Binary seeds for generative adversarial networks

Arnaud Sors*[1, 2], Stéphane Bonnet[1, 2], Laurent Vercueil[1, 4], and Jean-François Payen[1, 3]

[1]Univ. Grenoble Alpes, F-38000 Grenoble, France
[2]CEA Leti, MINATEC Campus, 17 rue des Martyrs, F-38054 Grenoble, France
[3]Grenoble University Hospital, Dpt. Anesthesia and Critical Care, Avenue Maquis du Grésivaudan, F-38700 La Tronche, France
[4]Grenoble University Hospital, Dpt. Exploration Fonctionnelle du Système Nerveux, Avenue du Maquis du Grésivaudan, F-38700 La Tronche, France

July 6, 2017

Abstract

Generative Adversarial Networks (GANs) are recent models for learning mappings between continuous data and latent variable representations of this data. GAN models typically use continuous distributions in the latent space. In this work we investigate the use of binary variables in the latent space in place or in addition to continuous variables, and show that they yield comparable or slightly better samples. We evaluate different combinations of binary and continuous variables for the generator seed using gradient-penalized Wasserstein GANs.

Keywords: Generative Adversarial Networks, Wasserstein-GAN, binary seed, binary latent variable.

1 Introduction

Generative models are models for generating *unseen data* that is similar to observed data. They are in contrast to discriminative models where the aim is to model target predictions given observed data. Generative modeling is studied from the perspective that a model that *generates realistic data* is a model that *understands the properties of this data*. In practice, generative models map samples from an imposed latent distribution to samples in the data space. Aside generation, an associated *inference model* can be used for recovering a latent space value from an observed data sample. Even for applications which involve *making predictions*, learning a unsupervised generation and inference model on data can be helpful, for example in the context of semi-supervised learning.

Generative Adversarial Networks (GANs) are a class of generative models that involve using two competing neural networks trained jointly in a minimax game. Contrary to other generative models such as Boltzmann machines [Hin10], GAN training does not require any sampling-based estimation of a partition function and can be trained by backpropagation only. GANs have received a great deal of attention recently.

GAN models typically use continuous distributions in the latent space, such as a multivariate normal or uniform distribution. In this paper we study the use of binary variables in the latent space, instead of continuous variables. Informally, the intuition behind this idea is that with latent space dimensions commonly used in GANs, *a multivariate binary distribution may already have enough 'possible combinations' to map data diversity*, and finer-grained distributions such as a continuous distribution may not be needed. Here we test this hypothesis using the converged critic loss of a Wasserstein GAN (WGAN) [ACB17] as a measure of generated samples quality. Our contribution is to *show experimentally that binary seeds yield as good samples as continuous seeds*, and propose a short examplification of how using a seed with both a binary and a continuous part can help generate data in semantically significant groups.

2 Related work

GANs theory and training tricks: This paper is related to the following pieces of work: [GPAM+14] devise

*arnaud.sors@cea.fr

Figure 1: Example MNIST digits generated from noise using the architecture described in 4 with $n_c = 7$ and $n_b = 12$.

the original GANs. [RMC15] strongly popularize them by suggesting architectural choices that stabilize the training. [MPPSD16] improve GAN stability and mode dropping by unrolling the discriminator. [MLX+16] suggest that a least squares loss in GANs avoids saturations observed with a sidmoid cross-entropy loss and yields better samples. [AB17] identify a number of theoretical reasons explaining why f-divergences are flawed for GAN-training, and suggest instance noise in data space as a fix. [ACB17] suggest a new measure between data and generated distributions along with a new training procedure based on this distance: Wasserstein GANs (WGANs). Finally, with recent gradient-penalized WGAN (GP-WGAN), [GAA+17] suggest a gradient norm penalty for faster convergence and greatly improved stability of WGANs.

Inference: [DBP+16] and [DKD16] concurrently propose similar frameworks for learning the generative network and the inverse mapping from data back into the latent space.

Binary representations: To our knowledge, the only paper considering binary seeds in GANs is InfoGAN [CDH+16]. A seed with a binary and a continuous part is used. A measure of mutual information between a subset of the latent variables and the generated samples - in the form of a reconstruction cost in latent space formulated with variational inference - is maximized during training.

