
Adversarially Learned Inference

Vincent Dumoulin*

vincent.dumoulin@umontreal.ca

Ishmael Belghazi*

ishmael.belghazi@gmail.com

Ben Poole [†]

poole@cs.stanford.edu

Alex Lamb*

alex6200@gmail.com

Martin Arjovsky*

martinarjovsky@gmail.com

Olivier Mastropietro*

oli.mastro@gmail.com

Aaron Courville* CIFAR Fellow

aaron.courville@gmail.com

Abstract

We introduce the adversarially learned inference (ALI) model, which jointly learns a generation network and an inference network using an adversarial process. The generation network maps samples from stochastic latent variables to the data space while the inference network maps training examples in data space to the space of latent variables. An adversarial game is cast between these two networks and a discriminative network that is trained to distinguish between joint latent/data-space samples from the generative network and joint samples from the inference network. We illustrate the ability of the model to learn mutually coherent inference and generation networks through the inspections of model samples and reconstructions and confirm the usefulness of the learned representations by obtaining a performance competitive with other recent approaches on the semi-supervised SVHN task.

1 Introduction

Deep directed generative models have emerged as a powerful technique for modeling complex high-dimensional datasets. These models permit fast ancestral sampling, but are often challenging to learn due to the complexities of inference. Recently, two classes of algorithms have emerged as effective for learning deep directed generative models: 1) Techniques based on the Variational Autoencoder (VAE) that aim to improve the quality and efficiency of inference by learning an inference machine (Kingma and Welling, 2013; Rezende *et al.*, 2014), and 2) techniques based on Generative Adversarial Networks (GANs) that bypass inference altogether (Goodfellow *et al.*, 2014). While both techniques are provably consistent given infinite capacity and data, they learn very different generative models on typical datasets.

VAE-based techniques aim to maximize the log-likelihood of the observed data \mathbf{x} by introducing a set of stochastic latent variables \mathbf{z} and marginalizing them out of the joint distribution $p(\mathbf{x}, \mathbf{z})$. While exact marginalization of the latent variables is generally intractable, the VAE introduces an approximate posterior $q(\mathbf{z} \mid \mathbf{x})$ and maximizes a variational lower bound on the log-likelihood of $p(\mathbf{x})$. Several VAE variants have been proposed. One approach improves the quality of the approximate posterior using hybrid Monte Carlo (Salimans *et al.*, 2014). Another approach, known as the DRAW model (Gregor *et al.*, 2015), subdivides the inference and generation processes into multiple sequential steps through the use of recurrent neural networks and an attention mechanism.

*MILA, Université de Montréal

[†]Neural Dynamics and Computation Lab, Stanford

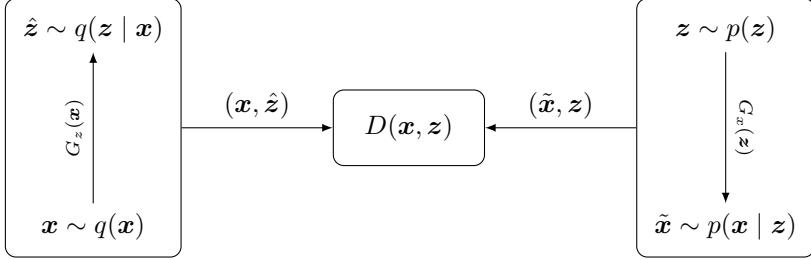


Figure 1: The adversarially learned inference (ALI) game.

Despite the impressive progress of VAE-based approaches for learning deep directed generative models, they still suffer from a well-recognized issue of the maximum likelihood training paradigm. Models trained to maximize likelihood of the training data tend to be conservative, distributing probability mass diffusely over the data space (Theis *et al.*, 2015). In the case of learning generative models of images, this results in almost all probability mass lying outside the relatively restrictive subset of pixel space occupied by natural images. The direct consequence of this is that image samples from VAE-trained models tend to be blurry (Goodfellow *et al.*, 2014; Larsen *et al.*, 2015).

On the other hand, GAN-based techniques are trained via an adversarial process that does not appear to suffer from the same probability mass diffusion problem as does maximum likelihood. In the GAN paradigm, a discriminative network is trained to distinguish between the empirical distribution and samples produced by a generative network. Meanwhile, the generative network is trained to produce samples intended to fool the discriminator into classifying them as true samples from the training set. While the adversarial setting presents a challenging (and often unstable) training paradigm, it results in a generative network capable of producing samples that exceed those of the best VAE techniques (Radford *et al.*, 2015; Larsen *et al.*, 2015). Recent GAN-based convolutional generative networks have been able to achieve dramatic improvements in the quality of synthetic images (Radford *et al.*, 2015; Denton *et al.*, 2015).

While GANs learn a generative model that produces higher-quality samples, only the VAE-based models learn an efficient mechanism for inference. For applications such as semi-supervised learning, GANs are insufficient as they do not provide an efficient inference mechanism. Recently, efforts have aimed to bridge the gap between VAEs and GANs, to learn generative models with higher-quality samples while learning an efficient inference network (Larsen *et al.*, 2015; Lamb *et al.*, 2016; Dosovitskiy and Brox, 2016). While this is certainly a promising research direction, VAE-GAN hybrids tend to manifest a compromise of the strengths and weaknesses of both approaches.

