IFT6269 Project Report on Adversarially Learned Inference

Aditya Joshi ADITYA.JOSHI@UMONTREAL.CA
Université de Montréal

Barleen Kaur BARLEEN.KAUR@MAIL.MCGILL.CA
McGill University

Timothy Nest TIMOTHY.NEST@UMONTREAL.CA
Université de Montréal

Deepak Sharma DEEPAK.SHARMA@MAIL.MCGILL.CA
McGill University

Abstract

The purpose of this document is to detail our experiments to reproduce the results of the paper that introduced the adversarially learned inference (Dumoulin et al., 2016) (ALI) model. This model trains an inference network along with a generative network under the adversarial setting, similar to generative adversarial networks (Goodfellow et al., 2014) (GAN).

1. Introduction

Generative models can be used for compression, denoising, inpainting, texture synthesis, semi-supervised learning, and unsupervised feature learning etc. Therefore, it is not surprising that there a variety of techniques available for their creation from high dimensionsal datasets. Recently, three techniques have emerged as effective for learning deep generative models: the Variational Autoencoder (Kingma & Welling, 2013) (VAE) that learns an inference mechanism, the Generative Adversarial Network (GAN), that learns latent representations without any means for inference, and autoregressive approaches, that model input variables directly, without the need for a latent representation in between. Theoretically, given infinite compute and memory, these approaches should be qualitatively equivalent. Practically, these models have different strengths and weaknesses. This project seeks to demonstrate the strengths of a new approach, namely the Adversarial Learned Inference (ALI) model, that combines the strengths of a GAN and a VAE.

VAEs perform variational inference to maximize the log-likelihood of the data, however their image samples are often blurry. This has been well understood as a conseqeuence of maximizing the log likelihood of the data under the factorization assumption of the output pixels given the latents. The learned likelihood distribution is diffused and results in noisy, blurry samples for complex datasets such as Imagenet and CIFAR. GANs produce much sharper images but lack an inference mechanism, whereas autoregressive models produce outstanding samples, but at the cost of slow sampling speed and without an abstract lower dimensional representation of the data.

The ALI model attempts to bridge the gap between VAEs and GANs. It simultaneously learns an inference network (encoder) and a deep generative network (decoder) by training a discriminator to discriminate between joint samples produced by both. The encoder produces a joint sample of the original image and its encoded representation, whereas the decoder produces joint samples of a sample image with the latent representation used to deocde it. As in a GAN, adding the discriminator sets up the optimization process as a minimax game with the goal to make the two joint distributions indistinguishable from each other. This implies that the ALI model has learned an inference and generative mechanism.

2. Generative Adversarial Networks

The Generative adversarial network consists of a generator network and a discriminator network which learn through a minimax two player game as shown in Figure 1. The generator takes in a random source of noise as input and tries to generate images that resemble the input data distribution closely in order to fool the discriminator into classifying a

fake image as a real image. On the other hand, the discriminator is getting trained to classify a given image as real or fake correctly.

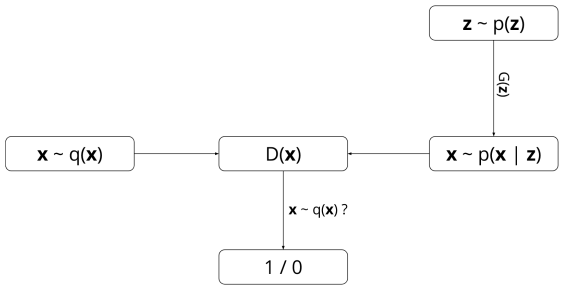


Figure 1. Generative Adversarial Network

2.1. What's missing in GANs?

- Inference answers the questions about the latent variables z given an input x.
- Could be useful to discover important properties of the data distribution $q(x)$.

3. Adversarially Learned Inference

The adversarially learned inference (ALI) model is a deep directed generative model which jointly learns an inference network (encoder) and a generation network (decoder) in a GAN-like adversarial process as shown in Figure 2. While a discriminator network is trained to distinguish between the joint samples of data and latent variable produced by the encoder and decoder, on the other hand, the encoder and decoder network are getting trained to fool the discriminator.

