Adversarial Generator-Encoder Networks

Dmitry Ulyanov [1 2] **Andrea Vedaldi** [3] **Victor Lempitsky** [1]

Abstract

We present a new autoencoder-type architecture, that is trainable in an unsupervised mode, sustains both generation and inference, and has the quality of conditional and unconditional samples boosted by adversarial learning. Unlike previous hybrids of autoencoders and adversarial networks, the adversarial game in our approach is set up directly between the encoder and the generator, and no external mappings are trained in the process of learning. The game objective compares the divergences of each of the real and the generated data distributions with the canonical distribution in the latent space. We show that direct generator-vs-encoder game leads to a tight coupling of the two components, resulting in samples and reconstructions of a comparable quality to some recently-proposed more complex architectures.

1. Introduction

Deep architectures such as autoencoders (Bengio, 2009) and their variational counterparts (VAE) (Rezende et al., 2014; Kingma & Welling, 2013) can learn in an unsupervised manner a bidirectional mapping between data characterised by a complex distribution, such as natural images, and a much simpler latent space. Sampling an image reduces then to drawing a sample in the latent space, which is easy, and then mapping the latter to a data sample by applying the learned generator function to it. Unfortunately, the perceptual quality of the generated samples remains relatively low. This is usually not a limitation of the neural network *per-se*, but rather of the simplistic loss functions used to train it. In particular, simple losses are unable to properly account for reconstruction ambiguities and result in blurry samples (regression to the mean). By contrast, Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) can learn complex loss functions as a part of the learning process, which allows them to generate better quality samples.

A disadvantage of GANs compared to autoencoders is that, in their original form, they are unidirectional: a GAN can only generate a data sample from a latent space sample, but it cannot reverse this mapping. By contrast, in autoencoders this inference process is carried by the encoder function, which is learned together with the generator function as part of the model. Therefore, there has been some interest in developing architectures that support both sampling and inference like autoencoders, while producing samples of quality comparable to GAN. For example, the adversarial autoencoders of (Makhzani et al., 2015) augment autoencoders with adversarial discriminators encouraging the alignment of distributions in the latent space, but the quality of the generated samples does not always match GANs. The approach (Larsen et al., 2015) adds instead an adversarial loss to the reconstruction loss of the variational autoencoder, with a sensible improvement in the quality of the samples. The method of (Zhu et al., 2016) starts by learning the decoder function as in GAN, and then learns a corresponding encoder post-hoc. The adversarially-learned inference (ALI) approach, simultaneously proposed by (Donahue et al., 2016; Dumoulin et al., 2016), consider parallel learning of encoders and generators, whereas the distribution of their input-output pairs are matched by an external discriminator.

All such approaches (Makhzani et al., 2015; Larsen et al., 2015; Donahue et al., 2016; Dumoulin et al., 2016) add to the encoder (or inference network) and decoder (or generator) modules a discriminator module, namely an external classifier whose goal is to align certain distributions in the latent or data space, or both.

In this paper, we propose a new approach for training encoder-generator pairs that uses an adversarial game between the encoder and the generator, without the need to add an external discriminator. Our architecture, called *Adversarial Generator-Encoder Network* (AGE Network), thus consists of only two feed-forward mappings (the encoder and the generator). At the same time, AGE uses adversarial training to optimize the quality of the generated samples by matching their higher-level statistics to those of real data samples. Crucially, we demonstrate that it is possible to use the encoder itself to extract the required statis-

[1]Skolkovo Institute of Science and Technology, Russia [2]Yandex, Russia [3]University of Oxford, UK. Correspondence to: Dmitry Ulyanov <dmitry.ulyanov.msu@gmail.com>.

Source code is available at https://github.com/DmitryUlyanov/AGE

tics.

Adversarial training in AGE works as follows. First, the encoder is used to map both real and generated data samples to the latent space, inducing in this space two empirical data distributions. Since the aim of learning is to make generated and real data statistically indistinguishable, the the generator is required to make these two latent distributions identical. At the same time, the (adversarial) goal of the encoder is to construct a latent space where any statistical difference that can be discovered between the real and generated data is emphasized.

Importantly, AGE compares the latent distributions indirectly, by measuring their divergence to a reference canonical distribution (e.g. an i.i.d. uniform or normal vector). The evaluation of the game objective thus requires estimation of a divergence between distributions represented by sets of samples and an "easy" tractable distribution, for which we use a simple parametric or non-parametric estimator. The encoder is required to push the divergence of the real data down, a goal shared with VAE which makes the distribution of real data in latent space a simple one. The generator is also required to push down the divergence of the latent distribution corresponding to the generated data; in this manner, the latent statistics of real and generated data is encouraged to match. However, adversarially, the encoder is also encouraged to maximise the divergence of the generated data.

The AGE adversarial game has a learning objective that is quite different from the objectives used by existing adversarial training approaches. Since our goal is to avoid introducing a discriminator function, this is partly out of necessity, since neither generator nor encoder can be used as binary classifiers. Furthermore, introducing a binary classifier that classifies individual samples as generated or real has known pitfalls such as mode collapse (Goodfellow, 2017); instead, the discriminator should look at the statistics of multiple samples (Salimans et al., 2016). Our approach does so by means of a new multi-sample objective for adversarial training.

As we show in the experiments, adversarial training with the new objective is able to learn generators that produces high-quality samples even without reconstruction losses. Our new game objective can thus be used for stand-alone adversarial learning. Our approach is evaluated on a number of standard datasets. We include comparisons with the adversarially-learned inference (ALI) system of (Dumoulin et al., 2016) as well as the base generative adversarial architecture (Radford et al., 2015). We show that, for many different datasets, AGE networks achieve comparable or better sampling and reconstruction quality than such alternatives.