In this paper, we propose a novel approach to integrate efficient inference with the GAN framework. Our approach, called Adversarially Learned Inference (ALI), casts the learning of both an inference machine (or encoder) and a deep directed generative model (or decoder) in an GAN-like adversarial framework. A discriminator is trained to discriminate joint samples of the data and the corresponding latent variable from the encoder (or approximate posterior) from joint samples from the decoder. In opposition to the discriminator, we have two generative models, the encoder and the decoder, trained together to fool the discriminator.

GAN is an example of how one can leverage highly effective discriminative training techniques in service of learning deep generative networks. Here, we are effectively doubling down on the gambit that we can exploit discriminative training. Not only are we asking the discriminator to distinguish synthetic samples from real data, but we are requiring it to distinguish between two joint distributions over the data space and the latent variables.

With experiments on a toy task, the Street View House Numbers (SVHN) dataset (Netzer *et al.*, 2011), the CIFAR-10 object recognition dataset (Krizhevsky and Hinton, 2009), the CelebA face dataset (Liu *et al.*, 2015) and a downsampled version of the ImageNet dataset (Russakovsky *et al.*, 2015), we show qualitatively that we maintain the high sample fidelity associated with the GAN framework, while gaining the ability to perform efficient inference.

2 Adversarially learned inference

Consider the two following probability distributions over \mathbf{x} and \mathbf{z} :

- the *encoder* joint distribution $q(\mathbf{x}, \mathbf{z}) = q(\mathbf{x})q(\mathbf{z} | \mathbf{x})$,
- the *decoder* joint distribution $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x} | \mathbf{z})$.

These two distributions have marginals that are known to us: the encoder marginal $q(\mathbf{x})$ is the empirical data distribution and the decoder marginal $p(\mathbf{z})$ is usually defined to be a simple, factorized distribution, such as the standard Normal distribution $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$. As such, the generative process between $q(\mathbf{x}, \mathbf{z})$ and $p(\mathbf{x}, \mathbf{z})$ is reversed.

ALI's objective is to match the two joint distributions. If this is achieved, then we are ensured that all marginals match and all conditional distributions also match. In particular, we are assured that the conditional $q(\mathbf{z} | \mathbf{x})$ matches the posterior $q(\mathbf{z} | \mathbf{x})$.

In order to match the joint distributions, an adversarial game is played. Joint pairs (\mathbf{x}, \mathbf{z}) are drawn either from $q(\mathbf{x}, \mathbf{z})$ or $p(\mathbf{x}, \mathbf{z})$, and a discriminator network learns to discriminate between the two, while the encoder and decoder networks are trained to fool the discriminator.

The value function describing the game is given by:

$$\begin{aligned} \min_G \max_D V(D, G) &= \mathbb{E}_{q(\mathbf{x})}[\log(D(\mathbf{x}, G_z(\mathbf{x})))] + \mathbb{E}_{p(\mathbf{z})}[\log(1 - D(G_x(\mathbf{z}), \mathbf{z}))] \\ &= \iint q(\mathbf{x})q(\mathbf{z} | \mathbf{x}) \log(D(\mathbf{x}, \mathbf{z})) d\mathbf{x} d\mathbf{z} \\ &\quad + \iint p(\mathbf{z})p(\mathbf{x} | \mathbf{z}) \log(1 - D(\mathbf{x}, \mathbf{z})) d\mathbf{x} d\mathbf{z}. \end{aligned} \tag{1}$$

Sampling from $q(\mathbf{z} | \mathbf{x})$ amounts to drawing an \mathbf{x} from the data distribution and propagating the sample through the encoder network $(\mu_z, \sigma_z) = G_z(\mathbf{x})$, then generating $\hat{\mathbf{z}} \sim \mathcal{N}(\mu_z, \sigma_z)$. Sampling from $p(\mathbf{x} | \mathbf{z})$ amounts to sampling from $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and propagating the sample through the decoder network to recover $\tilde{\mathbf{x}}$.

The discriminator is trained to distinguish between samples from the encoder $(\mathbf{x}, \hat{\mathbf{z}}) \sim q(\mathbf{x}, \mathbf{z})$ and samples from the decoder $(\tilde{\mathbf{x}}, \mathbf{z}) \sim p(\mathbf{x}, \mathbf{z})$. The generator is trained to fool the discriminator, i.e., to generate \mathbf{x}, \mathbf{z} pairs from $q(\mathbf{x}, \mathbf{z})$ or $p(\mathbf{x}, \mathbf{z})$ that are indistinguishable one from another. See Figure 1 for a diagram of the adversarial game and Algorithm 1 for an algorithmic description of the procedure.

In such a setting, and under the assumption of an optimal discriminator, the generator minimizes the Jensen-Shannon divergence (Lin, 1991) between $q(\mathbf{x}, \mathbf{z})$ and $p(\mathbf{x}, \mathbf{z})$. This can be shown using the same proof sketch as in the original GAN paper (Goodfellow *et al.*, 2014).

2.1 Relation to GAN

ALI bears close resemblance to GAN, but it differs from it in the two following ways:

- The generator has two components: the encoder, $G_z(\mathbf{x})$, which maps data samples \mathbf{x} to \mathbf{z} -space, and the decoder $G_x(\mathbf{z})$, which maps samples from the prior $p(\mathbf{z})$ (a source of noise) to the input space.
- The discriminator is trained to distinguish between joint pairs $(\mathbf{x}, \hat{\mathbf{z}} = G_x(\mathbf{x}))$ and $(\tilde{\mathbf{x}} = G_x(\mathbf{z}), \mathbf{z})$, as opposed to marginal samples $\mathbf{x} \sim q(\mathbf{x})$ and $\tilde{\mathbf{x}} \sim p(\mathbf{x})$.