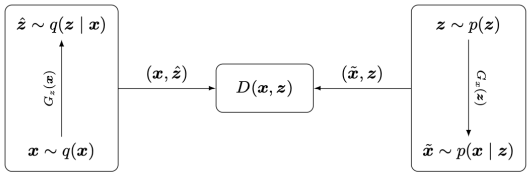


Figure 2. The Adversarial Learned Inference game

To better understand the ALI framework, let's consider the joint probability distribution of $\boldsymbol{x}$ and $\boldsymbol{z}$ in the encoder and decoder be as follows:

- *Encoder* joint distribution: $q(\boldsymbol{x}, \boldsymbol{z}) = q(\boldsymbol{x}) \cdot q(\boldsymbol{z} \mid \boldsymbol{x})$
- *Decoder* joint distribution: $p(\boldsymbol{x}, \boldsymbol{z}) = p(\boldsymbol{z}) \cdot q(\boldsymbol{x} \mid \boldsymbol{z})$

Here, the encoder marginal, $\boldsymbol{q(x)}$ is equal to the true data distribution which is known to us and decoder marginal is taken as $\boldsymbol{p(z) = \mathcal{N}(0, I)}$.

ALI tries to equalize the encoder and decoder joint distribution ensuring that all the marginals and all conditionals match too, i.e. $\boldsymbol{q(x) = p(x)}$ and $\boldsymbol{q(z \mid x) = p(z \mid x)}$. Joint pairs $\boldsymbol{(x, z)}$ are drawn either from $\boldsymbol{q(x, z)}$ or $\boldsymbol{p(x, z)}$, and the discriminator network learns to distinguish between the two, while the encoder and decoder networks are trained to fool the discriminator using the objective functions stated in equation (1). The generator minimizes the Jenson Shannon divergence (Lin, 1991) between the two joint distributions under the assumption of an optimal discriminator (Goodfellow et al., 2014). The algorithm for ALI framework is shown in Figure (3)

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{q(x)} [\log(D(\boldsymbol{x}, G_z(\boldsymbol{x})))] + \mathop{\mathbb{E}}_{p(z)} [\log(1 - D(G_x(\boldsymbol{z}), \boldsymbol{z}))] \quad (1)$$

Algorithm 1 The ALI training procedure.

$\theta_g, \theta_d \leftarrow$ initialize network parameters
repeat
  $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)} \sim q(\boldsymbol{x})$ ▷ Draw M samples from the dataset and the prior
  $\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(M)} \sim p(\boldsymbol{z})$
  $\hat{\boldsymbol{z}}^{(i)} \sim q(\boldsymbol{z} \mid \boldsymbol{x} = \boldsymbol{x}^{(i)}), \quad i = 1, \ldots, M$ ▷ Sample from the conditionals
  $\tilde{\boldsymbol{x}}^{(j)} \sim p(\boldsymbol{x} \mid \boldsymbol{z} = \boldsymbol{z}^{(j)}), \quad j = 1, \ldots, M$
  $\rho_q^{(i)} \leftarrow D(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{z}}^{(i)}), \quad i = 1, \ldots, M$ ▷ Compute discriminator predictions
  $\rho_p^{(j)} \leftarrow D(\tilde{\boldsymbol{x}}^{(j)}, \boldsymbol{z}^{(j)}), \quad j = 1, \ldots, M$
  $\mathcal{L}_d \leftarrow -\frac{1}{M}\sum_{i=1}^{M} \log(\rho_q^{(i)}) - \frac{1}{M}\sum_{j=1}^{M} log(1 - \rho_p^{(j)})$ ▷ Compute discriminator loss
  $\mathcal{L}_g \leftarrow -\frac{1}{M}\sum_{i=1}^{M} \log(1 - \rho_q^{(i)}) - \frac{1}{M}\sum_{j=1}^{M} \log(\rho_p^{(j)})$ ▷ Compute generator loss
  $\theta_d \leftarrow \theta_d - \nabla_{\theta_d}\mathcal{L}_d$ ▷ Gradient update on discriminator network
  $\theta_g \leftarrow \theta_g - \nabla_{\theta_g}\mathcal{L}_g$ ▷ Gradient update on generator networks
until convergence

Figure 3. ALI Training

3.1. GANs Vs ALI

The ALI architecture is similar to that of GAN's, with the differences as stated in Figure 9