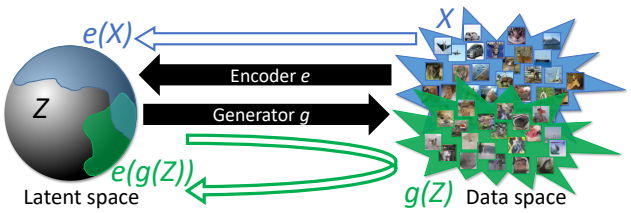


Figure 1. Our model (AGE network) has only two components: the generator g and the encoder e. The learning process adjusts their parameters in order to align a simple distribution Z in the latent space and the data distribution X. This is done by adversarial training, as updates for the generator aim to minimize the divergence between $e(g(Z))$ and Z (aligning green with gray), while updates for the encoder aim to minimize the divergence between $e(X)$ (aligning blue with gray) and to *maximize* the divergence between $e(g(Z))$ and Z (shrink green "away" from gray). We demonstrate that such adversarial learning gives rise to high-quality generators that result in the close match between the real distribution X and the generated distribution $g(Z)$. Our learning can also incorporate reconstruction losses to ensure that encoder-generator acts as autoencoder (section 2.2).

A note about notation: To ease the notation for a distribution Y and a deterministic mapping f, we use the shorthand $f(Y)$ to denote the distribution associated with the random variable $f(\mathbf{y})$, $\mathbf{y} \sim Y$.

2. Adversarial Generator-Encoder Networks

Adversarial Generator-Encoder (AGE) networks are composed of two parametric mappings that go in reverse directions between the *data space* $\mathcal{X}$ and the *latent space* $\mathcal{Z}$. The *encoder* $e_\psi(\mathbf{x})$ with the learnable parameters ψ maps data space to latent space, while the *generator* $g_\theta(\mathbf{z})$ with the learnable parameters θ maps latent space to data space.

The goal of the training process for AGE is to align a real data distribution X with a "fake" distribution $g_\theta(Z)$ and establish a reciprocal correspondence between X and Z at the sample level. The real data distribution X is represented by a sufficiently large number N of samples $\{\mathbf{x}_1, \mathbf{x}_2, ...\mathbf{x}_N\}$ that follow this distribution; in the latent space, a simple distribution Z is chosen, from which samples can be drawn easily. The training process corresponds to the tuning of the parameter sets ψ and θ. The process combines adversarial learning with the traditional minimization of reconstruction errors in both spaces.

The simple form of the distribution Z allows unconditional feed-forward sampling from the data distribution using AGE networks by sampling $\mathbf{z} \sim Z$ and computing $\mathbf{x} = g_\theta(\mathbf{z})$, exactly as it is done by sampling from a generator in GANs. In our experiments, we pick the latent space $\mathcal{Z}$ to be an M-dimensional sphere $\mathbb{S}^M$, and the latent distribution to be a uniform distribution on that sphere

$Z = \text{Uniform}(\mathbb{S}^M)$. We have also conducted some experiments with the unit Gaussian distribution in the Euclidean space and have obtained results comparable in quality.

In section 2.1 we introduce a game objective for which each saddle g delivers alignment $g(Z) = X$. We then augments it with additional terms that encourage the reciprocity of e and g in section 2.2. Section 2.3 describes the details of the learning process for the introduced game.

2.1. Adversarial distribution alignment

The conventional approach to aligning two distributions is implemented in existing GAN-based systems via an adversarial game based around ratio estimation (Goodfellow et al., 2014). The ratio estimation is performed by repeated fitting of the binary classifier that distinguishes between the two distributions (corresponding to real and generated samples). Here, we propose an alternative approach avoiding some of the pitfalls of the conventional GANs such as mode collapse.

Our primary goal is to find generators that produce distributions in the data space $g(Z)$ that are close to the true data distribution X. However, direct matching of the distributions in a high-dimensional data space can be very challenging, thus we would like to limit ourself to comparing only distributions defined in the latent space. In the following derivations, we therefore introduce a divergence measure $\Delta(P\|Q)$ between distributions defined in the latent space $\mathcal{Z}$. We only require this divergence to be non-negative and zero if and only if the distributions are identical $\Delta(P\|Q) = 0 \iff P = Q$ (triangle inequality and symmetry property should not necessarily hold). An encoder e maps distributions X and $g(Z)$ in the latent space to the distributions $e(X)$ and $e(g(Z))$ in the latent space. Below, we show how to design an adversarial game between e and g that ensures the alignment of $g(Z) = X$ in the data space, while only evaluating divergences in the latent space.

In the theoretical analysis below, we assume that considered encoders and decoders span the class of all measurable mappings between the corresponding spaces. Such assumption (often referred to as *non-parametric limit*) is justified by universality of neural networks (Hornik et al., 1989). We further make an **assumption** that there exists at least one "perfect" generator that matches the data distribution, i.e. $\exists g_0 : g_0(Z) = X$.

We start by considering a simple game with objective defined as:

$$\max_e \min_g V_1(g, e) = \Delta(\, e(g(Z))\|e(X)\,)\,. \qquad (1)$$

As the following theorem shows, perfect generators form saddle points (Nash equilibria) of the game (1) and all saddle points of the game (1) are based on perfect generators.

Theorem 1. *A pair (g^*, e^*) forms a saddle point of the game (1) if and only if the generator g^* matches the data distribution, i.e. $g^*(Z) = X$.*

The proof of the theorem is given in the appendix.

While the game (1) is sufficient for aligning distributions in the data space, finding its saddle points is complicated by the need to compare the two general-form distributions given in the form of samples. This is aided by redesigning the game as follows:

$$\max_e \min_g V_2(g, e) = \Delta(e(g(Z))\|Y) - \Delta(e(X)\|Y). \quad (2)$$

Here Y is a fixed distribution in the latent space. Importantly, as the following theorem demonstrates, the new game still retains the connection between generators that align fake and real distributions and saddle points.

