Inverting the Generator of a Generative Adversarial Network

Antonia Creswell and Anil Anthony Bharath

***Abstract*— Generative adversarial networks (GANs) learn a deep generative model that is able to synthesize novel, high-dimensional data samples. New data samples are synthesized by passing latent samples, drawn from a chosen prior distribution, through the generative model. Once trained, the latent space exhibits interesting properties that may be useful for downstream tasks such as classification or retrieval. Unfortunately, GANs do not offer an "inverse model," a mapping from data space back to latent space, making it difficult to infer a latent representation for a given data sample. In this paper, we introduce a technique, *inversion*, to project data samples, specifically images, to the latent space using a pretrained GAN. Using our proposed *inversion* technique, we are able to identify which attributes of a data set a trained GAN is able to model and quantify GAN performance, based on a reconstruction loss. We demonstrate how our proposed *inversion* technique may be used to quantitatively compare the performance of various GAN models trained on three image data sets. We provide codes for all of our experiments in the website (https://github.com/ToniCreswell/InvertingGAN).**

***Index Terms*— Backpropagation, feature extraction, image generation, multilayer neural network, pattern recognition, unsupervised learning.**

I. Introduction

GENERATIVE adversarial networks (GANs) [10], [20] are a class of generative model which are able to synthesize novel, realistic looking images of faces, digits, and street numbers [20]. GANs involve two networks: a generator, G, and a discriminator, D. The generator, G, is trained to generate synthetic images, taking a random vector, z, drawn from a prior distribution, $P(Z)$, as input. The prior is often chosen to be a normal or uniform distribution.

Radford *et al.* [20] demonstrated that GANs learn a "rich linear structure," meaning that algebraic operations in Z-space often lead to semantically meaningful synthetic samples in image space. Since images represented in Z-space are often meaningful, direct access to a $z \in Z$ for a given image, $x \in X$ may be useful for discriminative tasks such as retrieval or classification. Recently, it has also become desirable to be able to access Z-space in order to manipulate original images [27]. Thus, there are many reasons we may wish to invert the generator.

Typically, inversion is achieved by finding a vector $z \in Z$ which when passed through the generator produces an image that is very similar to the target image. If no suitable z exists, this may be an indicator that the generator is unable to model either the whole image or certain attributes of the image. We give a concrete example in Section VI-B. Therefore, inverting the generator, additionally, provides interesting insights to highlight what a trained GAN has learned.

Mapping an image, from image space, X, to Z-space is nontrivial, as it requires inversion of the generator, which is often many layered, nonlinear model [4], [10], [20]. Dumoulin *et al.* [9] (ALI) and Donahue *et al.* (BiGAN) [8] proposed learning a third, decoder network along side the generator and discriminator to map image samples back to Z-space. Collectively, they demonstrated results on MNIST, ImageNet, CIFAR-10, SVHN, and CelebA. However, reconstructions of inversions are often poor. Specifically, reconstructions of inverted MNIST digits using methods of Donahue *et al.* [7], often fail to preserve the style and character class. Recently, Li *et al.* [16] proposed a method to improve reconstructions. Some drawbacks to these approaches [8], [9], [16] include the need to train a third network, which increases the number of parameters that have to be learned; with more parameters, there is generally a greater chance of overfitting [23], or even of memorizing [12] input samples.

When employing a decoder model to perform inversion, its value as a diagnostic tool for evaluating GANs is hindered. GANs suffer from several pathologies [1], [2], [13], [19], [21], [26], including overfitting [11], [24], that we may be able to detect using inversion. If an additional encoder model is trained to perform inversion [8], [9], [16], [17], the encoder itself may overfit, thus not portraying the true nature of a trained GAN. Since our approach does not involve training an additional encoder model, we may use our approach for "trouble-shooting" and evaluating different pretrained GAN models.

In this paper, we make the following contributions.

1) We propose a novel approach to invert the generator of any pretrained GAN, provided that the computational graph for the generator network is available (Section II).
2) We demonstrate that, we are able to infer a Z-space representation for a target image, such that when passed through the GAN, it produces a sample visually similar to the target image (Section VI).
3) We demonstrate several ways in which our proposed inversion technique may be used to **both qualitatively** (Section VI-B) **and quantitatively** compare GAN models (Section VII).

Manuscript received February 15, 2018; revised August 14, 2018; accepted October 7, 2018. Date of publication November 2, 2018; date of current version June 14, 2019. The work of A. Creswell was supported by an EPSRC Doctoral Training Programme under Grant EP/L504786/1. *(Corresponding author: Antonia Creswell.)*

The authors are with BICV, Imperial College London, London SW7 2AZ, U.K. (e-mail: ac2211@ic.ac.uk; aab01@ic.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2875194

Algorithm 1: Algorithm for Inferring $z^* \in \Re^d$, the Latent Representation for an Image $x \in \Re^{m \times m}$

Result: Infer(x)
$z^* \sim P_z(Z)$;
while *NOT converged* **do**
  $L \leftarrow -(x \log[G(z^*)] + (1-x)\log[1-G(z^*)])$;
  $z^* \leftarrow z^* - \alpha \nabla_z L$;
end
return z^* ;

4) In addition, we show that batches of z samples can be inferred from batches of image samples, which improve the efficiency of the inversion process by allowing multiple images to be inverted in parallel (Section II-A).

