Understanding Mutual Information and its Use in InfoGAN

Katrina Evtimova *
Center for Data Science
New York University
kve216@nyu.edu

Andrew Drozdov *
Courant Institute for Mathematical Sciences
New York University
apd283@nyu.edu

Abstract

Interpretable variables are useful in generative models. Generative Adversarial Networks (GANs) are generative models that are flexible in their input. The Information Maximizing GAN (InfoGAN) ties the output of the generator to a component of its input called the latent codes. By forcing the output to be tied to this input component, we can control some properties of the output representation. It is notoriously difficult to find the Nash equilibrium when jointly training the discriminator and generator in a GAN. We uncover some successful and unsuccessful configurations for generating images using InfoGAN.

1 Introduction

The goal of a generative model is to approximate the distribution of some real data. In the case of labeled data and supervised discriminative learning, it is easy to approximate the distribution of labels using a classifier. It is somewhat more difficult to approximate the likelihood of every pixel and the structure between pixels when generating images. Approached from the supervised discriminative perspective, there will be a large label space that makes learning intractable.

When generating images without labels, Autoencoders provide an alternative solution. They treat the input as the target and calculate a reconstruction loss. That being said, there are many tricks required to get learning with Autoencoders to be effective such as choosing the right structure for intermediate layers and encouraging a sparse representation.

Even though Autoencoders and their variants (such as the Variational Autoencoder (VAE) [KW13] and Denoising Autoencoder [Vin+10]) are useful, it is desirable to have alternative models with different properties. One such model is the Generative Adversarial Network (GAN) [Goo+14]. The generator in GANs is similar in structure to the part of VAEs which approximates the probability of the input, say an image, given some latent state. Since the generator does not get an image as input, it is not possible to use reconstruction loss. Instead, GANs use a discriminator. We run the discriminator once on real input and once on generated input, then optimize using a summation of these two outputs. This is commonly known as the minimax game. In practice, one can try finding a balance in the "schedule", giving the discriminator batches of N_{real} real inputs and N_{fake} generated inputs to keep it from becoming too confident in either direction. However, at a workshop in this past NIPS, it has been recommended to avoid wasting time and resources in finding this schedule [Chi+16].

An obvious problem in the typical GAN setup is that the generator might become specialized at creating only certain types of images. In the case of generating handwritten digits, there is no specific component that indicates that the generator should uniformly create digits from 0-9 because the generator has no knowledge of which real image is being selected. We would like to force the GAN

*Both authors contributed equally and were advised by David Sontag.

to create images uniformly among the possible classes. Fortunately, there has been recent work that builds a setup along these lines.

InfoGAN is a model that retains mutual information between its input and output. Unlike the Denoising Autoencoder [Vin+10], InfoGAN gives freedom to specify which component of the input should be retained. If one of these components is a categorical variable, and the model learns to retain the class information based on this categorical variable, then we can solve the problem mentioned in the previous paragraph. Training models in this fashion has been otherwise called disentangling an interpretable representation [Kul+15].

Providing a categorical variable and then incorporating mutual information into the loss of the generator sounds reasonable, but it is actually remarkable that class information is retained. After all, there is no specific component indicating that the categorical variable represents class information. It could just as equally represent the tilt or width of the images. In this work we intend to empirically evaluate how well different latent codes work, and discuss ideas about why mutual information works so well.

The motivating questions for this paper are:

- How is InfoGAN any different from Vanilla GAN?
- Is prior knowledge about the data critical when assigning latent codes?
- Is the encoder network a classifier or something else?
- How can we leverage interpretable latent codes in other scenarios?

The first two questions are explored in the following sections. Meanwhile the second two remain unanswered, but we believe we've developed an understanding towards addressing them in future work.

2 Background

2.1 Vanilla GAN

General Adversarial Networks (GANs) [Goo+14] are a powerful framework for extracting high-level representations of data. The main idea behind GANs is to simultaneously train two networks acting against one another - a generator G and a discriminator D. The generator G creates a data sample given random noise z which comes from a prior distribution $p_z(z)$. The goal of G is to learn the underlying data distribution while the task of the discriminator is to distinguish between samples coming from G and samples coming from the original training data. The "vanilla" GAN objective function is given by:

$$V(D,G) = \mathbb{E}_{x\sim P_{\text{data}}}[\log(D(x))] + \mathbb{E}_{z\sim \text{noise}}[\log(1 - D(G(z)))] \tag{1}$$