Theorem 2. *If a pair (g^*, e^*) is a saddle point of game (2) then the generator g^* matches the data distribution, i.e. $g^*(Z) = X$. Conversely, if the generator g^* matches the data distribution, then for some e^* the pair (g^*, e^*) is a saddle point of (2).*

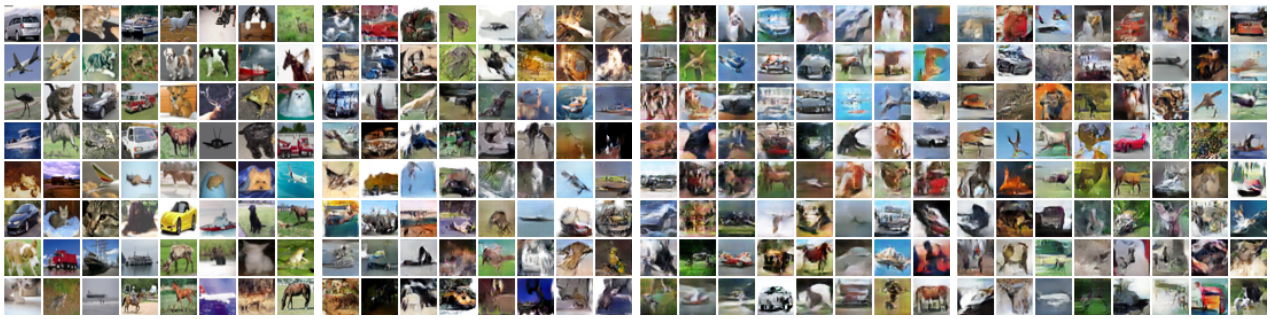
The proof is given in the appendix.

The important benefit of the new game formulation (2) is that the model distributions $e(g(Z))$ and $e(X)$ are now compared with a fixed Y. By picking Y suitably, we can ensure that the new divergence evaluations are more stable than a direct comparison $\Delta(\, e(g(Z))\|e(X)\,)$, as distributions $e(g(Z))$ and $e(X)$ are defined implicitly (we do not have access to the distributions directly, but can only sample from them).

Conveniently, one can pick the target distribution Y to coincide with the "canonical"(source) distribution Z, and we do so in all our experiments and further derivations. With $Y = Z$ the game objective (2) can be upgraded with reconstruction losses and permits convenient stochastic approximation as described in section 2.3.

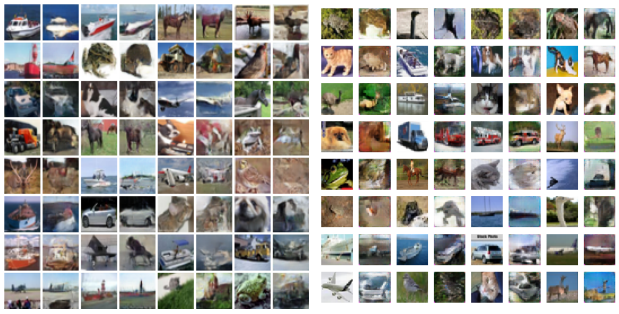
One could also interpret the game (2) as the comparison between the two distributions in the data space via comparison of certain statistics extracted by the encoders. Similarly to conventional GANs, these statistics are not fixed but evolve in the process of the game. In our approach the statistics have the form $F(Q) = \Delta(e(Q)\|Y)$ for a given distribution Q in the data space and a fixed distribution Y in the latent space. The statistics thus map a data space distribution into the latent space and then measure its divergence with the distribution Y.

We note that even away from the saddle point, the minimization $\min_g V_2(g, e)$ for some fixed e does not have a



(a) Real (b) DCGAN (c) Ours (game 2) (d) Ours (full)

Figure 2. We compare CIFAR10 samples from DCGAN (Radford et al., 2015) (b) to the samples generated using our ablated model trained without reconstruction terms in (). The model, trained with the reconstruction terms is still able to produce diverse samples (d), but also allows inference (Figure 3).



(a) Ours. (b) ALI (Dumoulin et al., 2016).

Figure 3. Comparison in reconstruction quality to ALI (Dumoulin et al., 2016) for the CIFAR10 dataset. For both figures real examples are shown in odd columns and their reconstructions are shown in the even columns. The real examples come from test set and were never observed by the model during training.

collapsing tendency for many reasonable divergences (e.g. KL-divergence). Indeed, any collapsed distribution would inevitably lead to a very high value of the first term in (2). Thus, unlike GANs, our approach can optimize the generator for a fixed adversary till convergence and obtain non-degenerate solution. On the other hand, the maximization $\max_e V_2(g, e)$ for some fixed g can lead to $+\infty$ score for some divergences.

2.2. Reconstruction losses

While the previous derivation demonstrates that X and $g(Z)$ can get aligned by adversarial learning, such alignment does not necessarily entails reciprocity of the e and g mappings at the level of individual samples. Such sample-level reciprocity is not needed if we only need highly-realistic samples from the generator, however it is desirable if good reconstructions from the composition of encoder and generator are sought.

The sample-level reciprocity can be measured either in the latent space or in the data space. This gives rise to the two loss functions:

$$L_{\mathcal{X}}(g_\theta, e_\psi) = \mathbb{E}_{\mathbf{x}\sim X}\|\mathbf{x} - g_\theta\left(e_\psi(\mathbf{x})\right)\|^2, \quad (3)$$

$$L_{\mathcal{Z}}(g_\theta, e_\psi) = \mathbb{E}_{\mathbf{z}\sim Z}\|\mathbf{z} - e_\psi\left(g_\theta(\mathbf{z})\right)\|^2. \quad (4)$$

Both losses (3) and (4) thus measure the reconstruction error and their minimization ensures the reciprocity of the two mappings. The loss (3) is the traditional loss used within autoencoders.

As we aim to improve game (2), we may want to understand if both losses (3) and (4) should be used or one of them would be sufficient in theory. The answer is given by the following statement:

Theorem 3. *Let the two distributions W and Q be aligned by the mapping f (i.e. $f(W) = Q$) and let $\mathbb{E}_{\mathbf{w}\sim W}\|\mathbf{w} - h\left(f(\mathbf{w})\right)\|^2 = 0$. Then, for $\mathbf{w} \sim W$ and $\mathbf{q} \sim Q$, we have $\mathbf{w} = h(f(\mathbf{w}))$ and $\mathbf{q} = f(h(\mathbf{q}))$ almost certainly, i.e. the mappings f and h invert each other almost everywhere on the supports of W and Q. More, Q is aligned with W by h: $h(Q) = W$.*

The proof is given in the appendix.