We begin, by describing our proposed inversion technique.

II. Method: Inverting the Generator

For a target image, $x \in \Re^{m \times m}$ we want to infer the Z-space representation, $z \in Z$, which when passed through the trained generator produces an image very similar to x. We refer to the process of inferring z from x as *inversion*. This can be formulated as a minimization problem, as follows, where $\mathbb{E}$ is the expectation:

$$z^* = \min_z -\mathbb{E}_x \log[G(z)]. \tag{1}$$

Provided that the computational graph for $G(z)$ is known, z^* can be calculated via gradient descent methods, taking the gradient of G with respect to z. This is detailed in Algorithm 1.

Provided that the generator is deterministic, each z value maps to a single image, x. A single z value cannot map to multiple images. However, it is possible that a single x value may map to several z representations, particularly if the generator has collapsed [21]. This suggests that there may be multiple possible z values to describe a single image. This is very different to a discriminative model, where multiple images, may often be described by the same representation vector [18], particularly when a discriminative model learns representations tolerant to variations.

The approach described in Algorithm 1 is similar in spirit to that of Mahendran and Vedaldi [18], but instead of inverting a representation to obtain the image that was responsible for it, we infer the latent representation that generates a particular image.

A. Inverting a Batch of Samples

Algorithm 1 shows how we can invert a single data sample. However, it may not be efficient to invert single images at a time; instead, a more practical approach is to invert many images at once. We will now show that we are able to invert batches of examples.

Let $\mathbf{z}_b \in \Re^{B \times n}$, $\mathbf{z}_b = \{z_1, z_2, \ldots z_B\}$ be a batch of B samples of z. This will map to a batch of image samples $\mathbf{x}_b \in \Re^{B \times m \times m}$, $\mathbf{x}_b = \{x_1, x_2, \ldots x_B\}$. For each pair (z_i, x_i), $i \in \{1 \ldots B\}$, a loss L_i, may be calculated. The update for z_i would then be $z_i \leftarrow z_i - \alpha(dL_i/dz_i)$.

If reconstruction loss is calculated over a batch, then the batch reconstruction loss would be $\sum_{i=\{1,2\ldots B\}} L_i$, and the update would be

$$\nabla_{\mathbf{z}_b} L = \frac{\partial \sum_{i \in \{1,2,\ldots B\}} L_i}{\partial(\mathbf{z}_b)} \tag{2}$$

$$= \frac{\partial(L_1 + L_2 \ldots + L_i)}{\partial(\mathbf{z}_b)} \tag{3}$$

$$= \frac{dL_1}{dz_1}, \frac{dL_2}{dz_2}, \ldots \frac{dL_B}{dz_B}. \tag{4}$$

Each reconstruction loss depends only on $G(z_i)$, so L_i depends only on z_i, which means $(\partial L_i/\partial z_j) = 0$, for all $i \neq j$. This shows that z_i is updated only by reconstruction loss L_i, and the other losses do not contribute to the update of z_i, meaning that it is valid to perform updates on batches. The ability to perform updates on batches means that multiple inversions may be run in parallel, making use of parallel architectures, and specifically general purpose graphical processing units.

B. Using Prior Knowledge of P(Z)

A GAN is trained to generate samples from a $z \in Z$ where the distribution over Z is a chosen prior distribution, $P(Z)$. $P(Z)$ is often a multivariate Gaussian or uniform distribution. If $P(Z)$ is a multivariate uniform distribution, $\mathcal{U}[a, b]$, then after updating z^*, it can be clipped to be between $[a, b]$. This ensures that z^* lies in the probable regions of Z. If $P(Z)$ is a multivariate Gaussian Distribution, $\mathcal{N}[\mu, \sigma^2]$, regularization terms may be added to the cost function, penalizing samples that have statistics that are not consistent with $P(Z) = \mathcal{N}[\mu, \sigma^2]$.

If $z \in Z$ is a vector of length d and each of the d elements in $z \in \Re^d$ is drawn independently and from identical distributions, we may be able to add a regularization term to the loss function. For example, if $P(Z)$ is a multivariate Gaussian distribution, then elements in a single z are independent and identically drawn from a Gaussian distribution. Therefore, we may calculate the likelihood of an encoding, z, under a multivariate Gaussian distribution by evaluating

$$\log P(z) = \log P(z^1, \ldots, z^d) = \frac{1}{d} \sum_{i=0}^{d} \log \mathcal{P}(z^i)$$

where z^i is the ith element in a latent vector z and $\mathcal{P}$ is the probability density function of a (univariate) Gaussian, which may be calculated analytically. Our new loss function may be given by

$$L(z, x) = \mathbb{E}_x \log[G(z)] - \beta \log P(z) \tag{5}$$

by minimizing this loss function (Equation 5), we encourage z^* to come from the same distribution as the prior.

