Multi-Modal Generative Adversarial Networks Make Realistic and Diverse but Untrustworthy Predictions When Applied to Ill-posed Problems

John S. Hyatt, Michael S. Lee
Computational & Information Sciences Directorate, DEVCOM Army Research Laboratory
john.s.hyatt11.civ@mail.mil, michael.s.lee131.civ@mail.mil

Abstract

Ill-posed problems can have a distribution of possible solutions rather than a unique one, where each solution incorporates significant features not present in the initial input. We investigate whether cycle-consistent generative neural network models based on generative adversarial networks (GANs) and variational autoencoders (VAEs) can properly sample from this distribution, testing on super-resolution of highly downsampled images. We are able to produce diverse and plausible predictions, but, looking deeper, we find that the statistics of the generated distributions are substantially wrong. This is a critical flaw in applications that require any kind of uncertainty quantification. We trace this to the fact that these models cannot easily learn a bijective, invertible map between the latent space and the target distribution. Additionally, we describe a simple method for constraining the distribution of a *deterministic* encoder's outputs via the Kullback-Leibler divergence without the reparameterization trick used in VAEs.

Introduction

A problem is well-posed if it satisfies three criteria (Hadamard 1902): (1) the problem has a solution, (2) the solution is unique, and (3) the solution is a continuous function of the initial conditions. Many real-world problems of interest are inherently ill-posed, meaning they violate one or more of these criteria, and modeling them correctly remains one of the outstanding challenges in machine learning (ML). Out-of-distribution inputs (data the model was not trained to understand) violate the first criterion, while adversarial examples exist because ML models tend to be very unstable with regard to small, carefully chosen perturbations (Wiyatno et al. 2019), effectively violating the third.

Violations of the second criterion can occur for relatively simple tasks such as classification of ambiguous inputs (Peterson et al. 2019), but they are ubiquitous in generative modeling tasks, where the desired output is complex and high-dimensional. The strongest possible model, given this type of ill-posed problem, is one that estimates the (conditional) posterior distribution of possible solutions.

Depending on the use case, it may not be necessary to model the full posterior; for example, if the objective is purely aesthetic (Pathak et al. 2016; Yang et al. 2019). For safety-critical applications, however, proper risk management requires quantifying the model's predictive uncertainty, as well as the error introduced when the model is used to approximate the true target distribution. The same considerations apply if the model feeds into some downstream analysis or decision-making process. Despite this, the literature on probabilistic generative neural network (NN) models rarely contains explicit verification of the learned distribution's statistics. Often, what is actually verified is that the generative model produces realistic outputs or has low reconstruction error in the data domain. Optimizing realism incidentally pseudo-optimizes the error in reconstructing data in a high-dimensional space from a low-dimensional latent representation, even if the model has not learned to encode and reconstruct features well. Prediction diversity and multimodality are usually only discussed qualitatively.

We examine several cycle-consistent architectures incorporating elements of popular generative NN models, namely generative adversarial networks (GANs) and variational autoencoders (VAEs), as well as deterministic encoders. Our focus on cycle-consistent architectures is motivated by the fact that GANs and VAEs do not contain a mechanism for reversing the generative transformation. By examining the learned maps between latent space (which has a simple prior sampled during generative inference) and feature space, we verify that even for simple problems, these architectures cannot model the true data distribution. This failure appears to be due to the models being neither invertible nor bijective, a state of affairs that persists even when the models are converted to deterministic maps. Our main contributions are:

- A null result, namely that even if generative models produce diverse and realistic predictions, they do not learn to bijectively map a latent distribution onto the true distribution represented by the training data. We use highly expressive models and follow best practices in network design and training; at minimum, our results argue that proper statistical behavior cannot be taken for granted. We imagine this will probably not surprise many ML practitioners, but also think it is worth making a point to test.

We have been unable to find any examples in the literature on GANs and VAEs that explicitly test for these properties, although some related concepts are well known, like the fact that gaps exist in a VAE's coverage in latent space.

- A simple extension of VAEs to deterministic latent vector sampling. Instead of using the reparameterization trick to sample from a multivariate normal distribution with learned mean and variance, we preserve the flow of information through the encoder-decoder stack and optimize KL divergence over *batches* of training examples.
- A proof that sampling from the space of solutions to a conditional inverse problem only requires pairs of conditioning information/ground truth examples, even when the conditioning information has high dimensionality (such as for super-resolution).

Our work does not come close to exploring all possible GAN- or VAE-based generative models, and it is *possible* that another architecture would learn a bijective map. We choose BicycleGAN as our starting point, as it is a state-of-the-art example of such models. Our chosen dataset, Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), is also very simple compared to standards like CIFAR-10 (Krizhevsky 2009) or ImageNet (Russakovsky et al. 2014). This is precisely our argument: if we cannot learn the statistics of even an "easy" dataset, given a reasonable choice of high-capacity model, we should assume *any* of this family of multi-modal generative models is statistically unreliable unless proven otherwise.

Related Work

Generative NNs come in a a variety of flavors. GANs and VAEs are the most studied; together with invertible neural networks (INNs), they encompass most methods that generate data from a latent space representation. Other methods exist that are based on sequential (Parmar et al. 2018) or Bayesian neural networks (Saatci and Wilson 2017).

A note on notation: in this paper, we use calligraphic letters, $\mathcal{X}$, for sets; upper-case letters, X, for random variables; (bold) lower-case letters, $\mathbf{x}$, for their (vector) values; and p for their probability densities.

Fundamentally, GANs, VAEs, and INNs operate similarly during the forward (generative) process: a generator $G : \mathbb{R}^m \rightarrow \mathbb{R}^n$ maps a vector $\mathbf{z} \in \mathcal{Z} \subseteq \mathbb{R}^m$ to a point in another space, $\mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n$. $\mathcal{Z}$ is the space of points sampled from some simple latent prior distribution, $\mathbf{z} \sim p_Z(\mathbf{z})$, and $\mathcal{X}$ is the complex space represented by the training data $p_X^{\text{data}}(\mathbf{x})$; for example, a collection of images belonging to some category. The true underlying distribution of the data, $p_X(\mathbf{x})$, is unknown.

For GANs and VAEs, $m \ll n$; the latent representation is compressed, with components of $\mathbf{z}$ corresponding to the presence or absence of major features in $\mathcal{X}$. Under certain circumstances, simple arithmetic operations on a vector $\mathbf{z}$ can add or subtract semantic features in the corresponding $\mathbf{x}$ (Radford, Metz, and Chintala 2016). For standard INNs, $m = n$, so there is no compression. Typically, the latent prior distribution is taken to be as simple as possible, for example $p_Z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$; this does not preclude a homeomorphic map between $\mathcal{Z}$ and $\mathcal{X}$, but may complicate it (Pérez Rey, Menkovski, and Portegies 2019).