3 Methods

3.1 Gradient-penalized Wasserstein GANs

We consider the problem of training a *generator model* G that takes samples from a latent space distribution and maps them to the data space, so that the distribution of generated samples matches the data distribution. In practice the model is a deep neural network. GANs in their original formulation propose to jointly train this generator network and a discriminator network. The discriminator is trained to *distinguish data samples from generated samples*, and the generator is trained to *fool the discriminator*, i.e. generate better samples. This minimax problem is a *proxy* to the problem of matching distributions. However, the equivalence is pathological [AB17] and training is particularly unstable in practice. Wasserstein GANs [ACB17] propose to avoid this proxy problem and *explicitly* minimize a distance measure between the data distribution and the generated distribution instead.

Let $\mathbb{P}_r$ denote the data distribution and $\mathbb{P}_g$ denote the distribution of generated samples $\mathbf{x}_g$:

$$\mathbf{x}_g = G(\mathbf{z}), \quad \mathbf{z} \sim p(\mathbf{z})$$

where p is a prior (noise) distribution on the latent space. The *Wasserstein distance* - also called *Earth Mover's distance* - between $\mathbb{P}_r$ and $\mathbb{P}_g$ is the *minimal mass transport* to transform the distribution $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x} - \mathbf{y}\|] \qquad (1)$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all possible transport plans that turn $\mathbb{P}_r$ into $\mathbb{P}_g$. In high dimensional spaces, it is computationally not feasible to estimate this Wasserstein distance, because the infinimum is not tractable. However, the Kantorovich-Rubinstein theorem[1][Vil08] turns this infinimum estimation problem into a maximum estimation problem in the dual space:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{f \in \mathcal{F}} \quad \mathbb{E}_{\mathbf{x}_r\sim\mathbb{P}_r}[f(\mathbf{x}_r)] - E_{\mathbf{x}_g\sim\mathbb{P}_g}[f(\mathbf{x}_g)] \qquad (2)$$

[1] Cedric Villani's book *Optimal transport, old and new*[Vil08], pages 63-66 has a nice examplification of the Kantorovich Rubinstein theorem with a bread distribution problem where bakeries produce bread and cafés sell it

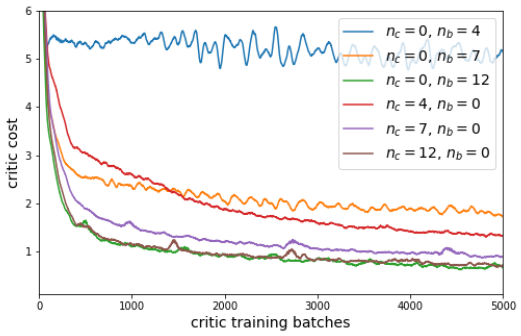

Figure 2: Evolution of the critic loss over training for a few selected seed configurations, described in 4.1. For this visualization, the loss is smoothed using a Hanning kernel of width 50 steps.

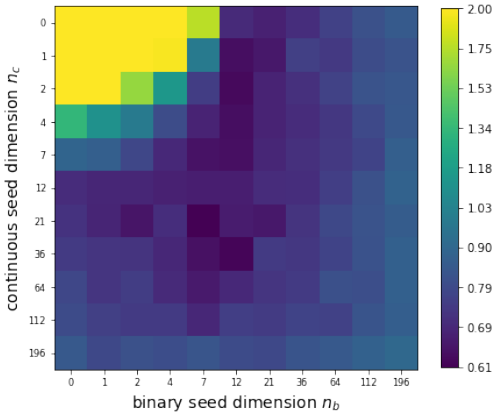

Figure 3: Graphical summary of the mean values of critic losses averaged over the last 200 critic training steps, after 9 epochs of training and for various seed configurations. Recall that under the GP-WGAN theory, critic loss is a measure of sample quality.

where $\mathcal{F}$ denotes the set of all 1-Lipschitz functions from data space to $\mathbb{R}$. The idea of Wasserstein GANs is to *use a critic deep neural network C to approximate f* and train it by gradient descent to estimate the supremum. Different methods exist for constraining C to be 1-Lipschitz. The original method of clipping its weights [ACB17] has caveats, and it was very recently proposed that penalizing the squared distance to unity of the *gradient of the critic with respect to its input* is a more desirable constraint [GAA+17]. The objective *for the critic* now takes the form:

$$\begin{aligned} \min_C \quad & \Big(\quad \mathbb{E}_{\mathbf{x}_g \sim \mathbb{P}_g}[C(\mathbf{x}_g)] - \mathbb{E}_{\mathbf{x}_r \sim \mathbb{P}_r}[C(\mathbf{x}_r)] \\ & + \lambda \quad \mathbb{E}_{\tilde{\mathbf{x}} \sim \mathbb{P}_{\tilde{\mathbf{x}}}}[(\|\nabla_{\tilde{\mathbf{x}}} C(\tilde{\mathbf{x}})\|_2 - 1)^2] \quad \Big) \\ & = \min_C \quad \big(L_1 + \lambda L_2\big) \end{aligned} \tag{3}$$