If we set p_r to be the real data distribution (which is unknown but can be sampled from) and p_g to be the distribution implicitly defined by the generator G (obtained from the generated samples $G(z)$), then [Goo+14] show that GANs optimize the Jensen-Shannon divergence derived from the KL divergence:

$$JSD(p_r \| p_g) = KL\Big(p_r \Big\| \frac{p_r + p_g}{2}\Big) + KL\Big(p_g \Big\| \frac{p_r + p_g}{2}\Big) \tag{2}$$

Although this framework is theoretically appealing, a common practical issue with GANs is that optimizing the loss function can suffer from instability, making them difficult to train [GBC16]. The original GAN paper [Goo+14] mentions the issue of saturation or, in other words, insufficient gradients for the generator G to learn. This phenomenon is observed early in training when G's performance is weak and the discriminator D can easily tell apart real from generated images. More specifically, the term $\log(1 - D(G(z)))$ in equation (1) is close to zero when D assigns a low probability for $G(z)$ coming from the real data distribution. The proposed approach for addressing this issue is the following: instead of minimizing $\log(1 - D(G(z)))$, the generator G can initially be trained to maximize $D(G(z))$. According to [Chi+16], however, this alternative objective does not help with stabilizing optimization. Understanding the optimization dynamics of GANs is an active area of research and [Chi+16] presents some key techniques for training GANs. In this paper, we justify empirically one of them, namely perturbing the generator's input.

2.2 InfoGAN

InfoGAN [Che+16] builds upon the the GAN framework by generating interpretable representations for the latent variables in the model. The key idea is to split latent variables into a set c of interpretable ones and a source of uninterpretable noise z. Interpretablity is encouraged by adding an extra term in the original GAN objective function capturing the mutual information between the interpretable variables c and the output from the generator. More precisely, the InfoGAN minimax optimization is defined as:

$$\min_G \max_D V_{IG}(D, G) = V(D, G) - \lambda \cdot \mathbb{I}(c; G(z, c)) \tag{3}$$

where G is the generator, D is the discriminator, z is the uninterpretable noise, c encodes the salient latent codes and the Mutual Information ($\mathbb{I}$) is given by:

$$\mathbb{I}(c; G(z, c)) = \text{Entropy(c)} - \text{Entropy(c|G(z, c))} \tag{4}$$

As seen in (3), the novelty of the InfoGAN objective function compared to the regular GAN one is the introduction of a regularization term involving the mutual information between the latent codes c and the generator G. Note that the second entropy term in (4) requires access to the posterior $p(c|G(z, c))$ which is approximated by the discriminator network.

As explained in [Vin+10], mutual information can be utilized whenever we are interested in learning a parametrized mapping from a given input X to a higher level representation Y which preserves information about the original input. More formally, if we are interested in a mapping $q(Y|X; \theta)$ which preserves information about X, where θ are the parameters to be learned, this can be achieved by maximizing the mutual information between X and Y, $\mathbb{I}(X; Y)$. This is referred to as the *infomax principle* and in the case of InfoGAN translates to maximizing the mutual information between the latent codes c and the output from the generative network, $G(z, c)$.

Moreover, [Vin+10] show that the task of maximizing mutual information is essentially equivalent to training an autoencoder to minimize reconstruction error. What this implies is that learning a mapping that ties c with $G(z, c)$ can be interpreted as incorporating or encoding c as closely as possible in the output $G(z, c)$. In a sense, the codes in c serve as a label for the outputs generated from G.