The details, including the procedure for training G, vary between the different types of generative model. We briefly discuss the specific properties of GANs and VAEs that complicate inverse problem solving below, with comparison to INNs, which are explicitly invertible but less well-studied.

Generative Adversarial Networks (GANs)

A basic GAN training algorithm contains two models, a generator G that and a discriminator D (Goodfellow et al. 2014). D is a binary classifier trained to differentiate between real training data $\mathbf{x}_{\text{real}}$ and generated data $\mathbf{x}_{\text{gen}} = G(\mathbf{z})$, while G is trained to generate outputs that appear real to D. The two models are trained alternately, with the goal that D should eventually learn to reject any $\mathbf{x}_{\text{gen}} \notin \mathcal{X}$ and push $G(\mathbf{z}) \in \mathcal{X} \; \forall \; \mathbf{z} \in \mathcal{Z}$.

However, there is no guarantee that the distribution modeled by G, $p_X^G(\mathbf{x})$, is the same as or even close to the true distribution, $p_X(\mathbf{x})$. Mode collapse, where $G(\mathbf{z})$ outputs the same realistic $\mathbf{x}_{\text{gen}} \; \forall \; \mathbf{z}$, is only the most extreme example of this. A diverse, multi-modal $p_X^G(\mathbf{x})$ is clearly better than the delta function distribution modeled by a mode-collapsed GAN, but diversity is not the same as representativeness, and without a theoretical guarantee or explicit testing, G cannot be a trustworthy model of the true distribution of solutions to an inverse problem. In fact, GANs do not explicitly attempt to model probability densities at all; moreover, on its own, a GAN generator cannot be inverted to map some $\mathbf{x}$ into $\mathcal{Z}$.

Variational Autoencoders (VAEs)

A basic VAE also incorporates two models, an encoder E and a decoder G (Kingma and Welling 2013). $E(\mathbf{x})$ encodes $\mathbf{x}$ into $\mathcal{Z}$, and G maps the latent vector back into $\mathcal{X}$. During training, E and G are updated simultaneously to minimize (i) some measure of the distance between $\mathbf{x}$ and $G(E(\mathbf{x}))$, and (ii) the error introduced by approximating $p_Z^E(\mathbf{z})$, the latent distribution modeled by E, as $p_Z(\mathbf{z})$, the latent prior. The latter is expressed in terms of the KL divergence (Kullback and Leibler 1951) from $p_Z(\mathbf{z})$ to $p_Z^E(\mathbf{z})$. During inference, E is discarded, latent vectors are sampled from the prior via $\mathbf{z} \sim p_Z(\mathbf{z})$, and samples are generated via $\mathbf{x}_{\text{gen}} = G(\mathbf{z})$.

Enforcing $p_Z^E(\mathbf{z}) \approx p_Z(\mathbf{z})$ is critical, as inputs to G are drawn from the former during training, but the latter during inference. This is typically done by assuming that $p_Z^E(\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the mean and variance of $E(\mathbf{x})$, respectively, and $p_Z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$. (Other latent priors are possible, but the standard normal is most common.) KL divergence is a simple function of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in this case (Kingma and Welling 2013).

However, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are statistical measures, meaning that they are defined in terms of a large number of observations, and a particular $\mathbf{x}$ represents only a single observation. Thus, rather than learning a deterministic encoding $E(\mathbf{x}) = \mathbf{z}_{\text{enc}}$, $E(\mathbf{x})$ predicts two vectors, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, that define a point cloud in latent space. Monte Carlo sampling of that point cloud is

performed via the reparameterization trick, $\mathbf{z}_{\text{enc}} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are themselves deterministic, making it easier to take the gradient of E's weights with respect to its outputs via backpropagation, but this variational approximation adds irreducible noise to the generative process and means E cannot invert G. Therefore, while a VAE can help G map every point in $\mathcal{Z}$ to a realistic $\mathbf{x}$, the map is not bijective, which makes it difficult to verify its statistical properties or perform analysis in the latent space.

Invertible Neural Networks (INNs)

INNs (Ardizzone et al. 2018) are composed of a stack of operations that are invertible by construction, meaning that the entire network can be inverted cheaply. Thus, INNs learn a bijective map between $\mathcal{Z}$ and $\mathcal{X}$ and E is simply G^{-1}. This means that solving the inverse problem is, in principle, as easy as inverting an INN that has been trained on the forward problem; bidirectional training is possible as well. Notably, these maps also have a tractable Jacobian, which means that the unknown data distribution can be written explicitly in terms of the latent prior. These properties address some of the shortcomings of other types of generative models.

The main disadvantage of INNs seems to be that they are a recent development, and consequently have not been refined to the same extent as GANs and VAEs. Also, as a consequence of guaranteeing bijectivity, $\mathcal{Z}$ has the same dimensionality as $\mathcal{X}$, meaning further processing of the latent space is necessary to efficiently represent the data and perform certain types of analysis such as feature extraction, feature arithmetic, and anomaly detection.

Conditional Generative Models

All of these types have been extended to the conditional case, when the generator outputs are conditioned on some partial information, such as class label. cGANs (Mirza and Osindero 2014) and cVAEs (Sohn, Lee, and Yan 2015) have been heavily studied since they were introduced several years ago; cINNs (Liu et al. 2019; Ardizzone et al. 2019) are relatively new. Regardless, the basic idea is the same: G has a second, conditioning input $\mathbf{y} \in \mathcal{Y} \subseteq \mathbb{R}^l$, where $\mathcal{Y}$ is the space of conditioning data, and implicitly represents the conditional distribution $p^G_{X|Y=\mathbf{y}}(\mathbf{x})$, which may or may not be verifiably close to the true conditional distribution, $p_{X|Y=\mathbf{y}}(\mathbf{x})$. In the case at hand, our training data includes a set of conditional information, $p^{\text{data}}_Y(\mathbf{y})$, where each $\mathbf{y}$ corresponds to an $\mathbf{x}$ in $p^{\text{data}}_X(\mathbf{x})$. Together, these $(\mathbf{x}, \mathbf{y})$ pairs represent samples from the joint distribution $p^{\text{data}}_{XY}(\mathbf{x}, \mathbf{y})$.

Multimodal Models

Conditional GANs have been incredibly successful in mapping from one complex data space to another, but not in learning to predict distributions of solutions to ill-posed problems. The pix2pix framework (Isola et al. 2017) produced a model $G : \mathcal{X}' \to \mathcal{X}$ that mapped, deterministically and in one direction, from a point $\mathbf{x}' \in \mathcal{X}'$ to a point $\mathbf{x} \in \mathcal{X}$. CycleGAN (Zhu et al. 2017a) extended this to include another model $F : \mathcal{X} \to \mathcal{X}'$, by including a cycle consistency loss to encourage F and G to invert one another. Neither was able to incorporate stochasticity via random sampling of $\mathbf{z}$; even when $\mathbf{z}$ was included as a second input in pix2pix, the model simply learned to ignore it, although some stochastic elements could be introduced by including random dropout in the model. Further, CycleGAN owed its success in part to the fact that the cycle consistency loss function simultaneously optimized F and G, leading them to cheat by encoding hidden information in their predictions (Chu, Zhmoginov, and Sandler 2017).