Recall that theorem 2 establishes that the solution (saddle point) of game (2) aligns distributions in the data space. Then, following theorem 3, adding the latent space loss (4) to the objective (2) is sufficient to ensure reciprocity.

2.3. Training AGE networks

Based on the theoretical analysis derived in the previous subsections, we now suggest the approach to the joint training of the generator in the encoder within the AGE networks. As in the case of GAN training, we set up the learning process for an AGE network as a game with the iterative updates over the parameters θ and ψ that are driven by the

optimization of different objectives. In general, the optimization process combines the maximin game for the functional (2) with the optimization of the reciprocity losses (3) and (4).

In particular, we use the following game objectives for the generator and the encoder:

$$\hat{\theta} = \arg\min_{\theta} \left(V_2(g_\theta, e_{\bar{\psi}}) + \lambda L_{\mathcal{Z}}(g_\theta, e_{\bar{\psi}}) \right) , \quad (5)$$

$$\hat{\psi} = \arg\max_{\psi} \left(V_2(g_{\bar{\theta}}, e_\psi) - \mu L_{\mathcal{X}}(g_{\bar{\theta}}, e_\psi) \right) , \quad (6)$$

where $\bar{\psi}$ and $\bar{\theta}$ denote the value of the encoder and generator parameters at the moment of the optimization and λ, μ is a user-defined parameter. Note that both objectives (5), (6) include only one of the reconstruction losses. Specifically the generator objective includes only the latent space reconstruction loss. In the experiments, we found that the omission of the other reconstruction loss (in the data space) is important to avoid possible blurring of the generator outputs that is characteristic to autoencoders. Similarly to GANs in (5), (6) we perform only several steps toward optimum at each iteration, thus alternating between generator and encoder updates.

By maximizing the difference between $\Delta(e_\psi(g_{\bar{\theta}}(Z))\|Z)$ and $\Delta(e_\psi(X)\|Z)$, the optimization process (6) focuses on the maximization of the mismatch between the real data distribution X and the distribution of the samples from the generator $g_{\bar{\theta}}(Z)$. Informally speaking, the optimization (6) forces the encoder to find the mapping that aligns real data distribution X with the target distribution Z, while mapping non-real (synthesized data) $g(Z)$ away from Z. When Z is a uniform distribution on a sphere, the goal of the encoder would be to uniformly spread the real data over the sphere, while cramping as much of synthesized data as possible together assuring non-uniformity of the distribution $e_\psi\left(g_{\bar{\theta}}(Z)\right)$.

Any differences (misalignment) between the two distributions are thus amplified by the optimization process (6) and forces the optimization process (5) to focus specifically on removing these differences. Since the misalignment between X and $g(Z)$ is measured after projecting the two distributions into the latent space, the maximization of this misalignment makes the encoder to compute features that distinguish the two distributions.

3. Experiments

3.1. Implementation details

Network architectures: In our experiments the generator and the encoder networks have a similar structure to the generator and the discriminator in DCGAN (Radford et al., 2015). As our encoder should produce a vector instead of a single number we modify the DCGAN's discriminator accordingly, expanding the last layer output to M dimensions. We also replace the sigmoid at the end with the normalization layer which projects the points onto the sphere.

Divergence measure: We use the following expression to measure the divergence between a distribution Q in the latent space and the distribution Z (which is the uniform distribution on the M-dimensional sphere $\mathbb{S}^M$):

$$\Delta(Q\|U(\mathbb{S}^M)) = \mathrm{KL}(Q\|\mathcal{N}(0, I)) - C\,. \quad (7)$$

Here, $\mathcal{N}(0, I)$ is the zero-mean unit-variance Gaussian distribution in the embedding space $\mathbb{R}^M$, and the constant C equals $C = \mathrm{KL}(U(\mathbb{S}^M)\|\mathcal{N}(0, I))$. Since the uniform distribution on the sphere minimizes the KL-divergence with the unit Gaussian out of all distributions on the sphere, the expression (7) is valid (is non-negative, and equals zero only for $Q = Z$).

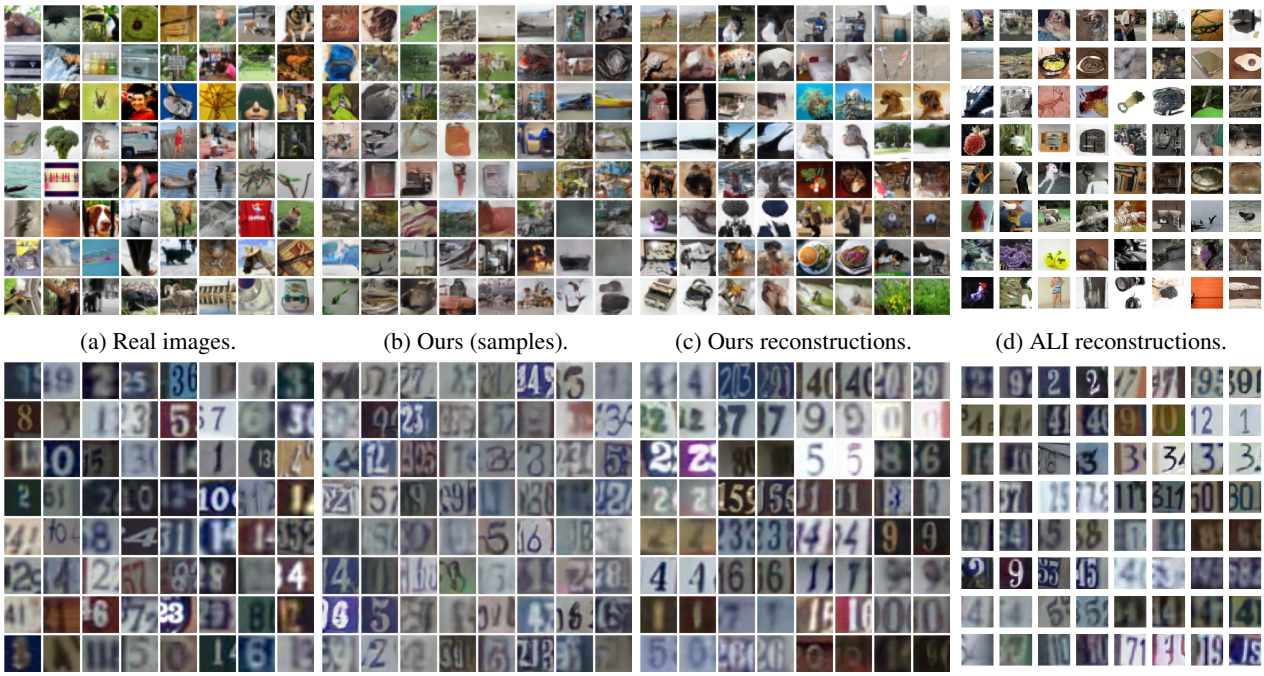
To measure the value $\mathrm{KL}(Q\|\mathcal{N}(0, I))$ for a mini-batch of examples, we used a parametric estimator, where a Gaussian in the embedded space is fitted to the mini-batch and the KL-divergence between Gaussians is computed analytically. Having a set of M dimensional samples $\mathbf{q}_i \sim Q$, $i = 1, \ldots, N$ with a component-wise mean m: $m_j = \frac{1}{N}\sum_{i=1}^{N} \mathbf{q}_{ij}$ and variance s: $s_j = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{q}_{ij} - m_j)^2$, the KL-divergence is approximated with:

$$\mathrm{KL}(Y\|\mathcal{N}(0, I)) \approx -\frac{M}{2} + \sum_{j=1}^{M} \frac{s_j^2 + m_j^2}{2} - \log(s_j)\,. \quad (8)$$

We tried both the parametric version from above and the non-parametric version based on Kozachenko-Leonenko estimator (Kozachenko & Leonenko, 1987). Both versions worked equally well, and we used a simpler parametric estimator in the presented experiments.

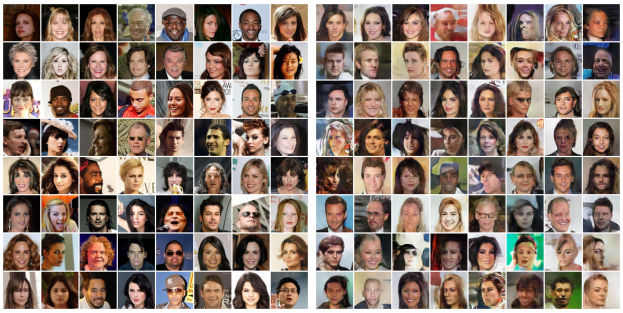
Controlling the training process: while training the models we ensure that the generator's latent distribution stays as close to uniform on the sphere as possible. It is easiest to examine component-wise mean and variance of $e(g(Z))$ for that purpose, as the mean should stay close to zero while variance should be approximately $1/\dim(\mathcal{Z})$.

Hyper-parameters: We use ADAM (Kingma & Ba, 2014) optimizer with the learning rate of 0.0002. We perform two generator updates per one encoder update for all datasets. For each dataset we tried $\lambda \in \{500, 1000, 2000\}$ and picked the best one. We ended up using $\mu = 10$ for all datasets. The dimensionality M of the latent space was manually set according to the complexity of the dataset. We thus used $M = 10$ for MNIST, $M = 64$ for CelebA and SVHN datasets, and $M = 128$ for the most complex datasets of Tiny ImageNet and CIFAR.



(a) Real images. (b) Ours (samples). (c) Ours reconstructions. (d) ALI reconstructions.

Figure 4. Samples (b) and reconstructions (c) for Tiny ImageNet dataset (top) and SVHN dataset (bottom). The results of ALI (Dumoulin et al., 2016) on the same datasets are shown in (d). In (c,d) odd columns show real examples and even columns show their reconstructions. Qualitatively, our method seems to obtain more accurate reconstructions than ALI (Dumoulin et al., 2016), especially on the Tiny ImageNet dataset, while having samples of similar visual quality.



(a) (b)

Figure 5. Real examples (a) and samples (b) from our model trained on CelebA dataset.

Figure 6. Latent space interpolation between two images from CelebA dataset. The original images are presented on the two sides.

Following the best practices[1] we construct different mini-batches for the real and fake data, which improved convergence.

Efficiency: For all datasets except Tiny ImageNet we train our models for about two hours on TITAN X Maxwell GPU while Tiny ImageNet model was trained for 6 hours. To the best of our knowledge it is considerably faster than (Dumoulin et al., 2016), which takes more than 8 hours to train on CelebA database.

3.2. Results

To evaluate our model, we provide a large number of samples for different datasets. All samples were drawn randomly (without replacement), while the results for (Dumoulin et al., 2016) are either reproduced with their code or copied from their paper.

We start by showing that the generated data distribution becomes aligned with the data distribution by using the proposed adversarial alignment procedure without the reconstruction losses. Figure 2 compares a DCGAN output (Radford et al., 2015)[2] to the proposed model trained

[1] https://github.com/soumith/ganhacks

[2] We used PyTorch implementation from https://github.com/pytorch/examples/tree/master/dcgan

Figure 7. Reconstruction quality comparison of our method with ALI (Dumoulin et al., 2016) and VAE (Kingma & Welling, 2013). The first column in each set shows examples from the test set of CelebA dataset. In the other columns the reconstructions for different methods are presented: column two for ours method, three for ALI and four for VAE.

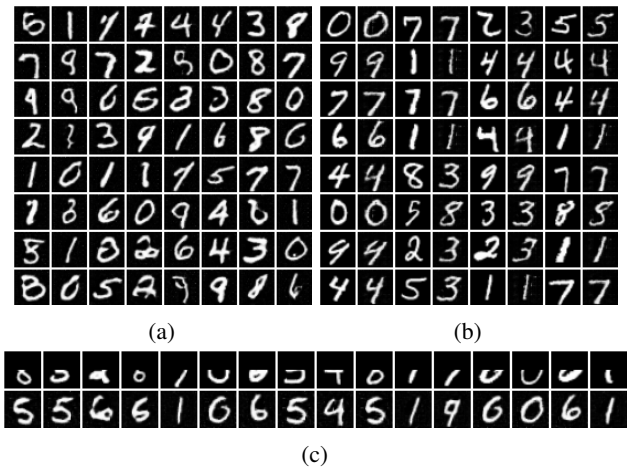


Figure 8. Samples (a) and reconstructions (b) for the MNIST dataset. In (b) odd columns show real examples and even columns show their reconstructions. In (c) we show a simple example of restoration with our model. The corrupted images (first row) are mapped to the latent space by the encoder and are then decoded with the generator (second row).

using adversarial distribution alignment loss alone. The samples from our model turn out to be diverse and visually resemble DCGAN's in quality and diversity. While the samples are comparable, our model can be augmented with the reconstruction enabling inference without harming the generation quality Figure 2(d).