III. Relation to Previous Work

In this paper, we build on our own work [6], which was previously presented at the NIPS 2016 Workshop on Adversarial Training, but has not yet been published.

We have augmented the paper by performing additional experiments on a shoe data set [17] and CelebA, as well

as repeating experiments on the Omniglot data set using the DCGAN model proposed by Radford *et al.* [20] rather than our own network [5]; we also perform experiments showing the ability to access the Wasserstein GAN (WGAN). In addition to proposing a novel approach for mapping data samples to their corresponding latent representation, we show how our approach may be used to quantitatively and qualitatively compare models.

Our approach to inferring z from x is similar to the previous work of Zhu *et al.* [27], but we make additional contributions.

Zhu *et al.* [27] calculated a reconstruction loss by comparing the features of x and $G(z^*)$ extracted from the layers of AlexNet [14], a convolutional neural network (CNN) trained on natural scenes. This loss is unlikely to be appropriate if the generated samples are either not of natural scenes (e.g., Omniglot handwritten characters), or are of signals. Our approach considers a raw pixel loss, providing a generic approach that is not specific to the data set. Furthermore, if our intention is to use the inversion to better understand the GAN model, it is not appropriate to incorporate information from other pretrained networks (e.g., AlexNet) in the inversion process (see [3] for empirical evidence of bias even in visual networks).

An alternative class of inversion methods involves training a separate encoding network to learn a mapping from image samples, x to latent samples z. Li *et al.* [16], Donahue *et al.* [8], and Dumoulin *et al.* [9] propose learning the encoder along side the GAN. Training an additional encoder network increases the parameter space for learning: for two GANs of the same combined generator/discriminator capacity, the additional parameters of an encoder network on one pair can lead to overfitting in that pair. Furthermore, this approach may not be applied to pretrained models.

Luo *et al.* [17], train an encoding network after a GAN has been trained, which means that their approach may be applied to pretrained models. However, as with any learning approach, the trained encoder may overfit to the examples it has been trained on. For this reason, the approach of Luo *et al.* [17] may not be suitable for inverting image samples that come from a different distribution to the training data. Luo *et al.* [17] only show "original" *reconstructed* image samples being inverted, not samples from a set of independent data; in other words, Luo *et al.* [17] showed results for inverting already synthesized samples, rather than *real* image samples from a test set.

In contrast, we apply our inversion process on data samples drawn from test sets of real data samples. To make the inversion more challenging, we sometimes invert image samples that come from a different distribution to the training data. For example, we inverted image samples from the Omniglot handwritten characters data set that come from a different set of alphabets to the set used to train the (Omniglot) GAN. We were still able to recover a latent encoding that captures **most** features of the test data samples.

Finally, previous inversion approaches that use learned encoder models [8], [9], [16], [17] may not be suitable for "trouble-shooting," as symptoms of the GAN may be exaggerated by an encoder that overfits. We discuss this in more detail in Section VII.

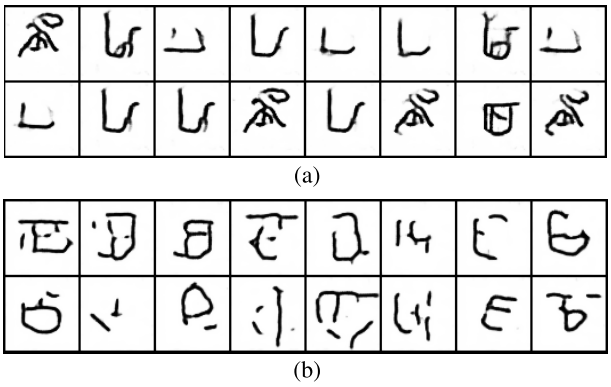

Fig. 1. *Synthetic Omniglot samples* represent samples synthesized using a (a) GAN and (b) WGAN.

IV. "PreTrained" Models

In this section, we discuss the training and architecture details of several different GAN models, trained on three different data sets, which we will use for our inversion experiments detailed in Section V. We show results on a total of 10 trained models (Sections VI and VII).

A. Omniglot

The Omniglot data set [15] consists of characters from 50 different alphabets, where each alphabet has at least 14 different characters. The Omniglot data set has a background data set, used for training and a test data set. The background set consists of characters from 30 writing systems, while the test data set consists of characters from the other 20. Note that characters in the training and testing data set come from different writing systems. We train both a DCGAN [20] and a WGAN [2] using a latent representation of dimension, $d = 100$. The WGAN [2] is a variant of the GAN that is easier to train and less likely to suffer from mode collapse; mode collapse is where synthesized samples look similar to each other. All GANs are trained with additive noise whose standard deviation decays during training [1]. Fig. 1 shows Omniglot samples synthesized using the trained models. Though it is clear from Fig. 1(a), that the GAN has collapsed, because the generator is synthesizing similar samples for different latent codes, it is less clear to what extent the WGAN [Fig. 1(b)] may have collapsed or overfit. It is also unclear from Fig. 1(b) what representative power, the (latent space of the) WGAN has. Results in Sections VI and VII will provide more insight into the representations learned by these models.

