https://abhinavsagar.github.io/`

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 (Oord et al., 2016), Pixel CNNs (Van den Oord et al., 2016), Plug and Play generative networks (Nguyen et al., 2017) 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 and Welling, 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 and Welling, 2013) but still it is in its infancy stage compared to VAEs. These are based on 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 and Ba, 2014) 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 (Larsen et al., 2016) and (Makhzani 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 resolution images similar to that produced using GANs. We trained and tested our network on three different datasets MNIST, LSUN and CelebA-HQ.

We summarize our main contributions as follows:

• An approach to address the mode collapse issue of GANs and blurred images generated using VAEs in a single network architecture.

• A theoretical analysis of our neural network backed by variational inference techniques which is trained using gradient descent.

• The details of our model architecture, hyperparameters, algorithm, latent variable size etc.

• Evaluation on MNIST, LSUN and CelebA-HQ dataset shows we outperform all previous state-of-the-art methods in terms of MMD, SSIM, log likelihood, reconstruction error, ELBO and KL divergence as the evaluation metrics.

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 in Equation 1 where the first term denotes the KL divergence between the original and the posterior distributions (Kingma and Welling, 2013). The second term denotes the reconstruction error of obtaining the sample from the latent distribution.

$$\log p_\theta(x) \geq -\text{KL}\left(q_\phi(z|x), p(z)\right) + \text{E}_{q_\phi(z|x)} \log p_\theta(x|z) \tag{1}$$

The right hand side in the equation above denotes 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 Equation 2:

$$\max_\theta \max_\phi \text{E}_{p_\mathcal{D}(x)} \left[-\text{KL}\left(q_\phi(z|x), p(z)\right) + \text{E}_{q_\phi(z|x)} \log p_\theta(x|z)\right] \tag{2}$$

The problem as reduced in the above equation can be further trained using gradient descent techniques for approximating the posterior distribution.

3.1 Variational Inference

Variational Inference is a set of classical methods to solve the heavy computational requirements of classical bayesian methods. It is used for approximating the distributions which uses optimization over the parameters to find the best approximation. According to bayes theorem, the computation of posterior requires three terms: a prior, a likelihood and an evidence. The first two terms are often part of the model itself while the third term requires computation as shown in Equation 3:

$$p(x) = \int_\theta p(x \mid \theta) p(\theta) d\theta \tag{3}$$

The challenge with the above equation is that it becomes intractable in higher dimensions. The computation of posterior hence becomes infeasible hence we require some approximation to compute it. This is where variational inference and Markov chain Monte Carlo (MCMC) methods come to rescue. These techniques are based on quantitatively measuring the distance between distributions using KL Divergence term. Considering p and q are two distributions, KL divergence is defined in Equation 4:

$$KL(p, q) = \mathbb{E}_{z \sim p}[\log p(z)] - \mathbb{E}_{z \sim p}[\log q(z)] \tag{4}$$

Since the posterior is intractable, hence we need an approximation to compute it. Using variational inference, the problem can be converted to an optimization problem as shown in Equation 5:

$$
\begin{aligned}
\omega^* &= \arg\min_{\omega \in \Omega} KL\left(f_\omega(z), p(z \mid x)\right) \\
&= \arg\min_{\omega \in \Omega} KL\left(f_\omega(z), p(x, z)\right) \\
&= \arg\max_{\omega \in \Omega} \left(-KL\left(f_\omega(z), p(x, z)\right)\right) \\
&= \arg\max_{\omega \in \Omega} \left(\mathbb{E}_{z \sim f_\omega}[\log p(z)] + \mathbb{E}_{z \sim f_\omega}[\log p(x \mid z)] - \mathbb{E}_{z \sim f_\omega}[\log f_\omega(z)]\right) \\
&= \arg\max_{\omega \in \Omega} \left(\mathbb{E}_{z \sim f_\omega}[\log p(x \mid z)] - KL\left(f_\omega, p(z)\right)\right)
\end{aligned}
\tag{5}
$$

The first term in the above equation is the expected log-likelihood while the second is the divergence between the approximation and the prior distributions.