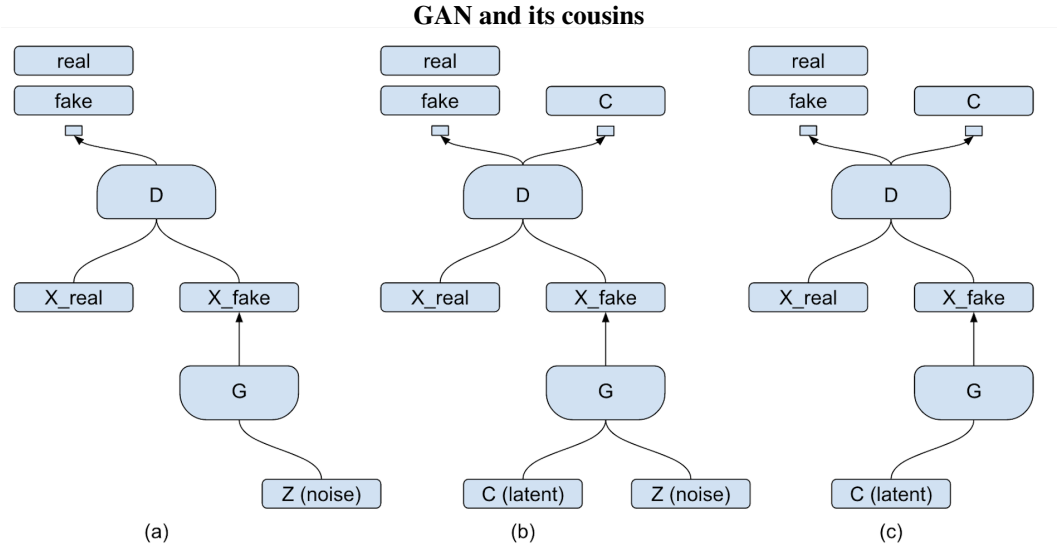


Figure 1: Graphical representation of different GAN configurations: (a) Vanilla GAN; (b) InfoGAN; (c) InfoGAN w/ only latent codes. The style of these diagrams was inspired by [OOS16].

3 Methods

3.1 InfoGAN and MNIST

The InfoGAN paper runs two experiments that hold particular interest. The first one is a GAN with an encoder network Q that maximizes the lower bound to the mutual information $\mathbb{I}(c; G(z, c))$ where

c contains one categorical and two continuous latent codes. The discriminator D, generator G, and encoder Q are optimized jointly. The second one has a similar setup but Q is only tied to D.

The first experiment demonstrates a configuration where mutual information is maximized while the second experiment does not, showing that there is nothing implicitly tying the input of the generator to its output (although perhaps this phenomenon could be observed in a different neural network architecture).

There are a few configurations that are not attempted in the InfoGAN paper that we attempt.

No noise. Why use a noise vector when there are interpretable latent codes? MNIST seems simple enough that there should still be plausible images.

No noise and no continuous codes. Using only a single categorical code, we would expect there to be only N possible images, where N are how many values this variable can represent.

No encoder. Although the original paper already has some negative results to this light, disregarding the encoder entirely is slightly different, and perhaps the GAN can succeed in realizing its latent codes have meaning.

3.2 InfoGAN and BasicProp

Does InfoGAN work with any data? This is a question we intend to investigate. Using synthetic data is a proven tactic for evaluating the power of models. We evaluate InfoGAN with a new dataset called BasicProp, laying the foundation to test the limitations of InfoGAN and similar models in a precise manner.

BasicProp has two flavors: line and rectangles. The line version is meant to be a proxy for MNIST. It has 10 positions, and therefore can be bucketed into 10 categories. The rectangles version is similar, except it contains two parallel rectangles, each of 10 possible heights, so 10^2 categories. We believe it would be interesting to see whether these extended categories can be captured with the same configuration, and optimistically hope the continuous codes might capture something useful such as a ratio of the heights.

BasicProp matches the dimensionality and color properties of MNIST. Nonetheless, we found it surprisingly difficult to train InfoGAN with BasicProp, which we discuss in more depth in a later section. Both variants of the dataset and the code used to generate them are available on our Github repository[2].

4 Experimental Setup

Table summarizing our experiments :

Model	$G(c, z)$	$\mathbb{I}(c; g)$	λ
Vanilla GAN (original)	$c = \emptyset$, $z \sim p_z$	-	0
Vanilla GAN + latent	$c = \{c_{cat}, c_{cont1}, c_{cont2}\}$, $z \sim p_z$	-	0
InfoGAN (original)	$c = \{c_{cat}, c_{cont1}, c_{cont2}\}$, $z \sim p_z$	$g = G(z, c)$	1
InfoGAN (only latent)	$c = \{c_{cat}, c_{cont1}, c_{cont2}\}$, no z	$g = G(c)$	1
InfoGAN (only latent categorical)	$c = \{c_{cat}\}$, no z	$g = G(c_{cat})$	1
InfoGAN (thought on $\mathbb{I}$)	$c = \{c_{cat}, c_{cont1}, c_{cont2}\}$, $z \sim p_z$	$g = x$	1
BASICPROP Line	$c = \{c_{cat}, c_{cont1}, c_{cont2}\}$, $z \sim p_z$	$g = G(z, c)$	1
BASICPROP Rectangle	$c = \{c_{cat}, c_{cont1}, c_{cont2}\}$, $z \sim p_z$	$g = G(z, c)$	1

5 Results & Analysis

Experiments in sections 5.1 - 5.4 use InfoGAN for MNIST, 5.5 - 5.6 Vanilla GAN (no mutual information) for MNIST, and 5.7 - 5.8 InfoGAN for BasicProp.