B. Shoes

The shoes data set [25] consists of 50 000 examples of shoes in RGB color, from four different categories and over 3000 different subcategories. The images are of dimensions 128×128. We leave 1000 samples out for testing and use the rest for training. We train two GANs using the DCGAN [20] architecture. We train one DCGAN with full sized images and

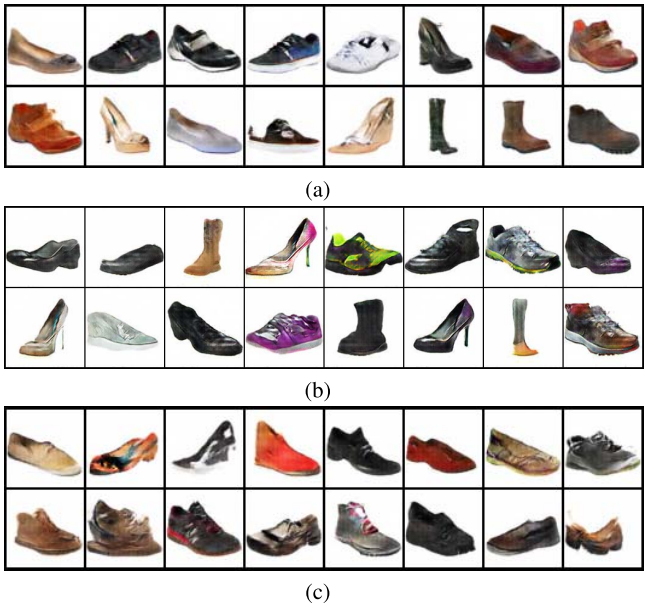

Fig. 2. *Shoe samples synthesized using GANs* represent samples from DCGANs trained on (a) lower resolution (64 × 64) images, (b) higher resolution images (128 × 128), and (c) samples from a WGAN.

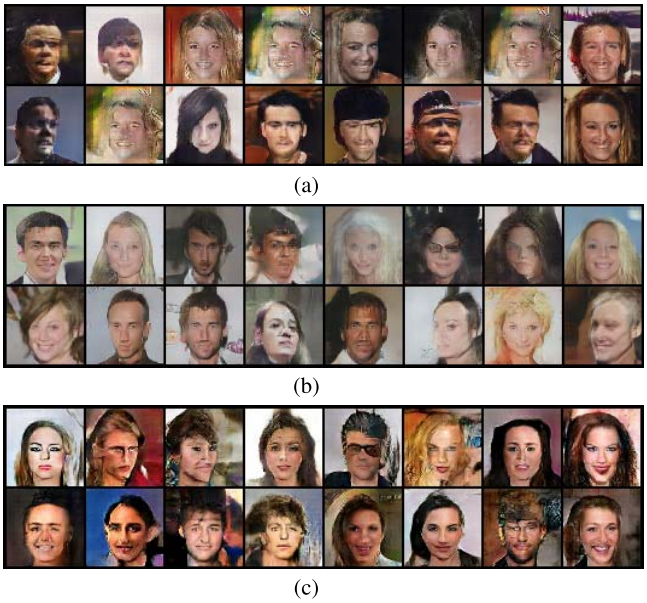

Fig. 3. *Celebrity faces synthesized using GANs* represent samples from DCGANs trained (a) without noise and (b) with noise. (c) Samples from a WGAN.

the second we train on 64 × 64 images. The networks were trained according to the setup described by Radford *et al.* [20], using a multivariate Gaussian prior. We also train a WGAN [2] on full sized images. All GANs are trained with additive noise whose standard deviation decays during training [1]. Fig. 3 shows samples randomly synthesized using the DCGAN models trained on shoes. The samples look quite realistic, but again, they do not tell us much about the representations learned by the GANs.

C. CelebA

The CelebA data set consists of 250 000 celebrity faces, in RGB color. The images are of dimensions 64 × 64 pixels. We leave 1000 samples out for testing and use the rest for training. We train three models, a DCGAN and WGAN trained with decaying noise [1] and a DCGAN trained without noise. The networks are trained according to the setup described by Radford *et al.* [20]. Fig. 3(c) shows examples of faces synthesized with and without noise. It is clear from Fig. 3(a) and (c) that the GAN trained without noise has collapsed, synthesizing similar examples for different latent codes. The WGAN produces the sharpest and most varied samples. However, these samples do not provide sufficient information about the representation power of the models.

V. Experiments

To obtain latent representations, z^* for a given image x we apply our proposed inversion technique to a batch of randomly selected test images, $x \in X$. To invert a batch of image samples, we minimized the cost function described by (5). In most of our experiments, we use $\beta = 0.01$, unless stated otherwise, and update candidate z^* using an RMSprop optimiser, with a learning rate of 0.01.