4 Method

4.1 Dataset

In this work, we have used 3 publicly available datasets for training and evaluation:

1. 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.

2. LSUN dataset - This dataset contains millions of color images with 10 scene categories and 20 object categories. This is one of the most common datasets for training and testing GAN based neural networks.

3. CelebA-HQ dataset -This is a large-scale face attributes dataset with more than 200K celebrity images, each with 40 attribute annotations. This is also one of the most common datasets for training and testing GAN based neural networks.

4.2 Network 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 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 1. 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.
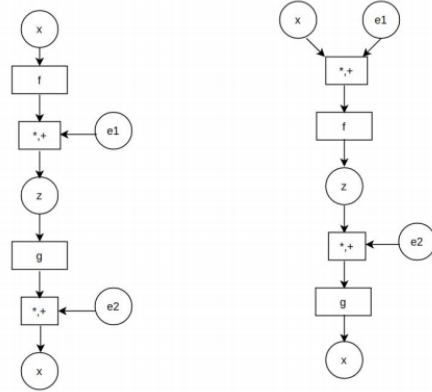

Figure 1: Comparison between standard VAE and our network where ϵ_1 and ϵ_2 denote samples from some noise distribution, x denotes image vector, z denotes latent space vector, f and g denotes encoder and decoder functions respectively and $+, *$ denotes addition and concat operators.

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. The model architecture is shown in Fig 2.
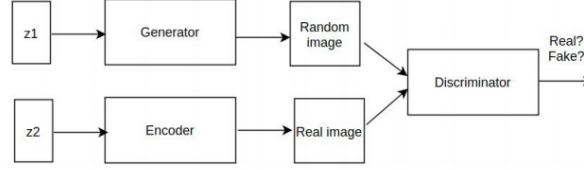

Figure 2: Our network architecture

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.

4.3 Algorithm

Next we present algorithm used in this work which is trained using Stochastic Gradient Descent (SGD):

Algorithm 1: Generative Variational Autoencoder (GVA)

---
i = 0
while *not converged* **do**
    Sample $\left\{x^{(1)}, \ldots, x^{(m)}\right\}$ from data distribution $p_{\mathcal{D}}(x)$
    Sample $\left\{z^{(1)}, \ldots, z^{(m)}\right\}$ from prior $p(z)$
    Sample $\left\{\epsilon^{(1)}, \ldots, \epsilon^{(m)}\right\}$ from $\mathcal{N}(0, 1)$
    $g_\theta \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\theta \log p_\theta \left(x^{(k)}|z_\phi \left(x^{(k)}, \epsilon^{(k)}\right)\right)$
    $g_\phi \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\phi \log p_\phi \left(x^{(k)}|z_\theta \left(x^{(k)}, \epsilon^{(k)}\right)\right)$
    Perform SGD-updates for θ, ϕ
    $i = i + 1$
end

The gradient descent used in the algorithm over the parameter θ is shown in Equation 6:

$$g_\theta \leftarrow \frac{1}{m} \sum_{k=1}^{m} \nabla_\theta \log p_\theta \left(x^{(k)}|z_\phi \left(x^{(k)}, \epsilon^{(k)}\right)\right) \tag{6}$$

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

4.4 Hyperparameters

The hyperparameters used in our network are specified in Table 3.