BicycleGAN (Zhu et al. 2017b) attempted to rectify these shortcomings by combining components from GANs and VAEs. As it was the inspiration for this work, BicycleGAN is discussed in more detail below.

Although they are not our focus, we also note that probabilistic models like Bayesian neural networks are inherently more suitable for modeling multi-modality, though less so for learning bijective maps or latent space representations.

Principled Two-cycle-consistent Generative Models for Inverse Problems

We chose the BicycleGAN framework (Zhu et al. 2017b) as a starting point. Because several of our design choices are intended to address specific concerns we have with this framework, we briefly repeat it here. We then describe our modified framework and justify it as a principled attempt to build a two-cycle-consistent generative model, introducing a new method for constraining the distribution output of a VAE-like encoder without losing cycle-critical information during a reparameterization step.

BicycleGAN

BicycleGAN simultaneously trains a conditional generator $G : \mathcal{Y}, \mathcal{Z} \to \mathcal{X}$; a VAE-based encoder $E : \mathcal{X} \to \mathcal{Z}$ that outputs two vectors, the mean and log variance of a point cloud in $\mathcal{Z}$; and a discriminator D to enforce realism in the outputs of G. BicycleGAN is built around two cycles: a conditional latent regressor (cLR) to enforce consistency on the path $\mathcal{Z} \to \mathcal{X} \to \mathcal{Z}$, and a conditional variational autoencoder (cVAE) to do the same for the path $\mathcal{X} \to \mathcal{Z} \to \mathcal{X}$. The models are trained to jointly optimize multiple loss functions, described below.

Standard cGAN loss This is the original conditional GAN loss function defined in (Mirza and Osindero 2014):

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{\mathbf{x} \sim p^{\text{data}}_X}[\log(D(\mathbf{x}))] \\ + \mathbb{E}_{\substack{\mathbf{y} \sim p^{\text{data}}_Y \\ \mathbf{z} \sim p_Z}}[\log(1 - D(G(\mathbf{y}, \mathbf{z})))],\end{aligned} \tag{1}$$

where $\mathbb{E}_p[\cdots]$ is the expected value under a distribution p. The two terms evaluate the realism of real and generated data, respectively.

cVAE-GAN loss Identical to the GAN loss, except that $\mathbf{z}$ is sampled from $E(\mathbf{x})$ via the reparameterization trick, rather than from $p_Z(\mathbf{z})$:

$$\begin{aligned}\mathcal{L}^{\text{VAE}}_{\text{GAN}}(G, E, D) = \mathbb{E}_{\mathbf{x} \sim p^{\text{data}}_X}[\log(D(\mathbf{x}))] \\ + \mathbb{E}_{\substack{\mathbf{x}, \mathbf{y} \sim p^{\text{data}}_{XY} \\ \mathbf{z} \sim (\boldsymbol{\mu} + \boldsymbol{\epsilon} \odot \boldsymbol{\sigma})|_{E(\mathbf{x})}}}[\log(1 - D(G(\mathbf{y}, \mathbf{z})))].\end{aligned} \tag{2}$$

As $\boldsymbol{\mu} \to \mathbf{0}$ and $\boldsymbol{\sigma} \to \mathbf{1}$, the cVAE-GAN loss term approaches the standard cGAN loss term.

Latent space reconstruction loss The L^1 distance between a randomly sampled latent vector and its reconstruction after passing through both G and E:

$$\mathcal{L}_1^Z(G, E) = \mathbb{E}_{\substack{\mathbf{y}\sim p_Y^{\text{data}} \\ \mathbf{z}\sim p_Z}}[\|\mathbf{z} - \boldsymbol{\mu}|_{E(G(\mathbf{y},\mathbf{z}))}\|_1]. \quad (3)$$

This is the cLR cyclic consistency term, intended to teach E to invert G.

Ground truth reconstruction loss The L^1 distance between a ground truth example and its reconstruction after passing through both E and G:

$$\mathcal{L}_1^X(G, E) = \mathbb{E}_{\substack{\mathbf{x},\mathbf{y}\sim p_{XY}^{\text{data}} \\ \mathbf{z}\sim (\boldsymbol{\mu}+\boldsymbol{\epsilon}\odot\boldsymbol{\sigma})|_{E(\mathbf{x})}}}[\|\mathbf{x} - G(\mathbf{y}, \mathbf{z})\|_1]. \quad (4)$$

This is the cVAE cyclic consistency term, intended to teach G to invert E.

KL Divergence from $p_Z(\mathbf{z})$ to $p_Z^E(\mathbf{z})$ Attempts to ensure that E maps into the simple prior distribution sampled from during inference:

$$\mathcal{L}_{\text{KL}}(E) = \mathbb{E}_{\mathbf{x}\sim p_X^{\text{data}}}[D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)\big|_{E(\mathbf{x})} \|\mathcal{N}(\mathbf{0}, \mathbf{1}))]. \quad (5)$$

This is necessary to ensure that G always receives latent vectors belonging to the same distribution. By design, E's outputs are always interpreted as point clouds, since they only go back into the training process via reparameterization (which is always Gaussian), justifying the above form of the KL divergence.

The models are trained according to combinations of these loss terms, namely

$$G^* = \arg\min_G [\mathcal{L}_{\text{GAN}} + \mathcal{L}_{\text{GAN}}^{\text{VAE}} + \lambda_1^X \mathcal{L}_1^X + \lambda_1^Z \mathcal{L}_1^Z], \quad (6)$$

$$E^* = \arg\min_E [\mathcal{L}_{\text{GAN}}^{\text{VAE}} + \lambda_1^X \mathcal{L}_1^X + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}}], \quad (7)$$

$$D^* = \arg\max_D [\mathcal{L}_{\text{GAN}} + \mathcal{L}_{\text{GAN}}^{\text{VAE}}], \quad (8)$$

where "*" represents an updated model after one training step and the λs are weights.

BicycleGAN is very successful at generating diverse and realistic outputs $G(\mathbf{y}, \mathbf{z})$, but the statistics of the learned distributions have not been verified. This means ensuring that the conditional probability distribution implicitly modeled by G, $p_{X|Y=\mathbf{y}}^G(\mathbf{x})$, well approximates the distribution described by the training data, $p_{X|Y=\mathbf{y}}^{\text{data}}(\mathbf{x})$; in the Appendix, we show that this is possible even if we can only sample pairs from the *joint* distribution, $p_{XY}^{\text{data}}(\mathbf{x}, \mathbf{y})$. It also means ensuring that the latent distribution modeled by E, $p_Z^E(\mathbf{z})$, resembles the prior, $p_Z(\mathbf{z})$. Our tests of BicycleGAN do produce diverse and realistic reconstructions, but we do not find that the learned distribution match the ground truth statistics.