Factor	GANs	ALI
Generator	Only one component: • $G_x(\boldsymbol{z})$: maps $\boldsymbol{z}$ to $\boldsymbol{x}$	Two components: • Encoder $G_z(\boldsymbol{x})$: maps $\boldsymbol{x}$ to $\boldsymbol{z}$ • Decoder $G_x(\boldsymbol{z})$: maps $\boldsymbol{z}$ to $\boldsymbol{x}$
Discriminator	Classifies $x \sim q(x)$ and $\tilde{x} \sim p(x)$	Classifies joint sample $q(x, \tilde{z})$ and $p(\tilde{x}, z)$

Figure 4. GANs Vs ALI

A side effect of using an adversarial technique is that we do not need to compute the conditional density $\boldsymbol{q(z \mid x)}$ but only sample from it. This is done using the reparametrization trick (Kingma et al., 2016). The latent representation z is sampled according to:

$$z = \mu(x) + \sigma(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

4. Related Work

A variety of approaches to generative modelling have sprouted in recent years. Models such as the InfoGAN (Chen et al., 2016) learn a feedforward inference network on the posterior of a subset c of the latent representation z. However the approximate posterior is incomplete as it does not fully learn z. The adversarial autoencoder model (Makhzani et al., 2015) replaces the KL-divergence term with a discriminator that is trained to distinguish between approximate posterior and prior samples, which provides a more flexible approach to matching the marginal $q(z)$ and the prior. The ALI approach is similar, since it implicitly minimizes the divergence between the prior $p(z)$ and the posterior $q(z \mid x)$. However, it does so under the adversarial setting with no explicit reconstruction loss and the discriminator receives joint pairs of samples (x, z) rather than marginal z samples. (Larsen et al., 2015) replace the VAE reconstruction loss with GANs and jointly train the decoder of a VAE with a GAN generator. This generative model simultaneously learns to encode and generate dataset samples. Independent work by (Donahue et al., 2016) proposes essentially the same model to ALI under the name Bidirectional GAN (BiGAN).

5. Experiments

We performed the following experiments on the CelebA, MNIST and SVHN dataset.

- **Generation:** Sampling from $p(\tilde{x})$. This is similar to the image generations in a traditional GAN framework.
- **Reconstruction:** Using the learned mapping from the input to the latent space, we can pass an input x through the trained encoder $G_z(x)$ and then decode it with the trained decoder $G_x(z)$, effectively carrying out the computation $G_x(G_z(x))$.
- **Interpolation:** We sample two images from the held out test set, x_1 and x_2, and compute their respective latent representations, z_1 and z_2 using the trained encoder. We then linearly interpolate points between z_1 and z_2 to generate intermediatary points in the z space, which are then mapped to the x space using the decoder.

5.1. Hyperparameters

We used the *Adam* optimizer for all the experiments. For all the datasets, we rescaled the images to 28 x 28 and trained the network using batch size of 100. The hyperparameters that we used are mentioned in Table 1. The network architecture for ALI is the same as mentioned in the reference paper (Dumoulin et al., 2016) and is provided in the appendix section.

Dataset	**Epochs**	z **dimension**	G **learning rate**	D **learning rate**
CelebA	100	256	2×10^{-4}	1×10^{-4}
MNIST	100	128	2×10^{-4}	1×10^{-5}
SVHN	100	256	2×10^{-4}	1×10^{-4}

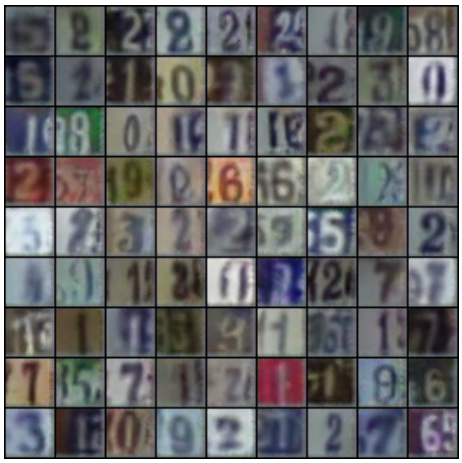
Table 1. Hyperparameters used for all experiments

6. Results and Discussion

6.1. SVHN

- **Generation:**

(a) Original



(b) Generated

Figure 5. Original images from the SVHN dataset are compared with generated images from the trained ALI model

- **Reconstruction:**

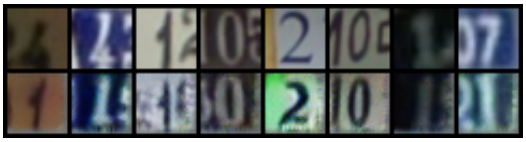


Figure 6. SVHN dataset: Original images in the top row are compared with their reconstructed counterparts in the bottom row.