2.2 Reparametrization trick

In order for the gradient to propagate from the discriminator network to the encoder and decoder networks, the gradient needs to flow through the sampling processes $\hat{\mathbf{z}} \sim q(\mathbf{z} | \mathbf{x})$ and $\tilde{\mathbf{x}} \sim p(\mathbf{x} | \mathbf{z})$.

To achieve this, the reparametrization trick (Kingma, 2013; Bengio *et al.*, 2013a,b) is employed. Instead of sampling directly from the desired distribution, the random variable is computed as a deterministic transformation of some noise such that its distribution is the desired distribution.

Algorithm 1 The ALI training procedure.

```

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

```

For instance, if $q(z \mid x) = \mathcal{N}(\mu(x), \sigma^2(x)I)$, one can draw samples by computing

$$z = \mu(x) + \sigma(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \quad (2)$$

2.3 Generator value function

As with GANs, when ALI's discriminator gets too far ahead, its generator may have a hard time minimizing the value function in Equation 1. If the discriminator's output is sigmoidal, then the gradient of the value function with respect to the discriminator's output vanishes to zero as the output saturates.

As a workaround, the generator is trained to maximize

$$V'(D, G) = \mathbb{E}_{q(\mathbf{x})}[\log(1 - D(\mathbf{x}, G_{\mathbf{z}}(\mathbf{x})))] + \mathbb{E}_{p(\mathbf{z})}[\log(D(G_{\mathbf{x}}(\mathbf{z}), \mathbf{z}))] \quad (3)$$

which has the same fixed points but whose gradient is stronger when the discriminator's output saturates.

The adversarial game does not require an analytical expression for the joint distributions. This means we can introduce variable changes without having to know the explicit distribution over the new variable. For instance, sampling from $p(z)$ could be done by sampling $\epsilon \sim \mathcal{N}(0, I)$ and passing it through an arbitrary differentiable function $z = f(\epsilon)$.

However, gradient propagation into the encoder and decoder networks relies on the reparametrization trick, which means that ALI is not directly applicable to either applications with discrete data or to models with discrete latent variables.

2.4 Discriminator optimality

Proposition 1. *Given a fixed generator G , the optimal discriminator is given by*

$$D^*(x, z) = \frac{q(x, z)}{q(x, z) + p(x, z)}. \quad (4)$$

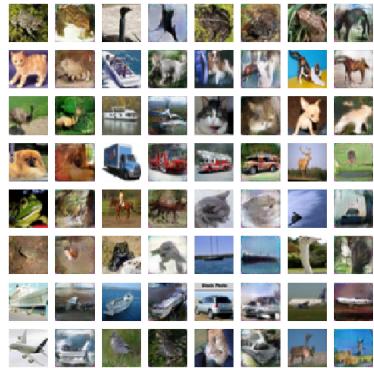
Proof. For a fixed generator G , the complete data value function is

$$V(D, G) = \mathbb{E}_{x, z \sim q(x, z)}[\log(D(x, z))] + \mathbb{E}_{x, z \sim p(x, z)}[\log(1 - D(x, z))]. \quad (5)$$

The result follows by the concavity of the log and the simplified Euler-Lagrange equation first order conditions on $(x, z) \rightarrow D(x, z)$. \square



(a) CIFAR10 samples.



(b) CIFAR10 reconstructions.

Figure 2: Samples and reconstructions on the CIFAR10 dataset. For the reconstructions, odd columns are original samples from the validation set and even columns are corresponding reconstructions (e.g., second column contains reconstructions of the first column’s validation set samples).



(a) SVHN samples.



(b) SVHN reconstructions.

Figure 3: Samples and reconstructions on the SVHN dataset. For the reconstructions, odd columns are original samples from the validation set and even columns are corresponding reconstructions.

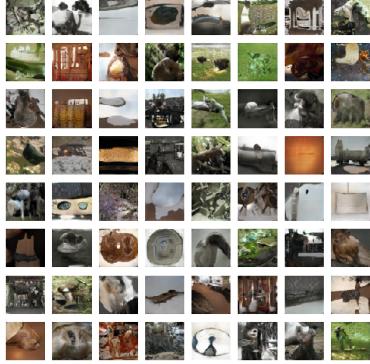


(a) CelebA samples.

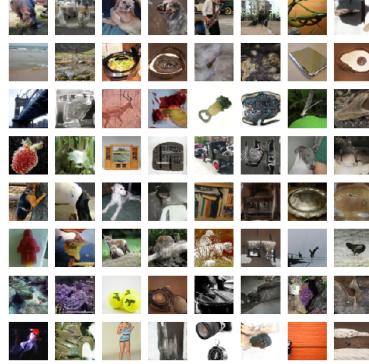


(b) CelebA reconstructions.

Figure 4: Samples and reconstructions on the CelebA dataset. For the reconstructions, odd columns are original samples from the validation set and even columns are corresponding reconstructions.



(a) Tiny ImageNet samples.



(b) Tiny ImageNet reconstructions.

Figure 5: Samples and reconstructions on the Tiny ImageNet dataset. For the reconstructions, odd columns are original samples from the validation set and even columns are corresponding reconstructions.