The original BicycleGAN has several features that potentially complicate learning a bijective map via enforcing two-cycle consistency:

1. G has two inputs, $\mathbf{y}$ and $\mathbf{z}$, but E only has one input, $\mathbf{x}$. This asymmetry means that E cannot invert G, since $E(G(\mathbf{y}, \mathbf{z}))$ has no way to disentangle the separate contributions of $\mathbf{y}$ and $\mathbf{z}$ in $G(\mathbf{y}, \mathbf{z})$.
2. E is trained using VAE-based methods and has two outputs, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, rather than a point in $\mathcal{Z}$. This is a second reason why E cannot actually invert $G(\mathbf{y}, \mathbf{z})$ to recover $\mathbf{z}$, and therefore cannot be trained to enforce cycle consistency in the latent space. $\mathcal{L}_1^Z$ actually attempts to minimize the distance between $\mathbf{z}$ and $\boldsymbol{\mu}|_{E(G(\mathbf{y},\mathbf{z}))}$.
3. The authors found that simultaneously training G and E on both cyclic consistency loss terms incentivizes cheating, similar to what was observed in CycleGAN (Chu, Zhmoginov, and Sandler 2017), so E is not trained to optimize $\mathcal{L}_1^Z$. However, when we attempt to replicate their approach we find indications of the same behavior when simultaneously training G and E to optimize $\mathcal{L}_1^X$.

We address these concerns primarily by changing E, and by changing the loss functions used in training—specifically, which loss functions are used to train which models. Our modified framework simultaneously trains a conditional generator $G : \mathcal{Y}, \mathcal{Z} \to \mathcal{X}$ and a *deterministic, conditional* encoder $E : \mathcal{Y}, \mathcal{X} \to \mathcal{Z}$. Each change is discussed in detail in the following sections.

Adversarial Losses

As with the original BicycleGAN, we use two cGAN-based loss terms to encourage realism in the outputs of G, one in which $\mathbf{z}$ is sampled from the latent prior and one in which $\mathbf{z}$ is encoded from an $(\mathbf{x}, \mathbf{y})$ pair. Rather than a discriminator, we use a Wasserstein critic C (Arjovsky, Chintala, and Bottou 2017), with a gradient penalty loss (Gulrajani et al. 2017). A discriminator is a binary classifier that can only return values of 0 (generated) or 1 (real), but a Wasserstein critic scores realism on a continuous scale of more negative (more likely to be generated) to more positive (more likely to be real). This provides more useful gradients to G during training, but is not a fundamental change to the framework.

The two loss terms used to train G are

$$\mathcal{L}_{\text{critic}}^{\text{cLR}}(G, C) = -\mathbb{E}_{\substack{\mathbf{y}\sim p_Y^{\text{data}} \\ \mathbf{z}\sim p_Z}}[1 \cdot C(\mathbf{x}_{\text{gen}})], \quad (9)$$

$$\mathcal{L}_{\text{critic}}^{\text{cAE}}(G, E, C) = -\mathbb{E}_{\mathbf{x},\mathbf{y}\sim p_{XY}^{\text{data}}}[1 \cdot C(\mathbf{x}_{\text{cyc}})], \quad (10)$$

where $\mathbf{x}_{\text{gen}} = G(\mathbf{y}, \mathbf{z})$ and $\mathbf{x}_{\text{cyc}} = G(\mathbf{y}, E(\mathbf{y}, \mathbf{x}))$. The explicit "1" indicates that G is being optimized to generate outputs considered "real" by C.

The complement to Eqs. 9 and 10 is

$$\mathcal{L}_{\text{critic}}^{\text{real}}(C) = -\mathbb{E}_{\mathbf{x}\sim p_X^{\text{data}}}[(1 \cdot C(\mathbf{x}))]. \quad (11)$$

Gradient penalty terms ensure that C is 1-Lipschitz:

$$\mathcal{L}_{\text{GP}}^{\text{cLR}}(G, C) = \mathbb{E}_{\substack{\mathbf{x},\mathbf{y}\sim p_Y^{\text{data}} \\ \mathbf{z}\sim p_Z}}[(\|\nabla_{\bar{\mathbf{x}}} C(\bar{\mathbf{x}}_{\text{gen}})\|_2 - 1)^2], \quad (12)$$

$$\mathcal{L}_{\text{GP}}^{\text{cAE}}(G, E, C) = \mathbb{E}_{\substack{\mathbf{x},\mathbf{y}\sim p_Y^{\text{data}} \\ \mathbf{z}\sim p_Z}}[(\|\nabla_{\bar{\mathbf{x}}} C(\bar{\mathbf{x}}_{\text{cyc}})\|_2 - 1)^2], \quad (13)$$

where gradients are taken with respect to randomly weighted averages $\bar{\mathbf{x}}_{\text{gen}} = u\mathbf{x} + (1-u)\mathbf{x}_{\text{gen}}$ and $\bar{\mathbf{x}}_{\text{cyc}} = u\mathbf{x} + (1-u)\mathbf{x}_{\text{cyc}}$, where u is uniform random noise on the interval $[0, 1]$. $\|\cdots\|_2$ is the L^2 norm.

Cycle Consistency Losses

As with BicycleGAN, we use two cycle consistency loss terms, one for the cLR path and one for the cAE path:

$$\mathcal{L}_1^Z(G,E) = \|\mathbf{z} - \mathbf{z}_{\text{cyc}}\|_1, \tag{14}$$

$$\mathcal{L}_1^X(G,E) = \|\mathbf{x} - \mathbf{x}_{\text{cyc}}\|_1, \tag{15}$$

where $\mathbf{z}_{\text{cyc}} = E(\mathbf{y}, G(\mathbf{y}, \mathbf{z}))$ and $\mathbf{x}_{\text{cyc}} = G(\mathbf{y}, E(\mathbf{y}, \mathbf{x}))$.

Calculating KL Divergence for a Deterministic Autoencoder

The two-output, probabilistic design of the encoder in a VAE enables calculation of the KL divergence from $p_Z(\mathbf{z})$ to $p_Z^E(\mathbf{z})$, an inherently statistical measure, from a single data point. However, this comes at the cost of cycle consistency in latent space, since E can no longer produce a single latent vector from $G(\mathbf{y}, \mathbf{z})$ to compare with the original $\mathbf{z}$. This is even worse for a cVAE: in order to reconstruct data from multiple classes, a non-conditioned VAE is forced to partition latent space into regions corresponding to the classes, which is in direct tension with the KL divergence loss's drive to map every $\mathbf{x}$ to $\mathcal{N}(\mathbf{0}, \mathbf{1})$. The extra information provided by the conditioning input negates the need to partition $\mathcal{Z}$, but that in turn means that $\boldsymbol{\mu}|_{E(\mathbf{x})} \to \mathbf{0}\ \forall\ \mathbf{x}$, and therefore that the reconstruction loss term defined in Eq. 3 will not be able to learn anything meaningful. Our initial attempts to train a BicycleGAN had precisely this problem, regardless of the relative weights assigned to the different loss terms.