We further test the reconstruction capabilities of the model on the standard datasets, presenting randomly sampled reconstruction examples alongside the results of (Dumoulin et al., 2016) for some of the datasets. The following datasets are considered: MNIST (LeCun & Cortes, 2010), CIFAR10 (Krizhevsky & Hinton, 2009), Tiny ImageNet (Russakovsky et al., 2015), CelebA faces (Liu et al., 2015), SVHN (Netzer et al., 2011) (Figures 3, 4, 8, 7, 5). Since our model uses the reconstruction loss in the image space we used holdout or test images for evaluation. We observe that our model captures color and shape, while preserving sharp edges. Similarly to (Dumoulin et al., 2016) the person identity may be completely changed when reconstructing CelebA samples, however overall our reconstructions seem to have higher fidelity to the input than in the case of (Dumoulin et al., 2016) across a number of datasets. For the MNIST dataset in Figure 8, we show a simple restoration example, which highlights the resilience of the encoder to severe degradations.

As discussed above, the generator in our method is never

supervised by the reconstruction loss (3), which is used solely for the encoder updates. To demonstrate the effect of the data reconstruction loss, we show the results obtained with a model trained with this loss being used by the generator updates Figure 9. As expected, the reconstructions become blurry, as if they were produced using (variational) autoencoder.

Our model can be further validated for overfitting by examining walks in its latent space. We followed a standard approach sampling pairs of examples $\mathbf{x}_1$, $\mathbf{x}_2$ from a CelebA holdout dataset and obtaining their latent representations $\mathbf{z}_1 = e(\mathbf{x}_1)$, $\mathbf{z}_2 = e(\mathbf{x}_2)$. We then move along the sphere interpolating between the latent codes and decoding the resulting representations with the generator. We observe (Figure 6) smooth interpolation with every intermediate image being plausible, which can serve as an indicator that the model does not suffer from strong overfitting to the train set.

Semi-supervised learning: finally, similarly to (Dumoulin et al., 2016; Donahue et al., 2016; Radford et al., 2015) we investigated whether the learned features are useful for discriminative tasks. We reproduced the evaluation pipeline from (Dumoulin et al., 2016) for SVHN dataset and obtained 23.7% error rate in the unsupervised feature learning protocol with our model, while their result is 19.14%. At the moment, it is unclear to us why AGE networks underperform ALI at this task.

4. Related work

As discussed in the introduction, joint learning of inference and generation is a hot topic of research. The methods that combine adversarial training with such simultaneous training, include approaches adversarially learned inference (adversarial feature learning) (Dumoulin et al., 2016; Donahue et al., 2016), the approach (Larsen et al., 2015), and adversarial autoencoders (Makhzani et al., 2015). Arguably, our approach has more similarity with (Dumoulin et al., 2016; Donahue et al., 2016), as in their case, the alignment involves the latent space distribution, and the generators are also involved into the adversarial learning game. Also, (Dumoulin et al., 2016) provides most extensive evaluation for the image sampling and reconstruction process, hence we make their method our main reference for comparisons.

Perhaps, the main novelty of our architecture is the new game objective for adversarial learning that is computed at batch level (rather than for individual samples). Recently, (Salimans et al., 2016) proposed to use batch-level information for adversarial training was first proposed in to prevent mode collapse. Their discriminator still performs individual sample classification, but admixes the descriptor based on distances to other samples in the batch into the individual sample feature descriptor. Interestingly, our method can also uses pairwise distances in a batch for the estimation of the divergences.

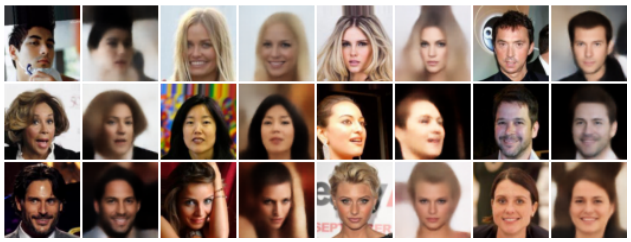


Figure 9. While we do not use data space reconstruction loss (3) to update generator during training, we found using it immediately leads to blurry, autoencoder-like results. These results are presented at this figure: odd columns correspond to real images and even to reconstructions.

Another avenue for improving the stability of GANs is the replacement of the classifying discriminator with the regression-based one as in energy-based GANs (Zhao et al., 2016) and Wasserstein GANs (Arjovsky et al., 2017). Our statistics (the divergence from the canonical distribution) can be seen as a very special form of regression. In this way, the encoder in our architecture can be seen as a discriminator computing a single number similarly to how it is done in (Zhao et al., 2016; Arjovsky et al., 2017).

5. Conclusion

We have introduced a new approach for simultaneous learning of generation and inference networks from unlabeled data. Crucially, we have demonstrated how to set up such learning as an adversarial game between generation and inference, which has a different type of objective from traditional GAN approaches. In particular the objective of the game considers divergences between distributions rather than discrimination at the level of individual samples. As a consequence, the game process is resilient to the collapse of the generator (although its adversary, the encoder, can collapse in some parts of the data space with low real data probability).

We demonstrate that on a range of standard datasets, the generators obtained by our approach provides high-quality samples, and that the reconstrunctions of real data samples passed subsequently through the encoder and the generator are of better fidelity than in (Dumoulin et al., 2016).

Our approach leaves a lot of room for further experiments. In particular, a more complex latent space distribution can be chosen as in (Makhzani et al., 2015), and other divergence measures between distributions can be easily tried.