A valid inversion process should map a target image sample, $x \in X$ to a $z^* \in Z$, such that when z^* is passed through the generative part of the GAN, it produces an image, $G(z^*)$, that is close to the target image, x. However, the quality of the reconstruction depends heavily on the latent representation that the generative model has learned. In the case, where a generative model is only able to represent some attributes of the target image, x, the reconstruction, $G(z^*)$ may only partially reconstruct x.

Thus, the purpose of our experiments is twofold.

1) To demonstrate qualitatively, through reconstruction, $(G(z^*))$, that for most well-trained GANs, our inversion process is able to recover a latent code, z^*, that captures **most of the important features** of a target image (Section VI).
2) To demonstrate how our proposed inversion technique may be used to both qualitatively (Section VI-B) and **quantitatively** compare GAN models (Section VII).

VI. Reconstruction Results

A. Omniglot

The Omniglot inversions are particularly challenging, as we are trying to find a set of z^*'s for a set of characters, x, from alphabets that were not in the training data. The inversion process will involve finding representations for data samples from alphabets that it has not seen before, using information about alphabets that it has seen. The original and reconstructed samples are shown in Fig. 4.

In our previous work [6], we showed that given the "correct" architecture, we are able to find latent representations that lead to excellent reconstructions. However, here we focus on evaluating standard models [20] and we are particularly interested in detecting (and quantifying) where models fail, especially since visual inspection of synthesized samples may not be sufficient to detect the model failure.

It is clear from Fig. 4 that the GAN has overfit; however, it was less clear whether or not the WGAN has overfit, since the samples appeared to be more varied. By attempting to

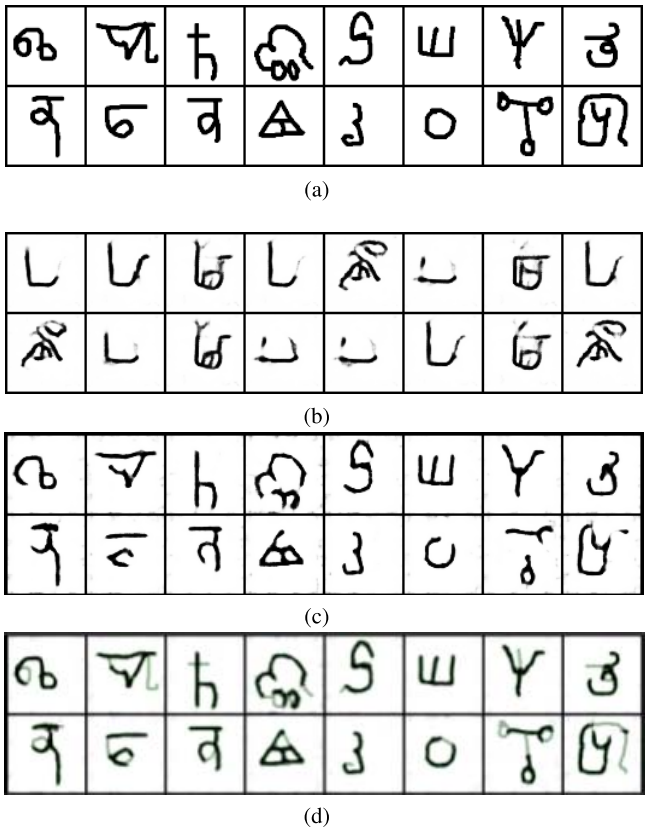

Fig. 4. *Reconstruction of Omniglot handwritten characters.* (a) Target Omniglot handwritten characters, x, from alphabets different to those seen during training. (b) Reconstructed data samples, $G(z^*)$, using a GAN. (c) Reconstructed data samples, $G(z^*)$, using a WGAN. (d) Reconstructed data samples, $G(z^*)$, using a WGAN overlaid with x.