A deterministic autoencoder preserves the flow of information through the cLR path, but prevents us from calculating the KL divergence on a per-example basis in the cAE path. Fortunately, because KL divergence is a statistical term, it can be calculated over a batch of training data. We therefore switch to a *batch-wise* KL divergence loss,

$$\mathcal{L}_{\text{KL}}^{\text{cAE}}(E) = \mathbb{E}_{\mathbf{x}\sim p_X^{\text{data}}}[D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}_{\text{enc}}, \boldsymbol{\sigma}_{\text{enc}}^2)\|\mathcal{N}(\mathbf{0}, \mathbf{1}))], \tag{16}$$

where E is now a deterministic autoencoder and $\boldsymbol{\mu}_{\text{enc}}$ and $\boldsymbol{\sigma}_{\text{enc}}$ are the mean and standard deviation of $\mathbf{z}_{\text{enc}} = E(\mathbf{y}, \mathbf{x})$, respectively, calculated over a batch of training data.

To make the framework symmetrical, we also include a similar loss term on the cLR path,

$$\mathcal{L}_{\text{KL}}^{\text{cLR}}(E) = \mathbb{E}_{\substack{\mathbf{y}\sim p_Y^{\text{data}} \\ \mathbf{z}\sim p_Z}}[D_{\text{KL}}(\mathcal{N}(\boldsymbol{\mu}_{\text{cyc}}, \boldsymbol{\sigma}_{\text{cyc}}^2)\|\mathcal{N}(\mathbf{0}, \mathbf{1}))], \tag{17}$$

where $\boldsymbol{\mu}_{\text{cyc}}$ and $\boldsymbol{\sigma}_{\text{cyc}}$ are calculated over a batch of $\mathbf{z}_{\text{cyc}} = E(\mathbf{y}, G(\mathbf{y}, \mathbf{z}))$.

These loss terms are unusual in that they are calculated once over the batch, instead of once for each example, followed by averaging over the batch. Their effectiveness is strongly dependent on batch size: for a batch of $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$, the batch size must exceed 100 for the calculated KL divergence to drop below 0.01. For small batch sizes, these terms do not provide much useful information.

The desired effect of these loss terms is that $E(\mathbf{y}, \mathbf{x})$ will learn to map the information in $\mathbf{x}$ that is not contained in $\mathbf{y}$ onto $\mathcal{Z}$, the space defined by the distribution $p_Z(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{1})$. On their own, $\mathcal{L}_{\text{KL}}^{\text{cLR}}$ and $\mathcal{L}_{\text{KL}}^{\text{cAE}}$ are not sufficient to guarantee this. For example, perhaps E could learn to assign a separate region in $\mathcal{Z}$ to each class that, averaged over a large batch, yields $\boldsymbol{\mu} = 0$, $\boldsymbol{\sigma} = 1$. However, cLR cycle consistency requires that $E(\mathbf{y}, G(\mathbf{y}, \mathbf{z})) \to \mathbf{z}\ \forall\ \mathbf{z} \sim p_Z(\mathbf{z})$. Therefore, optimizing cLR cycle consistency together with KL divergence requires that $E(\mathbf{y}, \mathbf{x})$ maps $(\mathbf{x}, \mathbf{y})$ pairs into $\mathcal{Z}$ independent of $\mathbf{y}$; or, in other words, that E learns common semantic features present in $\mathcal{X}$ but not $\mathcal{Y}$, and maps those features into $\mathcal{Z}$.

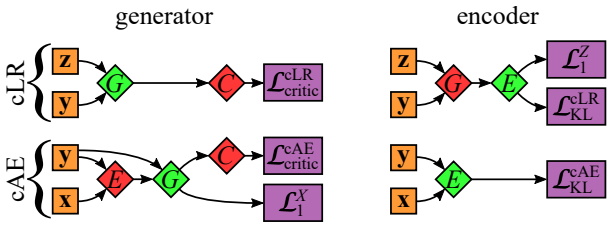

Figure 1: Training scheme for G and E. Orange indicates input. Red models are fixed, while green models are updated according to gradients obtained from the loss functions, which are purple. C is the critic.

Full Model

The models are trained according to

$$G^* = \arg\min_G [\mathcal{L}_{\text{critic}}^{\text{cLR}} + \mathcal{L}_{\text{critic}}^{\text{cAE}} + \lambda_1^X \mathcal{L}_1^X], \tag{18}$$

$$E^* = \arg\min_E [\mathcal{L}_{\text{KL}}^{\text{cLR}} + \mathcal{L}_{\text{KL}}^{\text{cAE}} + \lambda_1^Z \mathcal{L}_1^Z], \tag{19}$$

$$C^* = \arg\max_C [\mathcal{L}_{\text{critic}}^{\text{real}} - (\mathcal{L}_{\text{critic}}^{\text{cLR}} + \mathcal{L}_{\text{critic}}^{\text{cAE}})/2 + \lambda_{\text{GP}}(\mathcal{L}_{\text{GP}}^{\text{cLR}} + \mathcal{L}_{\text{GP}}^{\text{cAE}})]. \tag{20}$$

G attempts to generate realistic outputs, regardless of whether its $\mathbf{z}$ input comes from the prior distribution or E, while also attempting to invert E. E attempts to generate latent outputs consistent with the latent prior distribution, regardless of whether its $\mathbf{x}$ inputs come from the training data or G, while also attempting to invert G. C attempts to learn to differentiate between ground truth and generated $\mathbf{x}$ from both the cLR and cAE paths. The training scheme for G and E is shown in Fig. 1. The training scheme for C is standard for a Wasserstein critic except that there are two paths for generated samples; the factor of $1/2$ in Eq. 20 ensures that real and generated $\mathbf{x}$ are weighted equally so C does not just label everything as fake.

This formulation is symmetric. G is only trained to optimize cAE cycle consistency and the adversarial loss, a measure of realism or the (modeled) likelihood that $G(\mathbf{y}, \dots) \in \mathcal{X}$. Similarly, E is only trained to optimize cLR cycle consistency and the KL divergence loss, a measure of the likelihood that $E(\mathbf{y}, \dots) \in \mathcal{Z}$. The lack of common loss functions between G and E keeps them from learning to cheat.

We note that there is some redundancy between Eqs. 10 and 15, and between Eqs. 14 and 17. If E and G do truly learn to invert one another, as incentivized by Eqs. 14 and 15, then Eqs. 10 and 17 will no longer provide useful gradients, but at worst this might lead to some wasted computations late in training.

Methods

Our code is publicly available at https://github.com/USArmyResearchLab/ARL_Representativeness_of_Cyclic_GANs.