[2] `https://github.com/kevtimova/InfoGAN`

5.1 InfoGAN: Baseline

In [Che+16], three interpretable latent codes are fed to the generator G - one categorical (c_1) and two continuous uniformly distributed (c_2, c_3). Remarkably, the categorical code c_1 is able to capture the digit class of each generated image while c_2 and c_3 represent rotation and width, respectively.

5.2 InfoGAN: No Latent Noise

The incompressible noise variable z is removed and only interpretable latent codes c are present - one categorical and two uniform. The Mutual Information term is also present. This setup assumes that the model can capture all the information in the input data using some prior assumptions. This seems very similar to Gaussian Mixture Models. In practice, we found that using only latent codes is a bold way to model data. When using a single categorical code, the model can only generate 10 images. Adding more latent codes, which would usually capture something like tilt of the digit, seems to work well. This suggests that these continuous codes are being used to capture an excessive amount of information that can not necessarily be interpreted.

Generator outputs for different values of the latent codes are presented in Figure 2. It is interesting to note that the first uniform code is able to capture transformations between classes of numbers (e.g. 0 to 8, 4 to 7) compared to the original InfoGAN in which the uniform codes are capturing rotation and width. A possible interpretation of this result is that some of the variance in the data which the noise variable z accounts for in InfoGAN is passed onto the latent codes c in the case when z is removed, resulting in more complex transformations which are harder to interpret.

(a) Categorical Code (b) Uniform Code 1 (c) Uniform Code 2

Figure 2: Experiment removing incompressible noise z.

5.3 InfoGAN: Only a Categorical Code

The only noise variable is a latent categorical code c_{cat} of dimension 10 for the 10 classes of integers in MNIST. Visualization of the generator outputs for different values of c_{cat} are found in Figure 3 (a). It is evident that c_{cat} is still able to detect different classes of integers but we observe that the quality of the outputs and their variance is lower when compared with the outputs from models with higher dimensional latent code space.

5.4 IngoGAN: Thought Experiment on Mutual Information

Modifying the mutual information term $\mathbb{I}(c; G(z, c))$ in the objective function (3) by replacing $G(z, c)$, the sample coming from G, with the original input x. However, computing $\mathbb{I}(c, x)$ is not feasible since computing the entropy incorporated in mutual information would require knowledge of the original data distribution $p_{\text{real}}(X)$ which is not available. The generator mapping $G(z, c)$ is the closest approximation to $p_{\text{real}}(X)$ available in the InfoGAN model framework.

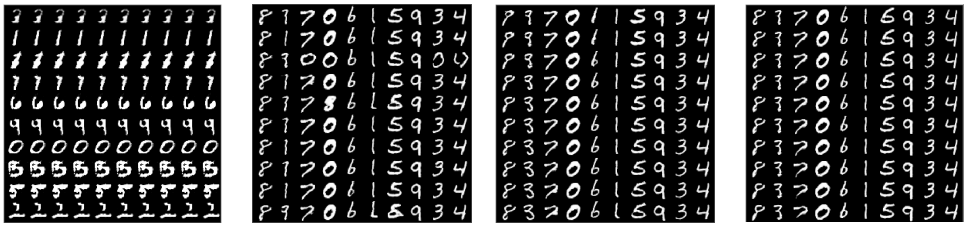
5.5 Vanilla GAN: No Latent Codes

A Vanilla GAN was trained - one without interpretable latent variables and no Mutual Information term in the objective function. We observed that training of the generator was unstable using the

built-in settings. Following advice from [Chi+16] and using SGD as optimization method for the discriminator and Adam for the generator did not yield better results, either. It is possible that adding noise to the real input images might have helped with training, as suggested in Section 6.

5.6 Vanilla GAN: Latent Codes

There is no mutual information but parts of the input fed to the generator G have prior distributions. As suspected, this model is unsuccessful in disentangling interpretable representations of the input data displayed in Figure 3 (b, c, d).