2.5 Relationship with the Jensen-Shannon divergence

Proposition 2. *Under an optimal discriminator D^* , the generator minimizes the Jensen-Shanon divergence which attains its minimum if and only if $q(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}, \mathbf{z})$.*

Proof. The proof is a straightforward extension of the proof in Goodfellow *et al.* (2014). \square

2.6 Multilayer extensions of ALI

It is straightforward to extend the ALI framework to multiple stochastic layers, i.e.,

$$p(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L) = p(\mathbf{z}_L)p(\mathbf{z}_{L-1} | \mathbf{z}_L) \cdots p(\mathbf{z}_1 | \mathbf{z}_2, \dots, \mathbf{z}_L)p(\mathbf{x} | \mathbf{z}_1, \dots, \mathbf{z}_L) \quad (6)$$

$$q(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L) = q(\mathbf{x})p(\mathbf{z}_1 | \mathbf{x})p(\mathbf{z}_2 | \mathbf{z}_1, \mathbf{x}) \cdots p(\mathbf{z}_L | \mathbf{z}_1, \dots, \mathbf{z}_{L-1}, \mathbf{x}). \quad (7)$$

In this case, the discriminator takes as input either $(\mathbf{x}, \hat{\mathbf{z}}_{1:L}) \sim q(\mathbf{x}, \mathbf{z}_{1:L})$ or $(\tilde{\mathbf{x}}, \mathbf{z}_{1:L}) \sim p(\mathbf{x}, \mathbf{z}_{1:L})$ and tries to determine which joint the samples were drawn from. The generator trains the distributions $q(\mathbf{z}_{1:L} | \mathbf{x})$, $p(\mathbf{z}_{1:L})$ and $p(\mathbf{x} | \mathbf{z}_{1:L})$ to fool the discriminator.

3 Related Work

Other recent papers explore hybrid approaches to generative modelling. One such approach is to relax the probabilistic interpretation of the VAE model by replacing either the KL-divergence term or the reconstruction term with variants that have better properties. 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 $q(\mathbf{z})$ and the prior. Other papers explore replacing the reconstruction term with either GANs or auxiliary networks. Larsen *et al.* (2015) collapse the decoder of a VAE and the generator of a GAN into one network in order to supplement the reconstruction loss with a learned similarity metric. Lamb *et al.* (2016) use the hidden layers of a pre-trained classifier as auxiliary reconstruction losses to help the VAE focus on higher-level details when reconstructing. Dosovitskiy and Brox (2016) combine both ideas into a unified loss function.

ALI's approach is also reminiscent of the adversarial autoencoder model, which employs a GAN to distinguish between samples from the approximate posterior distribution $q(\mathbf{z} | \mathbf{x})$ and prior samples. However, unlike adversarial autoencoders, no explicit reconstruction loss is being optimized in ALI, and the discriminator receives joint pairs of samples (\mathbf{x}, \mathbf{z}) rather than marginal \mathbf{z} samples.

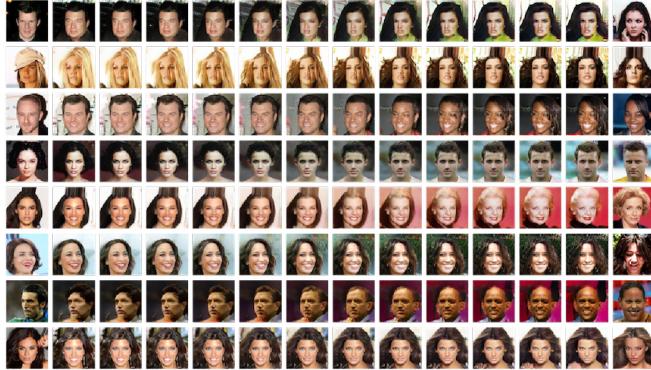


Figure 6: Latent space interpolations on the CelebA validation set. Left and right columns correspond to the original pairs \mathbf{x}_1 and \mathbf{x}_2 , and the columns in between correspond to the decoding of latent representations interpolated linearly from \mathbf{z}_1 to \mathbf{z}_2 .

Method	Error rate
KNN (as reported in Zhao <i>et al.</i> (2015))	77.93%
TSVM (Vapnik, 1998)	66.55%
VAE (M1 + M2) (Kingma <i>et al.</i> , 2014)	36.02%
SWWAE without dropout (Zhao <i>et al.</i> , 2015)	27.83%
SWWAE with dropout (Zhao <i>et al.</i> , 2015)	23.56%
DCGAN + L2-SVM (Radford <i>et al.</i> , 2015)	22.18% ($\pm 1.13\%$)
SDGM (Maaløe <i>et al.</i>, 2016)	16.61% ($\pm 0.24\%$)
ALI (ours)	19.14% ($\pm 0.50\%$)

Table 1: SVHN test set error rates for semi-supervised learning. 1000 training examples were used.

4 Experimental results

We applied ALI to four different datasets, namely CIFAR10 (Krizhevsky and Hinton, 2009), SVHN (Netzer *et al.*, 2011), CelebA (Liu *et al.*, 2015) and a center-cropped, 64×64 version of the ImageNet dataset (Russakovsky *et al.*, 2015).³

In addition to these larger scale experiments, we were interested in studying how modeling the joint distribution impacted ALI in its ability to capture the underlying data density. To this end, we applied ALI to a 2-D synthetic dataset.

The use of batch normalization (Ioffe and Szegedy, 2015) in all layers (except the output layer) of $G_z(\mathbf{x})$ and $G_x(\mathbf{z})$ was crucial to the generator’s success in fooling the discriminator. We also found that in order to stabilize training, batch normalization should not be applied in layers of the discriminator which are influenced by \mathbf{z} .

Transposed convolutions are used in $G_x(\mathbf{z})$. This operation corresponds to the transpose of the matrix representation of a convolution, i.e., the gradient of the convolution with respect to its inputs. For more details about transposed convolutions, see Dumoulin and Visin (2016).