Table 1: Generator architecture details

Layer Output	Output Shape	Filter
Fully Connected	$256 \times 32 \times 32$	$512 \rightarrow 256 \times 32 \times 32$
ResNet Block	$256 \times 32 \times 32$	$512 \rightarrow 256 \rightarrow 256$
ResNet Block	$256 \times 32 \times 32$	$256 \rightarrow 256 \rightarrow 256$
Upsampling	$256 \times 64 \times 64$	-
ResNet Block	$128 \times 64 \times 64$	$256 \rightarrow 128 \rightarrow 128$
ResNet Block	$128 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 128$
Upsampling	$128 \times 128 \times 128$	-
ResNet Block	$64 \times 128 \times 128$	$128 \rightarrow 64 \rightarrow 64$
ResNet Block	$64 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 64$
Conv2D	$3 \times 128 \times 128$	$64 \rightarrow 3$

Table 2: Discriminator architecture details

Layer Output	Output Shape	Filter
Conv2D	$64 \times 128 \times 128$	$3 \rightarrow 64$
ResNet Block	$64 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 64$
ResNet Block	$128 \times 128 \times 128$	$64 \rightarrow 64 \rightarrow 128$
AvgPool2D	$128 \times 64 \times 64$	-
ResNet Block	$128 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 128$
ResNet Block	$256 \times 64 \times 64$	$128 \rightarrow 128 \rightarrow 256$
AvgPool2D	$256 \times 32 \times 32$	-
Fully Connected	$256 \times 32 \times 32$	$256 \times 32 \times 32 \rightarrow 1000$

Table 3: Hyperparameters details

Parameter	Value
Batch Size	16
Optimizer	Adam
Learning Rate	0.0005

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 used two evaluation metrics to measure the performance of our network. 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 4. compares MMD and MS-SSIM scores with previous architectures.

Table 4: Quantitative results on MNIST

Architecture	MMD $\times$ 0.0001	MS-SSIM
WGAN-GP (Gulrajani et al., 2017)	0.327	0.996
VAE-GAN (Larsen et al., 2016)	0.075	0.972
α-GAN (Lutz et al., 2018)	0.131	0.843
Ours	0.068	0.818

We also tried varying the latent variable size to see if any correlation is present and found that the latent variable is indeed very important in getting the best results. We noticed 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 5 compares the effect of different latent variable sizes on the MMD and MS-SSIM scores respectively.

Table 5: Effect of latent vector on MMD and SSIM on MNIST

Latent variable size	MMD $\times$ 0.0001	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

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.

Four common evaluation metrics have been used in the literature for testing the performance of generative models. These are log-likelihood, reconstruction error, ELBO and 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 6.

Table 6: Comparison of results in original VAE vs our architecture on MNIST

Evaluation Metrics	VAE (Kingma and Welling, 2013)	Ours
log-likelihood	-1.568	-1.353
reconstruction error	88.5×0.001	4.27×0.001
ELBO	-1.697	-1.404
KL divergence	0.165	0.046

We also compare our log probability distribution value with those obtained by previous state of the art which is shown in Table 7. The log probability distribution is an important evaluation metric in the sense that it shows the diversity of the samples generated.

Table 7: Results for independent samples for a model trained on MNIST

Method	$logp(x) \geq$
VAE + NF (T=80) (Rezende and Mohamed, 2015)	-85.1
VAE + HVI (T=16) (Salimans et al., 2015)	-88.3
conv VAE + HVI (T=16) (Salimans et al., 2015)	-84.1
VAE + VGP (2hl) (Tran et al., 2015)	-81.3
DRAW + VGP (Tran et al., 2015)	-79.9
VAE + IAF (Kingma et al., 2016)	-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. The images generated using the CELEBA-HQ dataset is shown in Fig 3.

Figure 3: 1024×1024 images generated using the CELEBA-HQ dataset.

The images generated using the LSUN BEDROOM dataset is shown in Fig 4.

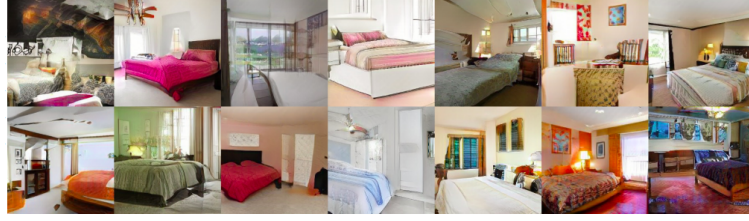Figure 4: 256×256 images generated using LSUN BEDROOM dataset

The images generated from different LSUN categories is shown in Fig 5.