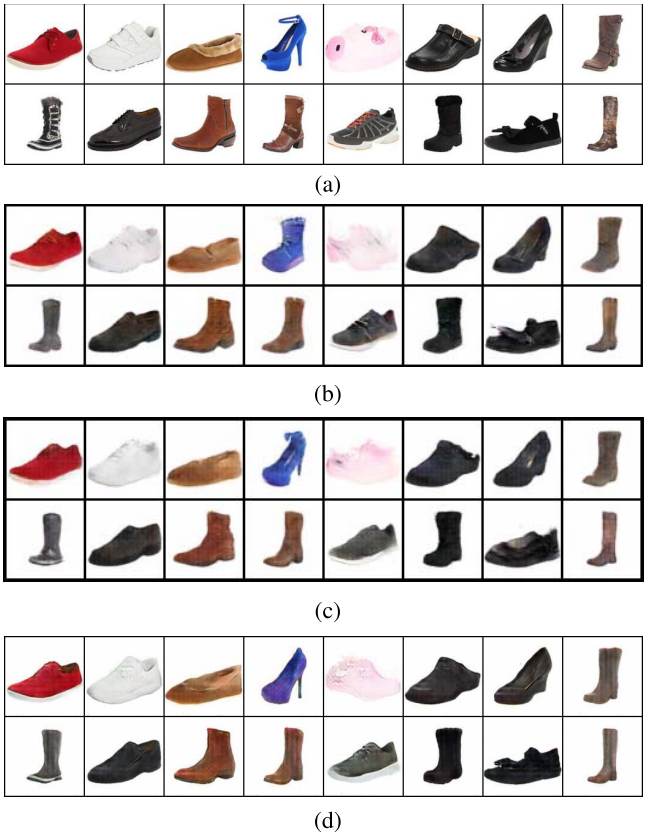

Fig. 5. *Reconstruction of Shoes.* (a) Shoe data samples, x, from a test set. (b) Reconstructed data samples, $G(z^*)$, using a GAN at resolution 64×64. (c) Reconstructed data samples, $G(z^*)$, using a WGAN at resolution 64×64. (d) Reconstructed data samples, $G(z^*)$, using a GAN at resolution 128×128. By comparing reconstructions, particularly of the blue shoe, we see that the higher resolution model (d) is able to capture some structural details, particularly the shoe's heel, which the lower resolution model (b) does not. Furthermore, the WGAN at 64×64 (c) is able to capture additional detail than the GAN at 64×64, including the blue shoe's strap. These results demonstrate how inversion may be a useful tool for comparing which features of a data set each model is able to capture.

perform inversion, we can see that the WGAN has indeed overfit, as it is only able to partially reconstruct the target data samples. In Section VII, we quantitatively compare the extent to which the GAN and WGAN trained on the Omniglot data set have overfit.

B. Shoes

In Fig. 5, we compare shoe reconstructions using a DCGAN trained on low- and high-resolution images. By comparing all reconstructions in Fig. 5(b) and (c) (particularly the blue shoe on the top row) we see that the lower resolution model has failed to capture some structural details, while the higher resolution model has not. This suggests that the model trained on higher resolution images is able to capture more structural details than the model trained on lower resolution images. Using our inversion technique to make comparisons between models is just one example of how inversion may also be used to "trouble-shoot" and identify which features of a data set our models are not capturing.

In addition, we may observe that while the GAN trained on higher resolution images preserves more structure than the GAN trained on lower resolution images, it still misses certain details. For example, the reconstructed red shoes do not have laces [Fig. 5(b) and (c) (top left)]. This suggests that the representation is not able to distinguish shoes with laces from those without. This may be important when designing representations for image retrieval, where a retrieval system using this representation may be able to consistently retrieve red shoes, but less consistently retrieve red shoes with laces. This is another illustration of how a good inversion technique may be used to better understand what representation is learned by a GAN.

Fig. 5(d) shows the reconstructions using a WGAN trained on low-resolution images. We see that the WGAN is better able to model the blue shoe, and some ability to model the ankle strap, compared to the GAN trained on higher resolution images. It is, however, difficult to access from reconstructions, which model represents the data best. In Section VII, we show how our inversion approach may be used to quantitatively compare these models, and determine which learns a better (latent) representation for the data.

Finally, we found that while the regularization of the latent space may not always improve reconstruction fidelity, it can be helpful for ensuring that latent encodings, z^*, found through inversion, correspond to images, $G(z^*)$ that look more like shoes. Our results in Fig. 5 were achieved using $\beta = 0.01$.

C. CelebA

Fig. 6 shows the reconstructions using three different GAN models. Training GANs can be very challenging, and so

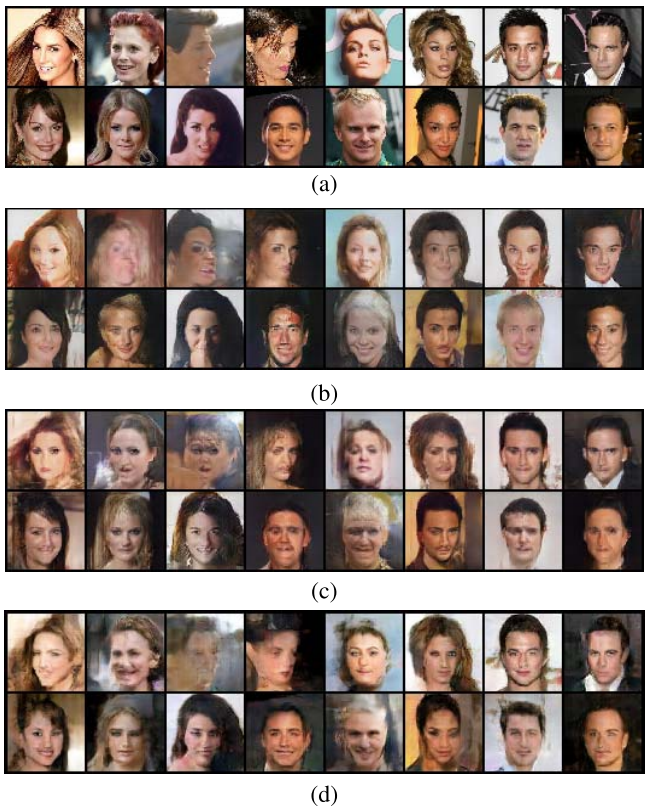

Fig. 6. *Reconstruction of celebrity faces.* (a) CelebA faces, x, from a test dataset. (b) Reconstructed data samples, $G(z^*)$, using a GAN. (c) Reconstructed data samples, $G(z^*)$, using a GAN + noise. (d) Reconstructed data samples, $G(z^*)$, using a WGAN. In this example, we use the inversion process to compare models trained on CelebA using (b) a GAN, (c) a GAN + noise, and (d) a WGAN. The visual results should be compared with the first column of Table I.