4.1 Samples

For each dataset, samples are presented (Figures 2a, 3a, 4a and 5a). They exhibit the same image fidelity as samples from other adversarially-trained models.

³The code for all experiments can be found at <https://github.com/IshmaelBelghazi/ALI>. Readers can also consult the accompanying website at <https://ishmaelbelghazi.github.io/ALI>.

4.2 Latent space interpolations

As a sanity check for overfitting, we look at latent space interpolations between validation set examples (Figure 6). We sample pairs of validation set examples \mathbf{x}_1 and \mathbf{x}_2 and project them into \mathbf{z}_1 and \mathbf{z}_2 by sampling from the encoder. We then linearly interpolate between \mathbf{z}_1 and \mathbf{z}_2 and pass the intermediary points through the decoder to plot the input-space interpolations.

We observe smooth transitions between pairs of example, and intermediary images remain believable. This is an indicator that ALI is not concentrating its probability mass exclusively around training examples, but rather has learned latent features that generalize well.

4.3 Reconstructions

We also qualitatively evaluate the fit between the conditional distribution $q(\mathbf{z} \mid \mathbf{x})$ and the posterior distribution $p(\mathbf{z} \mid \mathbf{x})$ by sampling $\hat{\mathbf{z}} \sim q(\mathbf{z} \mid \mathbf{x})$ and $\hat{\mathbf{x}} \sim p(\mathbf{x} \mid \mathbf{z} = \hat{\mathbf{z}})$ (Figures 2b, 3b, 4b and 5b). This corresponds to reconstructing the input in a VAE setting. Interestingly, even though reconstruction quality is not explicitly enforced in the objective function, the trained models achieve good reconstruction in practice, which suggests that the conditional $q(\mathbf{z} \mid \mathbf{x})$ is indeed a good approximation of the posterior $p(\mathbf{z} \mid \mathbf{x})$.

Reconstructions in ALI are quite different from reconstructions in VAE-like models. Instead of focusing on achieving a pixel-perfect reconstruction, ALI produces reconstructions that often faithfully represent more abstract features of the input images, while making mistakes in capturing exact object placement, color, style and (in extreme cases) object identity. These reconstructions suggest that the ALI latent variable representations are potentially more invariant to these less interesting factors of variation in the input and do not devote model capacity to capturing these factors.

4.4 Semi-supervised learning

The fact that ALI’s latent representation tends to focus on semantic information at the expense of low-level details leads us to believe that ALI may be well suited to semi-supervised tasks. We empirically verify this by achieving a competitive performance on the semi-supervised SVHN classification task. Results are presented in Table 1. The SDGM’s performance is achieved via a carefully designed two-layer architecture that explicitly takes label information into account in learning the representation. We expect that ALI would also gain by taking account of label information in learning the representation.

We follow the procedure outlined by Radford *et al.* (2015). We train an L2-SVM on the learned representations of a model trained on SVHN. The last three hidden layers of the encoder as well as its output are concatenated to form a 8960-dimensional feature vector. A 10,000 example held-out validation set is taken from the training set and is used for model selection. The SVM is trained on 1000 examples taken at random from the remainder of the training set. The test error rate is measured for 100 different SVMs trained on different random 1000-example training sets, and the average error rate is measured along with its standard deviation.

4.5 Comparison with GAN on a toy task

Figure 7 shows a comparison of the ability of GAN and ALI to fit a simple 2-dimensional synthetic gaussian mixture dataset. The decoder and discriminator networks are matched between ALI and GAN, and the hyperparameters are the same. In this experiment, ALI converges faster than GAN and to a better solution. Despite the relative simplicity of the data distribution, GAN partially failed to converge to the distribution, ignoring the central mode.

The toy task also exhibits nice properties of the features learned by ALI: when mapped to the latent space, data samples cover the whole prior, and they get clustered by mixture components, with a clear separation between each mode.

5 Conclusion

We introduced the adversarially learned inference (ALI) model, which jointly learns a generation network and an inference network using an adversarial process. The model learns mutually coherent

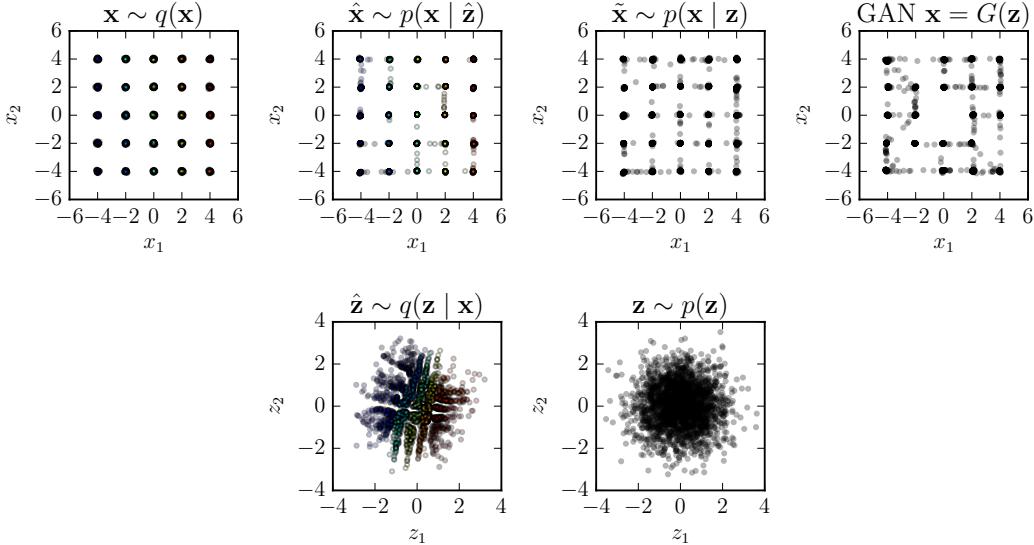


Figure 7: Comparison of ALI and GAN on a synthetic gaussian mixture dataset. In the top row, from left to right: data samples, ALI reconstructions, ALI samples, GAN samples. In the bottom row, from left to right: ALI encodings, prior samples. The nine mixture modes, their ALI encodings and their ALI reconstructions have been color-coded to facilitate visualization.