where the $\tilde{\mathbf{x}}$ used for evaluating gradients of the critic with respect to its input are sampled along straight lines between data samples and generated samples:

$$\epsilon \sim U[0,1], \quad \mathbf{x}_r \sim \mathbb{P}_r, \quad \mathbf{x}_g \sim \mathbb{P}_g$$
$$\tilde{\mathbf{x}} = \epsilon \mathbf{x}_r + (1-\epsilon)\mathbf{x}_g$$

The minimization in equation 3 has the effect that the 'difference' part L_1 becomes an estimate of the Wasserstein distance between $\mathbb{P}_r$ and the current $\mathbb{P}_g$. This estimate is differentiable with respect to neural network parameters. Therefore, it can be used for training the generator: G is trained to generate samples that are close in terms of Wasserstein distance to data samples. In other words, G is trained to maximize L_1 in equation 3 - or maximize $\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})}[C(G(\mathbf{z})]$.

The generator and critic are trained jointly, but for the validity of the Wasserstein distance estimate it is important that the critic remains optimal at all times. In practice this is achieved by training the critic more steps between each generator step. While originating from a different theoretical grounding, a WGAN is very similar in form and training procedure to an original GAN. The only difference is the absence of a sigmoid squashing function at the output of the critic, and the 1-Lispschitz penalization term.

Finally, a property of Wasserstein GANs that original GANs do not have is that *the critic loss is a measure of quality of the generation process*, as long as the critic is kept optimal. Using the same critic architecture, we exploit this property for comparing different choices of generator seeds.

3.2 Binary seeds

To our knowledge, most GANs and WGANs except InfoGAN [CDH+16] use continuous seeds $\mathbf{z}$ in the latent space. Typically, $\mathbf{z} \sim p(\mathbf{z})$ where p can be a uniform distribution. In this work we simply consider the use of binary seeds instead of or in addition to continuous seeds. The intuition behind the use of binary seeds for the generator is that while the data might *contain modes*, it may be disadvantageous to try to model it

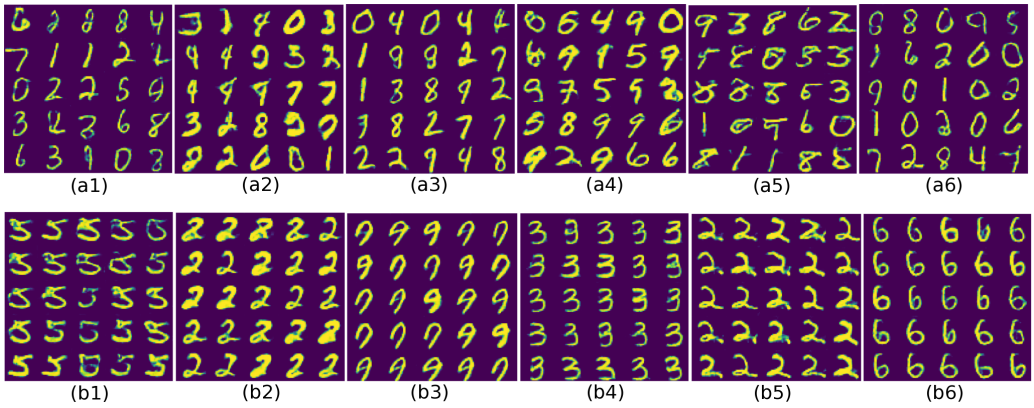

Figure 4: Visualization of generated digits. In each group (a1) to (a6), the continuous part $\mathbf{z}_c$ of the seed is fixed. In each group (b1) to (b6), the binary part $\mathbf{z}_b$ of the seed is fixed. We observe that $\mathbf{z}_b$ mostly codes for shape and $\mathbf{z}_c$ mostly codes for style. Examples and groups are chosen randomly and not cherry-picked.

with a generator whose input distribution is entirely continuous. For example, even the fact that a dataset *contains labels* hints at the fact that the discribution of its data is probably not equally spread but contains modes.

Let $\mathbf{z} = [\mathbf{z}_c, \mathbf{z}_b]$ denote a generator seed, where $\mathbf{z}_c$ is a continuous seed drawn from a normal distribution and $\mathbf{z}_b$ is a binary seed drawn from a Bernoulli distribution with parameter 0.5. Let also n_c and n_b denote the dimensions of respectively $\mathbf{z}_c$ and $\mathbf{z}_b$. We use this mixed seed $\mathbf{z}$ as generator seed in a GP-WGAN. This is the only modification we consider.