various modifications may be made to their training to make them easier to train. Two examples of modifications are: 1) adding corruption to the data samples during training [1] and 2) a reformulation of the cost function to use the Wasserstein distance. Although these techniques are known to make training more stable, and perhaps also prevent other pathologies found in GANs, e.g., mode collapse [22], we are interested in comparing the (latent) representations learned by these models.

The most faithful reconstructions appear to be those from the WGAN as shown in Fig. 6(b). This will be confirmed quantitatively in Section VII. By observing reconstruction results across all models in Fig. 6, it is apparent that all three models fail to capture a particular mode of the data; all three models fail to represent profile views of faces.

VII. Quantitatively Comparing Models

Failing to represent a mode in the data is commonly referred to a "mode dropping," and is just one of three common problems exhibited by GANs. For completeness, common problems exhibited by trained GANs include the following: 1) mode collapse [2], [22], this is where similar image samples are synthesized for different inputs; 2) mode dropping [19], where the GAN only captures certain regions of high density in the data generating distribution; and 3) training sample memorisation, where the GAN memorizes and reproduces samples seen in the training data. If a model exhibits these symptoms, we say that it has overfit; however, these symptoms are often difficult to detect.

TABLE I

Comparing Models Using Our Inversion Approach MSE Is Reported Across All Test Samples for Each Model Trained With Each Data Set. A Smaller MSE Suggests That the Model Is Better Able to Represent Test Data Samples

Model	CelebA	Shoes	Omniglot
GAN [20]	0.118	0.059	0.588
GAN+noise [1]	0.109	0.029	0.305
WGAN [2]	0.042	0.020	0.082
High Res. GAN	-	0.016	-

If a GAN is trained well and exhibits none of the above-mentioned three problems, it should be possible to perform inversion to find suitable representations for most test samples using our technique.

However, if a GAN does exhibit any of the three problems listed above, inversion becomes challenging, since certain regions of high density in the data generating distribution may not be represented by the GAN. Thus, we may compare GAN models, by evaluating the reconstruction error using our proposed inversion process. A high reconstruction error, in this case, mean squared error (MSE), suggests that a model has possibly overfit, and is not able to represent data samples well. By comparing MSE between the models, we can compare the extent to which one model has overfit compared to another.

Table I shows how our inversion approach may be used to quantitatively compare three models (four in the case of the shoes data set) across three data sets, CelebA, shoes, and Omniglot. This table gives mean squared reconstruction error on a large batch of test samples; for CelebA we used 100 samples and for Shoes and Omniglot we used 500 samples.

From Table I, we may observe the following.

A. CelebA

The (latent) representation learned by the WGAN generalizes to test samples, better than either the GAN or the GAN trained with noise. Results also suggest that training a GAN with noise helps to prevent overfitting. These conclusions are consistent with both empirical and theoretical results found in [1] and [2], suggesting that this approach for quantitatively comparing models is valid.

B. Shoes

Using inversion to quantify the quality of a representation allows us to make fine-grained comparisons between models. We see that training a model using higher resolution images reduces reconstruction error by almost a factor of two, in the case of the GAN + noise, compared to a similar model trained at a lower resolution. This is in agreement with earlier observations [Fig. 5(b)], in which we saw that certain structural detail—such as the presence of a well-defined protrusion—was lost.

Comparing models using our proposed inversion approach, in addition to classifier-based measures [21], helps to detect

fine-grained differences between models, which may not be detected using classification-based measures alone. A "good" discriminative model learns many features that help to make decisions about which class an object belongs to. However, any information in an image that does not aid classification is likely to be ignored; for example, when classifying cars and trucks, the color of the vehicle is unimportant. As an example, [18, Fig. 10 (bottom left)] shows that the first layer of a discriminatively trained CNN ignores the colors of the input image. Yet, we may want to compare representations that encode information that a classifier ignores (e.g., color of the car). For this reason, using only a classification-based measure [21] to compare representations learned by different models may not be enough, or may require very precise classifiers to detect differences.

C. Omniglot

From Fig. 4, it was clear that both models trained on the Omniglot data set had overfit, but not to the same extent. Here, we are able to quantify the degree to which each model has overfit. We see that the WGAN has overfit to a lesser extent than the GAN trained with noise, since the WGAN has a smaller MSE. Quantifying overfitting can be useful when developing new architectures, and training schemes, to objectively compare models.

In this section, we have shown how a particular inversion approach may be used to quantitatively compare representations learned by GANs. We intend this approach to provide a useful, quantitative means of evaluating and developing new GAN models and architectures for representation learning.