inference and generation networks, as exhibited by its reconstructions. The induced latent variable mapping is shown to be useful, achieving competitive results on semi-supervised SVHN house number classification.

Acknowledgments

The authors would like to acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada. We would also like to thank the developers of Theano (Bergstra *et al.*, 2010; Bastien *et al.*, 2012; Theano Development Team, 2016), Blocks and Fuel (van Merriënboer *et al.*, 2015), which were used extensively for the paper. Finally, we would like to thank Yoshua Bengio, David Warde-Farley, Yaroslav Ganin and Laurent Dinh for their valuable feedback.

References

- Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012). Theano: new features and speed improvements.
- Bengio, Y., Thibodeau-Laufer, E., Alain, G., and Yosinski, J. (2013a). Deep generative stochastic networks trainable by backprop. *arXiv preprint arXiv:1306.1091*.
- Bengio, Y., Léonard, N., and Courville, A. (2013b). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.
- Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX.
- Denton, E. L., Chintala, S., Fergus, R., *et al.* (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in Neural Information Processing Systems*, pages 1486–1494.
- Dosovitskiy, A. and Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*.

- Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.
- Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*.
- Gregor, K., Danihelka, I., Graves, A., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.
- Kingma, D. P. (2013). Fast gradient-based inference with continuous latent variable models in auxiliary form. *arXiv preprint arXiv:1306.0733*.
- Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.
- Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images.
- Lamb, A., Dumoulin, V., and Courville, A. (2016). Discriminative regularization for generative models. *arXiv preprint arXiv:1602.03220*.
- Larsen, A. B. L., Sønderby, S. K., and Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*.
- Lin, J. (1991). Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, **37**(1), 145–151.
- Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3730–3738.
- Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. (2016). Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*.
- Makhzani, A., Shlens, J., Jaitly, N., and Goodfellow, I. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, Spain.
- Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.
- Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*.
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, **115**(3), 211–252.
- Salimans, T., Kingma, D. P., and Welling, M. (2014). Markov chain monte carlo and variational inference: Bridging the gap. *arXiv preprint arXiv:1410.6460*.
- Theano Development Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

- Theis, L., van den Oord, A., and Bethge, M. (2015). A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*.
- van Merriënboer, B., Bahdanau, D., Dumoulin, V., Serdyuk, D., Warde-Farley, D., Chorowski, J., and Bengio, Y. (2015). Blocks and fuel: Frameworks for deep learning. *arXiv preprint arXiv:1506.00619*.
- Vapnik, V. N. (1998). *Statistical Learning Theory*. Wiley-Interscience.
- Zhao, J., Mathieu, M., Goroshin, R., and Lecun, Y. (2015). Stacked what-where auto-encoders. *arXiv preprint arXiv:1506.02351*.

A Hyperparameters

	Operation	Kernel	Strides	Feature maps	BN?	Dropout	Nonlinearity
$G_z(x) - 3 \times 32 \times 32$ input							
	Convolution	5×5	1×1	32	✓	0.0	Leaky ReLU
	Convolution	4×4	2×2	64	✓	0.0	Leaky ReLU
	Convolution	4×4	1×1	128	✓	0.0	Leaky ReLU
	Convolution	4×4	2×2	256	✓	0.0	Leaky ReLU
	Convolution	4×4	1×1	512	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	512	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	128	✗	0.0	Linear
$G_x(z) - 64 \times 1 \times 1$ input							
	Transposed convolution	4×4	1×1	256	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	2×2	128	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	1×1	64	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	2×2	32	✓	0.0	Leaky ReLU
	Transposed convolution	5×5	1×1	32	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	32	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	3	✗	0.0	Sigmoid
$D(x) - 3 \times 32 \times 32$ input							
	Convolution	5×5	1×1	32	✗	0.2	Maxout
	Convolution	4×4	2×2	64	✗	0.5	Maxout
	Convolution	4×4	1×1	128	✗	0.5	Maxout
	Convolution	4×4	2×2	256	✗	0.5	Maxout
	Convolution	4×4	1×1	512	✗	0.5	Maxout
$D(z) - 64 \times 1 \times 1$ input							
	Convolution	1×1	1×1	512	✗	0.2	Maxout
	Convolution	1×1	1×1	512	✗	0.5	Maxout
$D(x, z) - 1024 \times 1 \times 1$ input							
	<i>Concatenate $D(x)$ and $D(z)$ along the channel axis</i>						
	Convolution	1×1	1×1	1024	✗	0.5	Maxout
	Convolution	1×1	1×1	1024	✗	0.5	Maxout
	Convolution	1×1	1×1	1	✗	0.5	Sigmoid
Optimizer Adam ($\alpha = 10^{-4}$, $\beta_1 = 0.5$, $\beta_2 = 10^{-3}$)							
Batch size 100							
Epochs 6475							
Leaky ReLU slope, maxout pieces 0.1, 2							
Weight, bias initialization Isotropic gaussian ($\mu = 0$, $\sigma = 0.01$), Constant(0)							

Table 2: CIFAR10 model hyperparameters. Maxout layers (Goodfellow *et al.*, 2013) are used in the discriminator.