(a) InfoGAN w/ only c_{cat} (b) Vanilla GAN c_{cat} (c) Vanilla GAN c_{unif1} (d) Vanilla GAN c_{unif2}

Figure 3: InfoGAN + one categorical, Vanilla GAN + 3 latent codes

5.7 BasicProp: Line

The first example we decide to evaluate in the synthetic dataset is a white vertical line on a black background. It was necessary to add noise to the foreground in order for training to stabilize.

Interestingly, the cross entropy and mutual information between an MNIST run and BasicProp run are similar, but the generator and discriminator losses are drastically different. This indicates that although mutual information makes results more interpretable, having high mutual information is not a proxy for generating good images. This makes sense as the encoder has no premise of real and fake images.

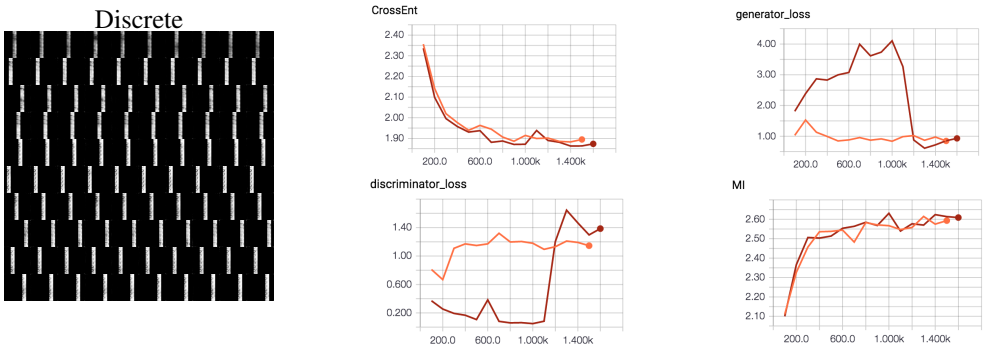


Figure 4: Various graphs that use the InfoGAN setup with 3 latent codes (1 discrete, 2 continuous). The orange line represents MNIST and the red line represents BASICPROP-line. The generated BASICPROP images were unrecognizable until the generator/discriminator loss hit a tipping point.

5.8 BasicProp: Rectangles

This experiment was not successful in disentangling interpretable latent variables. The first attempt used only foreground noise and eventually the ratio of the generated to real images went to 0. By decreasing the learning rate of the discriminator, we can train for longer but the generator never recovers. Adding background noise alleviates the problem, except the discriminator becomes too adept, and the generator still never recovers. A simple experiment which we did not get to try was

adding a second categorical variable. This experiment shows that having prior knowledge of your data is necessary to find a good configuration for InfoGAN.

6 Noisy Input Trick

It was intially surprising that InfoGAN worked with MNIST but not BasicProp. We realized an NaN occured in training because the discriminator became too confident simultaneously as the ratio of the probability of generated to real images went 0. Adding noise to the inputs indeed "stabilized" training. A successful run in our experiments is distinct in that there is a cliff in the loss when the generator finally succeeds to generate something that looks reasonable. Other runs might have near constant generator loss or a constantly increasing one.

The question is why did this noise help when it wasn't required for MNIST? The BasicProp data initially has pixels of only 0 and 255, making the probability of these images far too dense and unlikely to overlap with generated images [Søn16]. Fixing the generator's outputs to 0 or 255 might have been an alternative solution, but using noise is nice in that it is a trick that could be useful in other setups and is worth exploring.

"Instance noise" has been talked about in the GAN literature lately as a method to increase the chance of overlapping support in the real and generated distributions [Søn16; Chi+16; AB16]. Older works tend to use input noise as a regularizer for supervised learning [Von+88; HK92]. Both of these stories have the same goal of making the underlying model generalize better in its predictions and assist in avoiding local minima.

It's been mentioned that L2 weight decay (a specific instance of Tikhonov regularization [Bis95]) is an equivalent alternative for input noise, but [Vin+10] show this was not the case for the denoising autencoder. This could have been interesting to show when training a GAN to generate BasicProp, but we decided that the information from [Vin+10] was sufficient.

7 Conclusion

Our experiments confirm that tying latent codes with output from the generator through mutual information can yield interpretable representations on different datasets. The choice of latent codes depends on the input data. In particular, the codes should correspond to intrinsic properties of the data that they are intended to capture. For example, in the case of MNIST, it is reasonable to introduce a discrete categorical variable for the 10 classes of digits. We also confirm empirically that adding noise to the real data stabilizes training.