Figure 5: Sample 256×256 images generated from different LSUN categories

Next we compare the generated images with previous state of the art networks on MNIST dataset as shown in Fig 6.
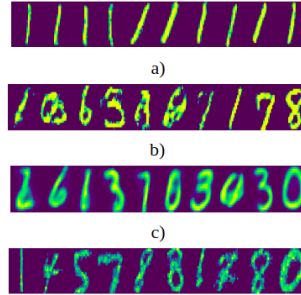

Figure 6: Generated MNIST images a) GAN b) WGAN c) VAE d) GVAE

7   Conclusions

In this paper, we presented a new training procedure for Variational Autoencoders based on generative models. 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 publicly available datasets and was able to generate images with much higher fidelity 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 resolution could be used for training more robust deep learning algorithms.

Acknowledgments

References

M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi. Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*, 2014.

X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*, 2016.

S. U. Dar, M. Yurt, L. Karacan, A. Erdem, E. Erdem, and T. Çukur. Image synthesis in multi-contrast mri with conditional generative adversarial networks. *IEEE transactions on medical imaging*, 38 (10):2375–2388, 2019.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. Improved training of wasserstein gans. In *Advances in neural information processing systems*, pages 5767–5777, 2017.

C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama. Gan-based synthetic brain mr image generation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 734–738. IEEE, 2018.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

F. Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

A. Kucukelbir, D. Tran, R. Ranganath, A. Gelman, and D. M. Blei. Automatic differentiation variational inference. *The Journal of Machine Learning Research*, 18(1):430–474, 2017.

A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566, 2016.

S. Lutz, K. Amplianitis, and A. Smolic. Alphagan: Generative adversarial networks for natural image matting. *arXiv preprint arXiv:1807.10088*, 2018.

L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

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*, pages 4467–4477, 2017.

S. Nowozin, B. Cseke, and R. Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in neural information processing systems*, pages 271–279, 2016.

A. Odena, C. Olah, and J. Shlens. Conditional image synthesis with auxiliary classifier gans. In *International conference on machine learning*, pages 2642–2651, 2017.

A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *arXiv preprint arXiv:1601.06759*, 2016.

B. Poole, A. A. Alemi, J. Sohl-Dickstein, and A. Angelova. Improved generator objectives for gans. *arXiv preprint arXiv:1612.02780*, 2016.

A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

R. Ranganath, D. Tran, J. Altosaar, and D. Blei. Operator variational inference. In *Advances in Neural Information Processing Systems*, pages 496–504, 2016.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*, 2015.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

M. Rosca, B. Lakshminarayanan, D. Warde-Farley, and S. Mohamed. Variational approaches for auto-encoding generative adversarial networks. *arXiv preprint arXiv:1706.04987*, 2017.

H. R. Roth, L. Lu, A. Farag, H.-C. Shin, J. Liu, E. B. Turkbey, and R. M. Summers. Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation. In *International conference on medical image computing and computer-assisted intervention*, pages 556–564. Springer, 2015.

T. Salimans, D. Kingma, and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International Conference on Machine Learning*, pages 1218–1226, 2015.

H.-C. Shin, N. A. Tenenholtz, J. K. Rogers, C. G. Schwarz, M. L. Senjem, J. L. Gunter, K. P. Andriole, and M. Michalski. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. In *International workshop on simulation and synthesis in medical imaging*, pages 1–11. Springer, 2018.

C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

D. Tran, R. Ranganath, and D. M. Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

A. Van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in neural information processing systems*, pages 4790–4798, 2016.

Y. Wu, Y. Burda, R. Salakhutdinov, and R. Grosse. On the quantitative analysis of decoder-based generative models. *arXiv preprint arXiv:1611.04273*, 2016.

B. Yu, L. Zhou, L. Wang, J. Fripp, and P. Bourgeat. 3d cgan based cross-modality mr image synthesis for brain tumor segmentation. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 626–630. IEEE, 2018.