	Operation	Kernel	Strides	Feature maps	BN?	Dropout	Nonlinearity
$G_z(x) - 3 \times 32 \times 32$ input							
Convolution	5×5	1×1	32		✓	0.0	Leaky ReLU
Convolution	4×4	2×2	64		✓	0.0	Leaky ReLU
Convolution	4×4	1×1	128		✓	0.0	Leaky ReLU
Convolution	4×4	2×2	256		✓	0.0	Leaky ReLU
Convolution	4×4	1×1	512		✓	0.0	Leaky ReLU
Convolution	1×1	1×1	512		✓	0.0	Leaky ReLU
Convolution	1×1	1×1	512		✗	0.0	Linear
$G_x(z) - 256 \times 1 \times 1$ input							
Transposed convolution	4×4	1×1	256		✓	0.0	Leaky ReLU
Transposed convolution	4×4	2×2	128		✓	0.0	Leaky ReLU
Transposed convolution	4×4	1×1	64		✓	0.0	Leaky ReLU
Transposed convolution	4×4	2×2	32		✓	0.0	Leaky ReLU
Transposed convolution	5×5	1×1	32		✓	0.0	Leaky ReLU
Convolution	1×1	1×1	32		✓	0.0	Leaky ReLU
Convolution	1×1	1×1	3		✗	0.0	Sigmoid
$D(x) - 3 \times 32 \times 32$ input							
Convolution	5×5	1×1	32		✗	0.2	Leaky ReLU
Convolution	4×4	2×2	64		✓	0.2	Leaky ReLU
Convolution	4×4	1×1	128		✓	0.2	Leaky ReLU
Convolution	4×4	2×2	256		✓	0.2	Leaky ReLU
Convolution	4×4	1×1	512		✓	0.2	Leaky ReLU
$D(z) - 256 \times 1 \times 1$ input							
Convolution	1×1	1×1	512		✗	0.2	Leaky ReLU
Convolution	1×1	1×1	512		✗	0.2	Leaky ReLU
$D(x, z) - 1024 \times 1 \times 1$ input							
<i>Concatenate $D(x)$ and $D(z)$ along the channel axis</i>							
Convolution	1×1	1×1	1024		✗	0.2	Leaky ReLU
Convolution	1×1	1×1	1024		✗	0.2	Leaky ReLU
Convolution	1×1	1×1	1		✗	0.2	Sigmoid
Optimizer	Adam ($\alpha = 10^{-4}$, $\beta_1 = 0.5$, $\beta_2 = 10^{-3}$)						
Batch size	100						
Epochs	100						
Leaky ReLU slope	0.01						
Weight, bias initialization	Isotropic gaussian ($\mu = 0$, $\sigma = 0.01$), Constant(0)						

Table 3: SVHN model hyperparameters.

	Operation	Kernel	Strides	Feature maps	BN?	Dropout	Nonlinearity
$G_z(x) - 3 \times 64 \times 64$ input							
	Convolution	2×2	1×1	64	✓	0.0	Leaky ReLU
	Convolution	7×7	2×2	128	✓	0.0	Leaky ReLU
	Convolution	5×5	2×2	256	✓	0.0	Leaky ReLU
	Convolution	7×7	2×2	256	✓	0.0	Leaky ReLU
	Convolution	4×4	1×1	512	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	512	✗	0.0	Linear
$G_x(z) - 512 \times 1 \times 1$ input							
	Transposed convolution	4×4	1×1	512	✓	0.0	Leaky ReLU
	Transposed convolution	7×7	2×2	256	✓	0.0	Leaky ReLU
	Transposed convolution	5×5	2×2	256	✓	0.0	Leaky ReLU
	Transposed convolution	7×7	2×2	128	✓	0.0	Leaky ReLU
	Transposed convolution	2×2	1×1	64	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	3	✗	0.0	Sigmoid
$D(x) - 3 \times 64 \times 64$ input							
	Convolution	2×2	1×1	64	✓	0.0	Leaky ReLU
	Convolution	7×7	2×2	128	✓	0.0	Leaky ReLU
	Convolution	5×5	2×2	256	✓	0.0	Leaky ReLU
	Convolution	7×7	2×2	256	✓	0.0	Leaky ReLU
	Convolution	4×4	1×1	512	✓	0.0	Leaky ReLU
$D(z) - 512 \times 1 \times 1$ input							
	Convolution	1×1	1×1	1024	✗	0.2	Leaky ReLU
	Convolution	1×1	1×1	1024	✗	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	2048	✗	0.2	Leaky ReLU
	Convolution	1×1	1×1	2048	✗	0.2	Leaky ReLU
	Convolution	1×1	1×1	1	✗	0.2	Sigmoid
Optimizer Adam ($\alpha = 10^{-4}$, $\beta_1 = 0.5$)							
Batch size 100							
Epochs 123							
Leaky ReLU slope 0.02							
Weight, bias initialization Isotropic gaussian ($\mu = 0$, $\sigma = 0.01$), Constant(0)							

Table 4: CelebA model hyperparameters.