Dataset

We attempt to solve a simple super-resolution inverse problem. Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017) is a collection of 28×28 grayscale images separated into 10 classes of clothing and accessories. Each class has 6,000 training examples and 1,000 test examples. Subjectively, most examples in each class fall into a small number of clusters—for example, most images in the "trousers" category look very similar, while there are several different apparent "sub-categories" among T-shirts. Each class has some examples, less than 10% or so, that vary strongly from other examples in the same class, while 90% are very similar. Some classes (e.g., T-shirt and shirt) have significant overlap.

We downsample the images once, using 2×2 average pooling, to get a set of 14×14 images, $\mathbf{x}'$, and then again to get a corresponding set of 7×7 images, $\mathbf{y}'$. We then upscale $\mathbf{y}'$ by doubling the number of pixels to get very low-resolution 14×14 conditioning images $\mathbf{y}$, and use those to obtain the residuals $\mathbf{x} = \mathbf{x}' - \mathbf{y}$. We rescale $(\mathbf{x}, \mathbf{y})$ pairs to be on the interval $[-1, 1]$, with training and test data rescaled separately.

The original images in fashion-MNIST are already low-resolution, and this much downsampling destroys significant feature information. Many $\mathbf{x}$ can plausibly be obtained from a given $\mathbf{y}$, according to some distribution $p_{X|Y=\mathbf{y}}(\mathbf{x})$. Although the dataset only includes one $(\mathbf{x}, \mathbf{y})$ pair per example, rather than a distribution $p^{\text{data}}_{X|Y=\mathbf{y}}(\mathbf{x})$, we can still justify supervised training using $p^{\text{data}}_{XY}(\mathbf{x}, \mathbf{y})$. This is discussed in the Appendix.

Conditioning C vs. $\mathcal{X} \to \mathcal{Y}$ Consistency Loss

Depending on the problem, it may be inappropriate to condition C. For example, in image inpainting, $\mathbf{y}$ includes the mask that defines the region to be filled in, which can be exploited to identify perceptual discontinuities between the original and generated portions of the image (Pathak et al. 2016). For our problem, this is not the case.

A conditioning input allows C to ask whether $G(\mathbf{y}, \dots)$ is consistent with $\mathbf{y}$. We know the map $\mathcal{X} \to \mathcal{Y}$ in our case; for a given $\mathbf{x}$ to be perfectly consistent with $\mathbf{y}$, a 2×2-average-pool-downsampled $\mathbf{x}$ must be 0 everywhere. By applying this to $G(\mathbf{y}, \dots)$, we can separate this consistency from C as a pair of supplemental loss terms,

$$\mathcal{L}^{\text{cLR}}_{\mathcal{XY}}(G) = \lambda^{\text{cLR}}_{\mathcal{XY}} \|\text{downsample}(\mathbf{x}_{\text{gen}})\|_1, \quad (21)$$

$$\mathcal{L}^{\text{cAE}}_{\mathcal{XY}}(G, E) = \lambda^{\text{cAE}}_{\mathcal{XY}} \|\text{downsample}(\mathbf{x}_{\text{cyc}})\|_1. \quad (22)$$

We do not observe a significant difference in results between implementing a conditional $C(\mathbf{y}, \mathbf{x})$ vs. an unconditioned $C(\mathbf{x})$ plus these supplemental losses, in terms of visual quality. The supplemental losses enforce the desired consistency explicitly, so we use them in our experiments.

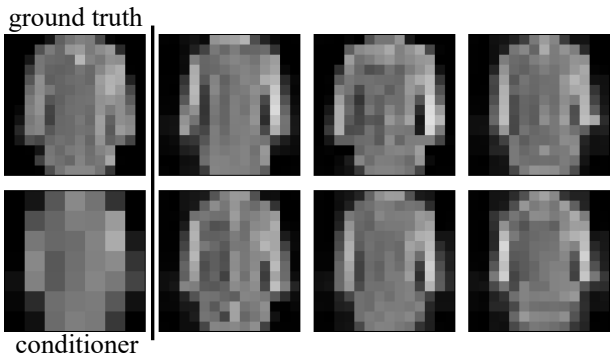

Figure 2: Six random samples of $G(\mathbf{y}, \mathbf{z})$, on the right, with the corresponding ground truth $\mathbf{x}$ and conditioner $\mathbf{y}$, on the left, for comparison. G was trained using Eqs. 18–20.

Implementation Details

We use deep ResNeXt networks with efficient grouped convolutions and identity shortcuts (Xie et al. 2016) and $\mathcal{O}(4 \times 10^6)$ parameters in each of G, E, and C. Activations are all LeakyReLU followed by layer normalization, except for the generator output, which uses $\tanh$. We inject latent vectors $\mathbf{z}$ only at the top layer of the model. We test $\dim(\mathbf{z}) \in \{10, 100, 1000\}$, finding that 100 performs best overall and therefore use that for most experiments.

We perform training in TensorFlow, using the Adam optimizer with default parameters. We choose a batch size of 200, since our batch-wise KL divergence is only useful for fairly large batch size. We train using instance noise (Sønderby et al. 2016) over $\mathbf{x}$, $\mathbf{x}_{\text{gen}}$, and $\mathbf{x}_{\text{cyc}}$, replacing $\mathbf{x}_{[\dots]} \leftarrow \alpha \mathbf{x}_{[\dots]} + (1 - \alpha)\mathbf{u}$, where α anneals from 0 to 1 in increments of 0.01 over the first 100 epochs of training, and $\mathbf{u}$ is uniform random noise on the interval $[-1, 1]$.

For every update of G and E, we train C continuously until its validation loss fails to improve for 5 consecutive batches to ensure it is approximately converged. Training continues until all models' validation losses have failed to improve for 20 consecutive epochs.

Results and Discussion

We test variational vs. deterministic encoders; different choices of hyperparameters and loss weights; and allowing vs. disallowing overlap between loss functions used to train E and G. The outcomes of these tests share two commonalities: G produces diverse and realistic predictions, exemplified in Fig. 2, but without cycle consistency or distribution matching in Z. Thus, their maps are not bijective and do not map the latent space onto the true data distribution.

Variational Frameworks

First, we test the BicycleGAN framework (the variational method) with a Wasserstein critic, trained with Eqs. 6 (modified to include Eqs. 9 and 10), 7, and 20. As noted previously, this generates realistic images on par with our other tests, but even strongly weighting $\mathcal{L}^Z_1$ does not produce any cLR cycle consistency, with $\boldsymbol{\mu}, \boldsymbol{\sigma}$ going to $\mathbf{0}, \mathbf{1}$ very rapidly.

This is true regardless of whether we train E but not G on $\mathcal{L}_1^Z$ and/or do not train E on $\mathcal{L}_1^X$.