Finally, we emphasize that while there are other techniques that provide inversion, our proposed technique is the only one that is both: 1) immune to overfitting, in other words, we do not train an encoder network that may itself overfit and 2) can be applied to any pretrained GAN model, provided that the computational graph is available.

VIII. Conclusion

The generator of a GAN learns the mapping $G : Z \rightarrow X$. It has been shown that z values that are close in Z-space produce images that are visually similar in image space, X [20]. We propose an approach to map data, x samples back to their latent representation, z^* (Section II).

For a generative model, in this case, a GAN, which is trained well and given target image, x, we should be able to find a representation, z^*, that when passed through the generator, produces an image, $G(z^*)$, that is similar to the target image. However, it is often the case that GANs are difficult to train, and there only exists a latent representation, z^*, which captures some of the features in the target image. When z^* only captures some of the features, this results in, $G(z^*)$, being a partial reconstruction, with certain features of the image missing. Thus, our inversion technique provides a tool, to provide qualitative information about what features are captured by the (latent) representation of a GAN. We showed several visual examples of this in Section VI.

Often, we want to compare models quantitatively. In addition to providing a qualitative way to compare models, we show how we may use mean squared reconstruction error between a target image, x and $G(z^*)$, to quantitatively compare models. In our experiments, in Section VII, we use our inversion approach to quantitatively compare three models trained on three data sets. Our quantitative results support claims from previous work that suggests, that certain modified GANs are less likely to overfit.

We expect that our proposed inversion approach may be used as a tool to access and compare various proposed modifications to generative models, and aid the development of new generative approaches to representation learning.

Acknowledgment

The authors would like to thank the Engineering and Physical Sciences Research Council for funding through a Doctoral Training studentship (EP/L504786/1).

References

[1] M. Arjovsky and L. Bottou. (2017). "Towards principled methods for training generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1701.04862

[2] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.

[3] P. Ballester and R. M. Araujo, "On the performance of GoogLeNet and AlexNet applied to sketches," in *Proc. AAAI*, 2016, pp. 1124–1128.

[4] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Adv. Neural Inf. Processing Syst.*, 2016, pp. 2172–2180.

[5] A. Creswell and A. A. Bharath. (2016). "Task specific adversarial cost function." [Online]. Available: https://arxiv.org/abs/1609.08661

[6] A. Creswell and A. A. Bharath. (2018). "Inverting the generator of a generative adversarial network (II)." [Online]. Available: https://arxiv.org/abs/1802.05701

[7] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2625–2634.

[8] J. Donahue, P. Krähenbühl, and T. Darrell. (2016). "Adversarial feature learning." [Online]. Available: https://arxiv.org/abs/1605.09782

[9] V. Dumoulin *et al.* (2016). "Adversarially learned inference." [Online]. Available: https://arxiv.org/abs/1606.00704

[10] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[11] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5767–5777.

[12] D. J. Im, C. D. Kim, H. Jiang, and R. Memisevic, "Generating images with recurrent adversarial networks," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR) Workshop Track*, 2016.

[13] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," in *Proc. Int. Conf. Learn. Represent.*, 2018.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[15] B. M. Lake, R. Salakhutdinov, and J. B. Tenebaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[16] C. Li *et al.*, "ALICE: Towards understanding adversarial learning for joint distribution matching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5501–5509.

[17] J. Luo, Y. Xu, C. Tang, and J. Lv, "Learning inverse mapping by AutoEncoder based generative adversarial nets," in *Proc. Int. Conf. Neural Inf. Process.* Springer, 2017, pp. 207–216.

[18] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5188–5196.

[19] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2017.

[20] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. 5th Int. Conf. Learn. Represent. (ICLR) Workshop Track*, 2016.
[21] T. Salimans *et al.*, "Improved techniques for training GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2234–2242.
[22] A. Srivastava, L. Valkoz, C. Russell, M. U. Gutmann, and C. Sutton, "VEEGAN: Reducing mode collapse in GANs using implicit variational learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3308–3318.
[23] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[24] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional GANs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 1–13.
[25] A. Yu and K. Grauman, "Fine-grained visual comparisons with local learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 192–199.
[26] H. Zhang *et al.*, "StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2017, pp. 5908–5916.
[27] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative visual manipulation on the natural image manifold," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 597–613.

Antonia Creswell recieved the M.Eng. degree in biomedical engineering from the Imperial College London, London, U.K., where she is currently pursuing the Ph.D. degree.

She has been an Intern with Magic Pony, Twitter, U.K., and DeepMind, Google, U.K.

Anil Anthony Bharath received the B.Eng. degree in electronic and electrical engineering from the University College London, London, U.K., in 1988, and the Ph.D. degree in signal processing from the Imperial College London, London, in 1993.

He was an Academic Visitor with the Signal Processing Group, University of Cambridge, Cambridge, U.K., in 2006. He is currently a Reader with the Department of Bioengineering, Imperial College London. He is a Fellow of the Institution of Engineering and Technology and the Imperial College Data Science Institute. He is also a Co-Founder of Cortexica Vision Systems, London. His current research interests include deep architectures for visual inference.