- **Interpolation:**

Figure 7. SVHN dataset: Latent space interpolations in the x space. The left and right columns correspond to the original pairs x_1 and x_2 and the columns in between correspond to the decoding of latent representations interpolated linearly from z_1 to z_2

6.2. MNIST

- **Generation:**

(a) Original

(b) Generated

Figure 8. Original images from the MNIST dataset are compared with generated images from the trained ALI model

- **Reconstruction:**

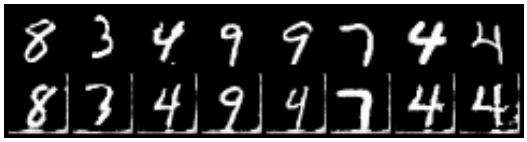
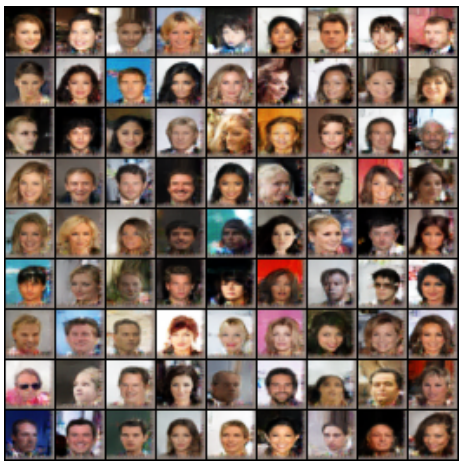


Figure 9. MNIST dataset: Original images in the top row are compared with their reconstructed counterparts in the bottom row

6.3. CelebA

- **Generation:**

(a) Original

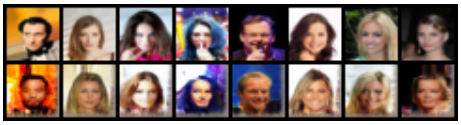


(b) Generated

Figure 10. Original images from the CelebA dataset are compared with generated images from the trained ALI model.

- **Reconstruction:**

(a) Reconstruction 1



(b) Reconstruction 2

Figure 11. CelebA dataset: Original images in the top row are compared with their reconstructed counterparts in the bottom row

- **Interpolation:**

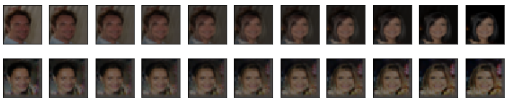


Figure 12. CelebA dataset: Latent space interpolations in the x space. The left and right columns correspond to the original pairs x_1 and x_2 and the columns in between correspond to the decoding of latent representations interpolated linearly from z_1 to z_2

6.4. Training Instability

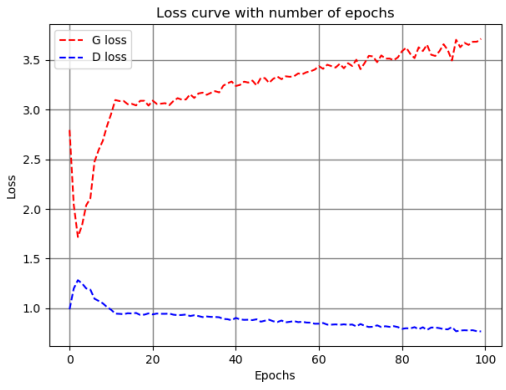


Figure 13. SVHN dataset: Generator and Discriminator losses tend to hit optimal and diverge as seen here.

The training of the ALI model was unstable (the model fails to converge). We used some *hacks*[1] to make training more efficient, like adding artificial noise to the input of the discriminator. This technique is inspired by work carried out in (Arjovsky & Bottou, 2017). The loss curve as shown in Figure 13 is a little peculiar, the generator and discriminator losses are diverging even though the quality of the images produced by the generator is improving. One possible reason is that the discriminator might be overfitting over the training dataset, effectively limiting the generator's capacity to fool the discriminator.