Deterministic Frameworks

As mentioned previously, we observe some steganographic collaboration between models that train G and E with overlapping loss functions. For example, some information is hidden in the black background of each $\mathbf{x}_{\text{gen}}$ as an imperceptible, low-amplitude signal. Truncating the values of those pixels to -1 with no other changes produces a sizable change in $\|\mathbf{z} - \mathbf{z}_{\text{cyc}}\|_1$, increasing it from 0.07 to 0.4. This emphasizes that cycle consistency in the models does not mean they have learned a meaningful map. We therefore restrict our experiments to models trained without overlapping loss functions, using Eqs. 18–20.

In this scenario we observe no cLR cycle consistency, as E is unable to extract latent information from $\mathbf{x}_{\text{gen}}$. We see this regardless of the relative weights in Eq. 19, even if we weight $\mathcal{L}_{\text{KL}}^{\text{cLR}}$ and $\mathcal{L}_{\text{KL}}^{\text{cAE}}$ independently. A typical result is shown in Fig. 3a: the model is penalized by $\mathcal{L}_1^Z$ for wrongly predicting $\mathbf{z}_{\text{cyc}}$, and is also unable to find a path to learn to predict $\mathbf{z}_{\text{cyc}}$ correctly. Accordingly, given some $\mathbf{z}$ (red line) it simply predicts $\mathbf{z}_{\text{cyc}} \approx \mathbf{0} \; \forall \; \mathbf{y}$ (black lines), which results in a smaller penalty than if it had predicted a nonzero, incorrect $\mathbf{z}_{\text{cyc}}$. This is a failure to optimize both $\mathcal{L}_1^Z$, because the red and black lines do not overlap, and $\mathcal{L}_{\text{KL}}^{\text{cLR}}$, because the standard deviation of the elements in $\mathbf{z}_{\text{cyc}}$ is much less than 1. Only when we set $\lambda_1^Z \approx 0$ are the KL-divergence loss terms able to enforce good statistics.

This is not because E is simply unable to learn to invert G. Rather, it appears to be unable to do so quickly and robustly. Once the model converges, we train E only for 1000 more epochs, holding G and C constant. This does slightly improve cycle consistency in $\mathbf{z}$, as shown in Fig. 3b. $\mathcal{L}_1^Z$ takes "only" several hundred epochs to plateau, so training time is not the only limiting factor. The small overall improvement in $\mathcal{L}_1^Z$ (about 4%) belies a noticeable qualitative change, due to a minority of $\mathbf{z}$-coordinates being very wrong. This still falls far short of allowing E to truly invert G, and further, it is not stable, disappearing with another update to G.

$\mathcal{L}_1^X$ has limited success in optimizing cAE cycle consistency. This happens to some extent even if we do not optimize on $\mathcal{L}_1^Z$, reflecting the fact that optimizing realism pseudo-optimizes the L^1 distance between $\mathbf{x}$ and $\mathbf{x}_{\text{cyc}}$. $\mathcal{L}_1^X$ then depends on only a handful of pixels for most images, which dominate the expected value in Eq. 15. In support of this, we find that $\mathbf{x}_{\text{cyc}}$ does not reproduce rare features such as text, symbols, and some patterns and orientations.

Conclusion

Our experiments are consistent with the idea that cycle-consistent GANs are not good vehicles for obtaining representative maps, largely because they do not learn cycle consistency well in the first place. They still produce diverse and realistic outputs, but are not representative, in effect mapping onto an unknown subset of $\mathcal{X}$.

Two-cycle consistency is a surrogate for bijectivity. The fact that it is so difficult to train G and E to invert one another even for a simple problem such as the one we test indicates that an explicit guarantee of bijectivity is probably the best path forward for achieving this, which in turn will allow inverse problems to be solved rigorously and probabilistically via simple Monte Carlo sampling. In the absence of such a guarantee, we believe representativeness in the generated distribution must be explicitly tested for in generative models, especially when risk or bias assessment, uncertainty quantification, and similar considerations are important.

INNs are inherently bijective and do not require explicit enforcement of two-cycle consistency. It seems likely that, even if there is a way to enforce bijectivity in a GAN-like construct, INNs provide a simpler path to this result.

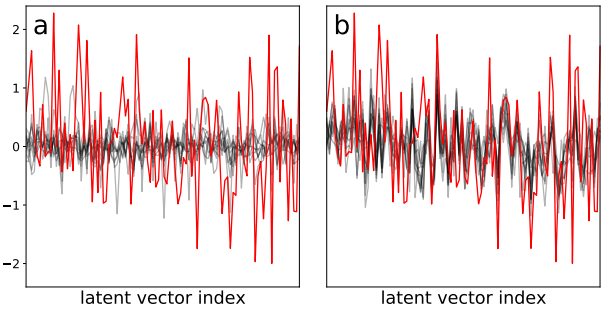

Figure 3: A plot of ten different $\mathbf{z}_{\text{cyc}}$ (black) obtained from one $\mathbf{z}$ (red), using one $\mathbf{y}$ from each of the ten classes. (a) E minimizes $\mathcal{L}_1^Z$ by setting $\mathbf{z}_{\text{cyc}} \approx \mathbf{0}$, because it cannot learn cycle consistency during normal training. (b) E does develop some fragile cycle consistency after additional training for 10^3 epochs holding G fixed.

Acknowledgments

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Combat Capabilities Development Command (DEVCOM) Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. The authors thank Dr. Ting Wang for valuable discussions and Mr. Matt Ziemann for careful reading of the manuscript. Computer time was provided by the DEVCOM ARL DSRC.

Appendix: Sampling Inverse Problem Solutions

G and E implicitly define distributions $p_{X|Y=\mathbf{y}}^G(\mathbf{x})$ and $p_Z^E(\mathbf{z})$, respectively:

$$\int_{\mathbb{R}^m} p_Z(\mathbf{z}) \; \mathrm{d}\mathbf{z} \cdot f(G(\mathbf{y}, \mathbf{z})) = \int_{\mathbb{R}^n} p_{X|Y=\mathbf{y}}^G(\mathbf{x}) \; \mathrm{d}\mathbf{x} \cdot f(\mathbf{x}), \tag{A.1}$$

$$\int_{\mathbb{R}^n} p_{X|Y=\mathbf{y}}(\mathbf{x}) \; \mathrm{d}\mathbf{x} \cdot f(E(\mathbf{y}, \mathbf{x})) = \int_{\mathbb{R}^m} p_Z^E(\mathbf{z}) \; \mathrm{d}\mathbf{z} \cdot f(\mathbf{z}) \tag{A.2}$$

or, equivalently,

$$\mathbb{E}_{\mathbf{z}\sim p_Z}[f(G(\mathbf{y},\mathbf{z}))] = \mathbb{E}_{\mathbf{x}\sim p^G_{X|Y=\mathbf{y}}}[f(\mathbf{x})], \tag{A.3}$$

$$\mathbb{E}_{\mathbf{x}\sim p_{X|Y=\mathbf{y}}}[f(E(\mathbf{y},\mathbf{x}))] = \mathbb{E}_{\mathbf{z}\sim p^E_Z}[f(\mathbf{z})]. \tag{A.4}$$

If our model performs as desired, the distribution of points in $\mathbb{R}^n$ obtained by sampling $\mathbf{z} \sim p_Z(\mathbf{z})$ and evaluating $G(\mathbf{y},\mathbf{z})$ should resemble the true conditional distribution of possible $\mathbf{x}$ consistent with a given $\mathbf{y}$, $p_{X|Y=\mathbf{y}}(\mathbf{x})$. Similarly, the distribution of points in $\mathbb{R}^m$ obtained by sampling $\mathbf{x} \sim p_{X|Y=\mathbf{y}}(\mathbf{x})$ and evaluating $E(\mathbf{y},\mathbf{x})$ should resemble the latent prior, $p_Z(\mathbf{z})$. However, while we *can* sample $\mathbf{z}$ from the latent prior, we *cannot* sample $\mathbf{x}$ from the true conditional distribution of X, since we usually have only one pair each of ground truth $(\mathbf{y},\mathbf{x})$ in our training data.