References

[Von+88] A Von Lehmen et al. "Factors influencing learning by backpropagation". In: *Neural Networks, 1988., IEEE International Conference on*. IEEE. 1988, pp. 335–341.

[HK92] Lasse Holmstrom and Petri Koistinen. "Using additive noise in back-propagation training". In: *IEEE Transactions on Neural Networks* 3.1 (1992), pp. 24–38.

[Bis95] Chris M Bishop. "Training with noise is equivalent to Tikhonov regularization". In: *Neural computation* 7.1 (1995), pp. 108–116.

[Vin+10] Pascal Vincent et al. "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *J. Mach. Learn. Res.* 11 (Dec. 2010), pp. 3371–3408. ISSN: 1532-4435. URL: `http://dl.acm.org/citation.cfm?id=1756006.1953039`.

[KW13] Diederik P. Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013). URL: `http://arxiv.org/abs/1312.6114`.

[Goo+14] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems*. 2014, pp. 2672–2680.

[Kul+15] Tejas D Kulkarni et al. "Deep convolutional inverse graphics network". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2539–2547.

[AB16] Martin Arjovsky and Leon Bottou. "Towards Principled Methods For Training Generative Adversarial Networks". In: *NIPS 2016 Workshop on Adversarial Training*. In review for ICLR 2017. 2016. URL: https://openreview.net/pdf?id=Hk4_qw5xe.

[Che+16] Xi Chen et al. "InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets". In: *arXiv preprint arXiv:1606.03657* (2016).

[Chi+16] Soumith Chintala et al. "How to Train a GAN? Tips and tricks to make GANs work". In: (2016). URL: https://github.com/soumith/ganhacks#13-add-noise-to-inputs-decay-over-time.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep Learning". Book in preparation for MIT Press. 2016. URL: http://www.deeplearningbook.org.

[OOS16] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional Image Synthesis With Auxiliary Classifier GANs". In: *arXiv preprint arXiv:1610.09585* (2016).

[Søn16] Casper Kaae Sønderby. "Instance Noise: A trick for stabilising GAN training". In: (2016). URL: http://www.inference.vc/instance-noise-a-trick-for-stabilising-gan-training/.

Appendix

A BasicProp++

We initially proposed more variants of BasicProp, displayed in Figure 5. However, we found the line and rectangles to be sufficient.

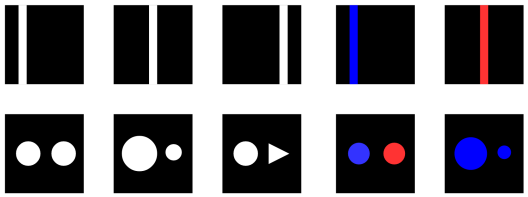


Figure 5: Hypothetical images for the synthetic dataset. This set contains spatial translation, ratios of diameters, colorings, and alternating shapes, all of which could potentially be captured by a combination of categorical and continuous variables.

B BasicProp: Rectangles

We were unsuccessful disentangling latent codes with rectangles. The last recorded images are show in figure 6.

Figure 6: The orange line denotes using only foreground noise, purple line is with a decreased discriminator learning rate with foreground noise, and the blue line uses background and foreground noise with the normal learning rate. The triangle indicates that an NaN was detected. It's evident the NaN is directly linked to the discriminator loss reaching 0.

C Additional Experiments

Below are some experiments which might provide additional insight into what to our study of Mutual Information and interpretable representations.

- Observing that InfoGAN is more or less a GAN with a denoising autoencoder built-in, we wonder if we can reformulate the objective to simply minimize the encoder loss rather than maximize the the mutual information. This is simple enough to setup for categorical codes.
- What happens if more degrees of freedom are provided in the latent codes? In particular, what would happen if the categorical latent code used for MNIST has dimension higher than 10? One hypothesis is that it would be able to discern subcategories in each digit class.

D Semi-Supervised Learning with Categorical Codes

We describe here in detail a method for using InfoGAN as a data augmentation in a general classification task:

1. Configure InfoGAN to have a categorical code that matches the labels of your data.
2. Learn a generative model of your data with InfoGAN using this configuration.
3. Generate Y images of each of the Z classes. Perturb the other latent codes and noise vector during image generation, so that there is some variation in the images.
4. Either using a pretrained classifier or with manual input, identify each of the Z sets of images.
5. Add the new images to the class's set and train a new classifier using the images as if they were real data.
6. This is incredibly similar to the technique of transforming images prior to training such as was done by Lecun et al.