We also observed that the reconstruction quality (measured by MSE loss between the reconstructed images and their true counterparts) decreased for more complex datasets like CIFAR-10.

To assess the question of instability, we also implemented an alternate training objective, using the 'Stochastic Extragradient Method' [SEM] outlined by (Gidel et al., 2018)

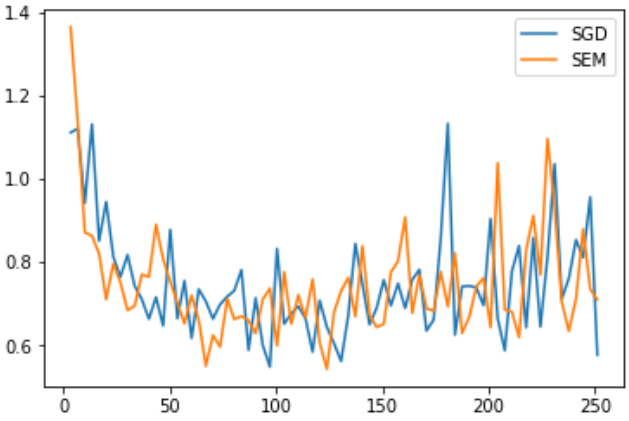


Figure 14. CIFAR-10 dataset: Comparisons of MSE loss on reconstructed images vs. CPU time (in minutes) with SGD vs SEM-based training with resampling. Importantly this comparison uses 5e-4 for generator step-size, and 1e-4 for discriminator, and thus fails to leverage known advantages of SEM with increased learning rates.

7. Conclusion

We qualitatively assessed the results of the ALI model by performing the *generation*, *reconstruction* and *interpolation* tasks on several datasets. We observed that the generated images are of comparable quality to those of GAN's. The interpolation task is useful to check if the model is overfitting the training data. Since the transitions are smooth, we can guess that the model has not simply memorized the training dataset (i.e., concentrated the probability mass near the training examples), but has learnt an efficient representation of the latent space.

8. Future Work

In future work we would like to extend the ALI paradigm to explore hierarchical variants which have been shown to

[1] GAN hacks

substantially mitigate reconstruction loss (Belghazi et al., 2018). We believe the stability issues observed here could also be better managed with further experiments on hyper-parameters.

We also plan to use this model to augment medical imaging datasets with fake medical images with certain characteristics, inferred from medical images that are more scarce. Moreover, we plan to compare this approach with semi-supervised learning as well as incorporate active learning approaches to the value function to increase the model training efficiency from a pool of real and fake medical imaging data.

References

Arjovsky, Martin and Bottou, Léon. Towards principled methods for training generative adversarial networks. *arXiv preprint arXiv:1701.04862*, 2017.

Belghazi, Mohamed Ishmael, Rajeswar, Sai, Mastropietro, Olivier, Rostamzadeh, Negar, Mitrovic, Jovana, and Courville, Aaron. Hierarchical adversarially learned inference. *arXiv preprint arXiv:1802.01071*, 2018.

Chen, Xi, Duan, Yan, Houthooft, Rein, Schulman, John, Sutskever, Ilya, and Abbeel, Pieter. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pp. 2172–2180, 2016.

Donahue, Jeff, Krähenbühl, Philipp, and Darrell, Trevor. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

Dumoulin, Vincent, Belghazi, Ishmael, Poole, Ben, Mastropietro, Olivier, Lamb, Alex, Arjovsky, Martin, and Courville, Aaron. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

Gidel, Gauthier, Berard, Hugo, Vincent, Pascal, and Lacoste-Julien, Simon. A variational inequality perspective on generative adversarial nets. *CoRR*, abs/1802.10551, 2018. URL http://arxiv.org/abs/1802.10551.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

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

Kingma, Durk P, Salimans, Tim, Jozefowicz, Rafal, Chen, Xi, Sutskever, Ilya, and Welling, Max. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pp. 4743–4751, 2016.

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

Lin, Jianhua. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37 (1):145–151, 1991.