Fortunately, we can get around this if the learned encoder distribution, $p^E_Z(\mathbf{z})$, is independent of Y, which is a reasonable assumption if the reconstruction process can identify common features applicable to many different conditioners, and is further supported by the KL divergence loss terms.

Taking the expectation under $p_Y(\mathbf{y})$ in Eq. A.2 gives:

$$\int_{\mathbb{R}^n}\int_{\mathbb{R}^l} p_{X|Y=\mathbf{y}}(\mathbf{x})\, p_Y(\mathbf{y})\ \mathrm{d}\mathbf{x}\ \mathrm{d}\mathbf{y}\cdot f(E(\mathbf{y},\mathbf{x})) = \int_{\mathbb{R}^m} p^E_Z(\mathbf{z})\ \mathrm{d}\mathbf{z}\cdot f(\mathbf{z}) \int_{\mathbb{R}^l} p_Y(\mathbf{y})\ \mathrm{d}\mathbf{y}. \tag{A.5}$$

The second integral on the RHS evaluates to one by definition, and Bayes's theorem lets us rewrite the LHS to get

$$\int_{\mathbb{R}^n}\int_{\mathbb{R}^l} p_{XY}(\mathbf{x},\mathbf{y})\ \mathrm{d}\mathbf{y}\ \mathrm{d}\mathbf{x}\cdot f(E(\mathbf{y},\mathbf{x})) = \int_{\mathbb{R}^m} p^E_Z(\mathbf{z})\ \mathrm{d}\mathbf{z}\cdot f(\mathbf{z}), \tag{A.6}$$

where $p_{XY}(\mathbf{x},\mathbf{y})$ is the true joint probability density function of X and Y. Unlike $p_{X|Y=\mathbf{y}}(\mathbf{x})$, we *can* sample from $p_{XY}(\mathbf{x},\mathbf{y})$; the training data, $p^{\text{data}}_{XY}(\mathbf{x},\mathbf{y})$, does just that. We can then enforce a distribution constraint on $p^E_Z(\mathbf{z})$ as usual.

References

Ardizzone, L.; Kruse, J.; Wirkert, S. J.; Rahner, D.; Pellegrini, E. W.; Klessen, R. S.; Maier-Hein, L.; Rother, C.; and Köthe, U. 2018. Analyzing inverse problems with invertible neural networks. *CoRR* abs/1808.04730.

Ardizzone, L.; Lüth, C.; Kruse, J.; Rother, C.; and Köthe, U. 2019. Guided image generation with conditional invertible neural networks. *CoRR* abs/1907.02392.

Arjovsky, M.; Chintala, S.; and Bottou, L. 2017. Wasserstein GAN. *CoRR* abs/1701.07875.

Chu, C.; Zhmoginov, A.; and Sandler, M. 2017. Cyclegan, a master of steganography. *CoRR* abs/1712.02950.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27*. Curran Associates, Inc. 2672–2680.

Gulrajani, I.; Ahmed, F.; Arjovsky, M.; Dumoulin, V.; and Courville, A. C. 2017. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, 5767–5777.

Hadamard, J. 1902. Sur les problèmes aux dérivées partielles et leur signification physique. *Princeton university bulletin* 49–52.

Isola, P.; Zhu, J.; Zhou, T.; and Efros, A. A. 2017. Image-to-image translation with conditional adversarial networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5967–5976.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report.

Kullback, S., and Leibler, R. A. 1951. On information and sufficiency. *Ann. Math. Statist.* 22(1):79–86.

Liu, R.; Liu, Y.; Gong, X.; Wang, X.; and Li, H. 2019. Conditional adversarial generative flow for controllable image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mirza, M., and Osindero, S. 2014. Conditional generative adversarial nets. *CoRR* abs/1411.1784.

Parmar, N.; Vaswani, A.; Uszkoreit, J.; Kaiser, L.; Shazeer, N.; and Ku, A. 2018. Image transformer. *CoRR* abs/1802.05751.

Pathak, D.; Krähenbühl, P.; Donahue, J.; Darrell, T.; and Efros, A. A. 2016. Context encoders: Feature learning by inpainting. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2536–2544.

Pérez Rey, L.; Menkovski, V.; and Portegies, J. 2019. Can vaes capture topological properties? NeurIPS 2019 Workshop ; Conference date: 13-12-2019 Through 13-12-2019.

Peterson, J.; Battleday, R.; Griffiths, T.; and Russakovsky, O. 2019. Human uncertainty makes classification more robust. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9616–9625.

Radford, A.; Metz, L.; and Chintala, S. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M. S.; Berg, A. C.; and Li, F. 2014. Imagenet large scale visual recognition challenge. *CoRR* abs/1409.0575.

Saatci, Y., and Wilson, A. G. 2017. Bayesian gan. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 3622–3631.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning structured output representation using deep conditional generative models. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 3483–3491.

Sønderby, C. K.; Caballero, J.; Theis, L.; Shi, W.; and Huszár, F. 2016. Amortised MAP inference for image super-resolution. *CoRR* abs/1610.04490.

Wiyatno, R. R.; Xu, A.; Dia, O.; and de Berker, A. 2019. Adversarial examples in modern machine learning: A review. *CoRR* abs/1911.05268.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR* abs/1708.07747.

Xie, S.; Girshick, R. B.; Dollár, P.; Tu, Z.; and He, K. 2016. Aggregated residual transformations for deep neural networks. *CoRR* abs/1611.05431.

Yang, W.; Zhang, X.; Tian, Y.; Wang, W.; Xue, J.; and Liao, Q. 2019. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia* 21(12):3106–3121.

Zhu, J.-Y.; Park, T.; Isola, P.; and Efros, A. A. 2017a. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, 2223–2232.

Zhu, J.-Y.; Zhang, R.; Pathak, D.; Darrell, T.; Efros, A. A.; Wang, O.; and Shechtman, E. 2017b. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, 465–476.