References

Arjovsky, Martín, Chintala, Soumith, and Bottou, Léon. Wasserstein GAN. *CoRR*, abs/1701.07875, 2017.

Bengio, Yoshua. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

Donahue, Jeff, Krähenbühl, Philipp, and Darrell, Trevor. Adversarial feature learning. *CoRR*, abs/1605.09782, 2016.

Dumoulin, Vincent, Belghazi, Ishmael, Poole, Ben, Lamb, Alex, Arjovsky, Martín, Mastropietro, Olivier, and Courville, Aaron C. Adversarially learned inference. *CoRR*, abs/1606.00704, 2016.

Goodfellow, Ian J. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017. URL http://arxiv.org/abs/1701.00160.

Goodfellow, Ian J., Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron C., and Bengio, Yoshua. Generative adversarial nets. In *Proc. NIPS*, pp. 2672–2680, 2014.

Hornik, Kurt, Stinchcombe, Maxwell, and White, Halbert. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2 (5):359 – 366, 1989. ISSN 0893-6080. doi: http://dx.doi.org/10.1016/0893-6080(89)90020-8. URL http://www.sciencedirect.com/science/article/pii/0893608089900208.

Kingma, Diederik P. and Ba, Jimmy. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Kingma, Diederik P. and Welling, Max. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Kozachenko, L. F. and Leonenko, N. N. Sample estimate of the entropy of a random vector. *Probl. Inf. Transm.*, 23(1-2):95–101, 1987.

Krizhevsky, Alex and Hinton, Geoffrey. Learning multiple layers of features from tiny images. 2009.

Larsen, Anders Boesen Lindbo, Sønderby, Søren Kaae, and Winther, Ole. Autoencoding beyond pixels using a learned similarity metric. *CoRR*, abs/1512.09300, 2015.

LeCun, Yann and Cortes, Corinna. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Liu, Ziwei, Luo, Ping, Wang, Xiaogang, and Tang, Xiaoou. Deep learning face attributes in the wild. In *ICCV*, pp. 3730–3738. IEEE Computer Society, 2015.

Makhzani, Alireza, Shlens, Jonathon, Jaitly, Navdeep, and Goodfellow, Ian J. Adversarial autoencoders. *CoRR*, abs/1511.05644, 2015. URL http://arxiv.org/abs/1511.05644.

Marzouk, Youssef, Moselhy, Tarek, Parno, Matthew, and Spantini, Alessio. An introduction to sampling via measure transport. *arXiv preprint arXiv:1602.05023*, 2016.

Netzer, Yuval, Wang, Tao, Coates, Adam, Bissacco, Alessandro, Wu, Bo, and Ng, Andrew Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Owen, G. *Game Theory*. Academic Press, 1982. ISBN 9780125311502. URL https://books.google.ru/books?id=pusfAQAAIAAJ.

Radford, Alec, Metz, Luke, and Chintala, Soumith. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015.

Rezende, Danilo Jimenez, Mohamed, Shakir, and Wierstra, Daan. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael S., Berg, Alexander C., and Li, Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

Salimans, Tim, Goodfellow, Ian J., Zaremba, Wojciech, Cheung, Vicki, Radford, Alec, and Chen, Xi. Improved techniques for training gans. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 2226–2234, 2016.

Villani, Cédric. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Zhao, Junbo Jake, Mathieu, Michaël, and LeCun, Yann. Energy-based generative adversarial network. *CoRR*, abs/1609.03126, 2016. URL http://arxiv.org/abs/1609.03126.

Zhu, Jun-Yan, Krähenbühl, Philipp, Shechtman, Eli, and Efros, Alexei A. Generative visual manipulation on the natural image manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.

A. Appendix

Let X and Z be distributions defined in the data and the latent spaces $\mathcal{X}$, $\mathcal{Z}$ correspondingly. We assume X and Z are such, that there exists an invertible almost everywhere function e which transforms the latent distribution into the data one $g(Z) = X$. This assumption is weak, since for every atomless (i.e. no single point carries a positive mass) distributions X, Z such invertible function exists. For a detailed discussion on this topic please refer to (Villani, 2008; Marzouk et al., 2016). Since Z is up to our choice simply setting it to Gaussian distribution (for $\mathcal{Z} = \mathbb{R}^M$) or uniform on sphere for ($\mathcal{Z} = \mathbb{S}^M$) is good enough.

Lemma A.1. *Let X and Y to be two distributions defined in the same space. The distributions are equal $X = Y$ if and only if $e(X) = e(Y)$ holds for for any measurable function $e : \mathcal{X} \to \mathcal{Z}$.*

Proof. It is obvious, that if $X = Y$ then $e(X) = e(Y)$ for any measurable function e.

Now let $e(X) = e(Y)$ for any measurable e. To show that $X = Y$ we will assume converse: $X \neq Y$. Then there exists a set $B \in \mathcal{F}_X$, such that $0 < \mathbb{P}_X(B) \neq \mathbb{P}_Y(B)$ and a function e, such that corresponding set $C = e(B)$ has B as its preimage $B = e^{-1}(C)$. Then we have $\mathbb{P}_X(B) = \mathbb{P}_{e(X)}(C) = \mathbb{P}_{e(Y)}(C) = \mathbb{P}_Y(B)$, which contradicts with the previous assumption. □

Lemma A.2. *Let (g', e') and (g^*, e^*) to be two different Nash equilibria in a game $\min_g \max_e V(g, e)$. Then $V(g, e) = V(g', e')$.*

Proof. See chapter 2 of (Owen, 1982). □

Theorem 1. *For a game*

$$\max_e \min_g V_1(g, e) = \Delta(\, e(g(Z)) \| e(X)\,) \tag{9}$$

(g^*, e^*) is a saddle point of (9) if and only if g^* is such that $g^*(Z) = X$.

Proof. First note that $V_1(g, e) \geq 0$. Consider g such that $g(Z) = X$, then for any e: $V_1(g, e) = 0$. We conclude that (g, e) is a saddle point since $V_1(g, e) = 0$ is a maximum over e and minimum over g.