	Operation	Kernel	Strides	Feature maps	BN?	Dropout	Nonlinearity
$G_z(x) - 3 \times 64 \times 64$ input							
	Convolution	4×4	2×2	64	✓	0.0	Leaky ReLU
	Convolution	4×4	1×1	64	✓	0.0	Leaky ReLU
	Convolution	4×4	2×2	128	✓	0.0	Leaky ReLU
	Convolution	4×4	1×1	128	✓	0.0	Leaky ReLU
	Convolution	4×4	2×2	256	✓	0.0	Leaky ReLU
	Convolution	4×4	1×1	256	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	2048	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	2048	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	512	✗	0.0	Linear
$G_x(z) - 256 \times 1 \times 1$ input							
	Convolution	1×1	1×1	2048	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	256	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	1×1	256	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	2×2	128	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	1×1	128	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	2×2	64	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	1×1	64	✓	0.0	Leaky ReLU
	Transposed convolution	4×4	2×2	64	✓	0.0	Leaky ReLU
	Convolution	1×1	1×1	3	✗	0.0	Sigmoid
$D(x) - 3 \times 64 \times 64$ input							
	Convolution	4×4	2×2	64	✗	0.2	Leaky ReLU
	Convolution	4×4	1×1	64	✓	0.2	Leaky ReLU
	Convolution	4×4	2×2	128	✓	0.2	Leaky ReLU
	Convolution	4×4	1×1	128	✓	0.2	Leaky ReLU
	Convolution	4×4	2×2	256	✓	0.2	Leaky ReLU
	Convolution	4×4	1×1	256	✓	0.2	Leaky ReLU
$D(z) - 256 \times 1 \times 1$ input							
	Convolution	1×1	1×1	2048	✗	0.2	Leaky ReLU
	Convolution	1×1	1×1	2048	✗	0.2	Leaky ReLU
$D(x, z) - 2304 \times 1 \times 1$ input							
Concatenate $D(x)$ and $D(z)$ along the channel axis							
	Convolution	1×1	1×1	4096	✗	0.2	Leaky ReLU
	Convolution	1×1	1×1	4096	✗	0.2	Leaky ReLU
	Convolution	1×1	1×1	1	✗	0.2	Sigmoid
Optimizer	Adam ($\alpha = 10^{-4}$, $\beta_1 = 0.5$, $\beta_2 = 10^{-3}$)						
Batch size	128						
Epochs	125						
Leaky ReLU slope	0.01						
Weight, bias initialization	Isotropic gaussian ($\mu = 0$, $\sigma = 0.01$), Constant(0)						

Table 5: Tiny ImageNet model hyperparameters.

B A generative story for ALI

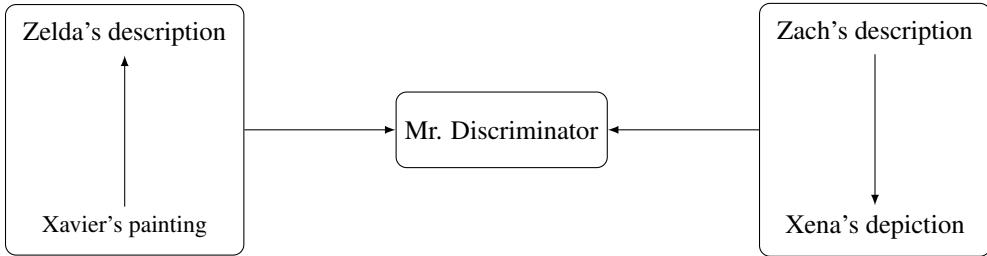


Figure 8: A Circle of Infinite Painters' view of the ALI game.

The Circle of Infinite Painters is a very prolific artistic group. Very little is known about the Circle, but what we do know is that it is composed of two very brilliant artists. It has produced new paintings almost daily for more than twenty years, each one more beautiful than the others. Not only are the paintings exquisite, but their title and description is by itself a literary masterpiece.

However, some scholars believe that things might not be as they appear: certain discrepancies in the Circle's body of work hints at the Circle being composed of more than one artistic duo. This is what Joseph Discriminator, art critique and world expert on the Circle, believes. He's recently been working intensively on the subject. Without knowing it, he's right: the Circle is not one, but two artistic duos.

Xavier and Zach Prior form the creative component of the group. Xavier is a painter and can, in one hour and starting from nothing, produce a painting that would make any great painter jealous. Impossible however for him to explain what he's done: he works by intuition alone. Zach is an author and his literary talent equals Xavier's artistic talent. His verb is such that the scenes he describes could just as well be real.

By themselves, the Prior brothers cannot collaborate: Xavier can't paint anything from a description and Zach is bored to death with the idea of describing anything that does not come out of his head. This is why the Prior brothers depend on the Conditional sisters so much.

Zelda Conditional has an innate descriptive talent: she can examine a painting and describe it so well that the original would seem like an imitation. Xena Conditional has a technical mastery of painting that allows her to recreate everything that's described to her in the most minute details. However, their creativity is inversely proportional to their talent: by themselves, they cannot produce anything of interest.

As such, the four members of the Circle work in pairs. What Xavier paints, Zelda describes, and what Zach describes, Xena paints. They all work together to fulfill the same vision of a unified Circle of Infinite Painters, a whole greater than the sum of its parts.

This is why Joseph Discriminator's observations bother them so much. Secretly, the Circle put Mr. Discriminator under surveillance. Whatever new observation he's made, they know right away and work on attenuating the differences to maintain the illusion of a Circle of Infinite Painters made of a single artistic duo.

Will the Circle reach this ideal, or will it be unmasksed by Mr. Discriminator?