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

9. Appendix

Operation	Kernel	Strides	Feature maps	BN?	Dropout	Nonlinearity
$G_z(x)$ – $3 \times 32 \times 32$ input						
Convolution	5×5	1×1	32	$\surd$	0.0	Leaky ReLU
Convolution	4×4	2×2	64	$\surd$	0.0	Leaky ReLU
Convolution	4×4	1×1	128	$\surd$	0.0	Leaky ReLU
Convolution	4×4	2×2	256	$\surd$	0.0	Leaky ReLU
Convolution	4×4	1×1	512	$\surd$	0.0	Leaky ReLU
Convolution	1×1	1×1	512	$\surd$	0.0	Leaky ReLU
Convolution	1×1	1×1	512	$\times$	0.0	Linear
$G_x(z)$ – $256 \times 1 \times 1$ input						
Transposed convolution	4×4	1×1	256	$\surd$	0.0	Leaky ReLU
Transposed convolution	4×4	2×2	128	$\surd$	0.0	Leaky ReLU
Transposed convolution	4×4	1×1	64	$\surd$	0.0	Leaky ReLU
Transposed convolution	4×4	2×2	32	$\surd$	0.0	Leaky ReLU
Transposed convolution	5×5	1×1	32	$\surd$	0.0	Leaky ReLU
Convolution	1×1	1×1	32	$\surd$	0.0	Leaky ReLU
Convolution	1×1	1×1	3	$\times$	0.0	Sigmoid
$D(x)$ – $3 \times 32 \times 32$ input						
Convolution	5×5	1×1	32	$\times$	0.2	Leaky ReLU
Convolution	4×4	2×2	64	$\surd$	0.2	Leaky ReLU
Convolution	4×4	1×1	128	$\surd$	0.2	Leaky ReLU
Convolution	4×4	2×2	256	$\surd$	0.2	Leaky ReLU
Convolution	4×4	1×1	512	$\surd$	0.2	Leaky ReLU
$D(z)$ – $256 \times 1 \times 1$ input						
Convolution	1×1	1×1	512	$\times$	0.2	Leaky ReLU
Convolution	1×1	1×1	512	$\times$	0.2	Leaky ReLU
$D(x, z)$ – $1024 \times 1 \times 1$ input						
Concatenate $D(x)$ and $D(z)$ along the channel axis						
Convolution	1×1	1×1	1024	$\times$	0.2	Leaky ReLU
Convolution	1×1	1×1	1024	$\times$	0.2	Leaky ReLU
Convolution	1×1	1×1	1	$\times$	0.2	Sigmoid

Figure 15. ALI Architecture (taken from the reference paper)

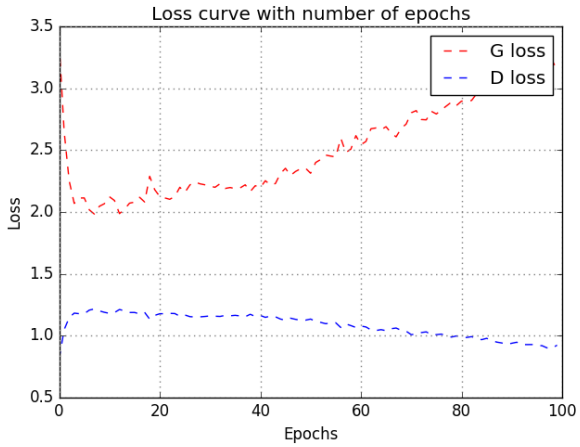


Figure 16. Generator and Discriminator loss with epochs on CIFAR10 dataset

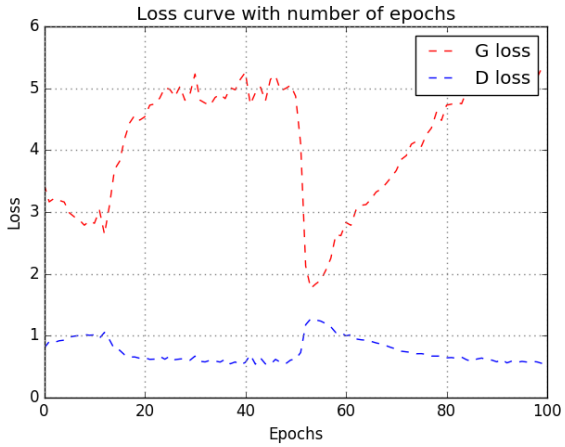


Figure 17. Generator and Discriminator loss with epochs on MNIST dataset