Using lemma A.2 for saddle point (g^*, e^*): $V_1(g^*, e^*) = 0 = \max_e V_1(g^*, e)$, which is only possible if for all e: $V_1(g^*, e) = 0$ from which immediately follows $g(Z) = X$ by lemma A.1. □

Lemma A.3. *Let function $e : \mathcal{X} \to \mathcal{Z}$ be X-almost everywhere invertible, i.e. $\exists e^{-1} : \mathbb{P}_X(\{\mathbf{x} \neq e^{-1}(e(\mathbf{x}))\}) = 0$. Then if for a mapping $g : \mathcal{Z} \to \mathcal{X}$ holds $e(g(Z)) = e(X)$, then $g(Z) = X$.*

Proof. From definition of X-almost everywhere invertibility follows $\mathbb{P}_X(A) = \mathbb{P}_X(e^{-1}(e(A)))$ for any set $A \in \mathcal{F}_X$. Then:

$$\mathbb{P}_X(A) = \mathbb{P}_X(e^{-1}(e(A))) = \mathbb{P}_{e(X)}(e(A)) = \\ = \mathbb{P}_{e(g(Z))}(e(A)) = \mathbb{P}_{g(Z)}(A).$$

Comparing the expressions on the sides we conclude $g(Z) = X$.

□

Theorem 2. *Let Y to be any fixed distribution in the latent space. Consider a game*

$$\max_e \min_g V_2(g, e) = \Delta(e(g(Z)) \| Y) - \Delta(e(X) \| Y)\,. \tag{10}$$

If the pair (g^*, e^*) is a Nash equilibrium of game (10) then $g^*(Z) \sim X$. Conversely, if the fake and real distributions are aligned $g^*(Z) \sim X$ then (g^*, e^*) is a saddle point for some e^*.

Proof.

- As for a generator which aligns distributions $g(Z) = X$: $V_2(g, e) = 0$ for any e we conclude by A.2 that the optimal game value is $V_2(g^*, e^*) = 0$. For an optimal pair (g^*, e^*) and arbitrary e' from the definition of equilibrium:

$$0 = \Delta(e^*(g^*(Z)) \| Y) - \Delta(e^*(X) \| Y) \geq \\ \geq \Delta(e'(g^*(Z)) \| Y) - \Delta(e'(X) \| Y)\,. \tag{11}$$

  For invertible almost everywhere encoder e' such that $\Delta(e'(X) \| Y) = 0$ the first term is zero $\Delta(e'(g^*(Z)) \| Y) = 0$ since inequality (11) and then $e'(g^*(Z)) = e'(X) = Y$. Using result of the lemma A.3 we conclude, that $g^*(Z) = X$.

- If $g^*(Z) = X$ then $\forall e : e(g^*(Z)) = e(X)$ and $V_2(g^*, e^*) = V_2(g^*, e) = 0 = \max_{e'} V_2(g^*, e')$.

  The corresponding optimal encoder e^* is such that $g^* \in \arg\min_g \Delta(e^*(g(Z)) \| Y)$.

□

Note that not for every optimal encoder e^* the distributions $e^*(X)$ and $e^*(g^*(Z))$ are aligned with Y. For example if e^* collapses $\mathcal{X}$ into two points then for any distribution X: $e^*(X) = e^*(g^*(Z)) = Bernoulli(p)$. For the optimal generator g^* the parameter p is such, that for all other generators g' such that $e^*(g'(Z)) \sim Bernoulli(p')$: $\Delta(e^*(g^*(Z)) \| Y) \leq \Delta(e^*(g'(Z)) \| Y)$.

Theorem 3. *Let the two distributions W and Q be aligned by the mapping f (i.e. $f(W) = Q$) and let $\mathbb{E}_{\mathbf{w}\sim W}\|\mathbf{w} - h\big(f(\mathbf{w})\big)\|^2 = 0$. Then, for $\mathbf{w} \sim W$ and $\mathbf{q} \sim Q$, we have $\mathbf{w} = h(f(\mathbf{w}))$ and $\mathbf{q} = f(h(\mathbf{q}))$ almost certainly, i.e. the mappings f and h invert each other almost everywhere on the supports of W and Q. More, Q is aligned with W by h: $h(Q) = W$.*

Proof. Since $\mathbb{E}_{\mathbf{w}\sim W}\|\mathbf{w} - h\big(f(\mathbf{w})\big)\|^2 = 0$, we have $\mathbf{w} = h(f(\mathbf{w}))$ almost certainly for $\mathbf{w} \sim W$. We can substitute $h(f(\mathbf{w}))$ with $\mathbf{w}$ under an expectation over W. Using this and the fact that $f(\mathbf{w}) \sim Q$ for $\mathbf{w} \sim W$ we derive:

$$\mathbb{E}_{\mathbf{q}\sim Q}\|\mathbf{q} - f\big(h(\mathbf{q})\big)\|^2 = \mathbb{E}_{\mathbf{w}\sim W}\|f(\mathbf{w}) - f\big(h(f(\mathbf{w}))\big)\|^2 = \\ = \mathbb{E}_{\mathbf{w}\sim W}\|f(\mathbf{w}) - f(\mathbf{w})\|^2 = 0\,.$$

Thus $\mathbf{q} = f(h(\mathbf{q}))$ almost certainly for $\mathbf{q} \sim Q$.

To show alignment $h(Q) = W$ first recall the definition of alignment. Distributions are aligned $f(W) = Q$ iff $\forall \bar{Q} \in \mathcal{F}_Q$: $\mathbb{P}_Q(\bar{Q}) = \mathbb{P}_W(f^{-1}(\bar{Q}))$. Then $\forall \bar{W} \in \mathcal{F}_W$ we have

$$\mathbb{P}_W(\bar{W}) = \mathbb{P}_W(h(f(\bar{W}))) = \mathbb{P}_W(f^{-1}(f(\bar{W}))) = \\ = \mathbb{P}_Q(f(\bar{W})) = \mathbb{P}_Q(h^{-1}(\bar{W}))\,.$$

Comparing the expressions on the sides we conclude $h(Q) = W$. □