4 Experiments

In the following experiments we train WGANs on the task of generating MNIST digits [LCB98]. MNIST is a simple and well explored dataset which is appropriate for running a large hyperparameter search with modest hardware. We tried a range of fully-connected and convolutional architectures and note that with GP-WGANs training is stable regardless of the architectural choices, contrary to other types of GANs. The following results are obtained with a standard convolutional architecture: for the generator the architecture is: *(6272-fc)-(128-reshape)-(64-conv-5-2)-(1-conv-5-2)*, where *(k-fc)* denotes a dense layer with k output units, *(k-reshape)* reshapes to k square feature maps and *(i-conv-j-k)* designates a 2D convolution layer with i output feature maps, convolution kernels of size [j, j], and strides [k, k]. All convolutions use a 'same' padding. All nonlinearities are leaky rectified linear units (with slope 0.1) except a sigmoid for the last layer. The critic has the same (transposed) architecture except that there is no activation function on the output. For GP-WGAN we use $\lambda = 10$ and 6 critic iterations per generator iteration, and occasionally run 100 critic iterations to make sure the critic has remained optimal. Also, in all experiments we use the Adam [KB14] optimizer with learning rate $1e-3$ and a batch size of 128.

4.1 Hyperparameter search

The first experiment is a hyperparameter search over different combinations (n_c, n_b), where n_c and n_b take all values among (0, 2, 4, 7, 12, 21, 36, 64, 112, 196). Figure 2 shows the evolution of the critic loss over training, for a few seed configurations. Because the same critic architecture is used for all generators and because the critic remains optimal, the critic loss can be used as a measure of generation performance. The absence of overfitting was monitored on the test set using the same critic loss. We note that all configurations roughly follow the same training dynamics, except the configuration $n_c = 4$, $n_b = 0$ which trains slowlier. We hypothesize that in with smaller $\mathbf{z}_c$ dimensions and continuous seeds, the units of $\mathbf{z}$ are used in a less independent way, i.e. each unit can 'encode for more than one variation of the data'. This behavior is not observed for larger seed continuous seeds, and impossible with binary seeds.

Figure 3 shows a graphical summary of the critic loss - averaged over the last 200 batches - for the different seed configurations after 9 epochs of training. We observe that the best critic loss for *all-binary* configurations is obtained for a same seed dimension as

the best critic loss for *all-continuous* configurations: $n_c = n_b = 12$. Also, the associated *all-binary* critic loss is comparable or slightly better than the *all-continuous one*. Finally, combining a binary part and a continuous part in the seed yields the best samples, increasing sample quality by a small margin compared to the the best *all-binary* or the best *all-continuous* configurations.

4.2 Visualizations

As a second exploratory experiment we choose a *mixed configuration* where $\mathbf{z}$ has a continuous and a binary part. We choose a configuration that has good generation performance according to the hyperparameter search described in 4.1 - $n_c = 7$ and $n_b = 12$ - and generate samples. Figure 4 shows generations from all-random seeds, generations with fixed binary and variable continuous seeds, and generation with fixed continuous and variable binary seeds. For this configuration of seed dimensions, we observe that the network learns that the most efficient way to generate faithful data is to have the binary seed encode shape diversity and the continuous seed encode style diversity. This result compares to the results of InfoGAN [CDH$^+$16] except that no prior on the number of classes was used, and no reconstruction cost in latent space or mutual information measure between seed and data was used.

5 Conclusion

In this short paper we explored the use of binary variables instead of or additionally to continuous variables for GAN seeds. We showed that binary variables can yield generations of comparable quality to continuous variables for a similar seed dimension, and that combining continuous and binary variables can provide an original way of unentangling style and shape for MNIST digit generations. As a perspective, the use of *multistep* discrete variables could also be explored. Also, although we only studied generation here, we also conjecture that such binary representations can be useful when an encoder is used for semi-supervised learning, or as compact representations for data compression.

References

[AB17] Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. In *NIPS 2016 Workshop on Adversarial Training. In review for ICLR*, volume 2016, 2017.

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

[CDH$^+$16] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016.

[DBP$^+$16] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[DKD16] Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. *arXiv preprint arXiv:1605.09782*, 2016.

[GAA$^+$17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017.

[GPAM$^+$14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[Hin10] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Momentum*, 9(1):926, 2010.

[KB14] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[LCB98] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[MLX$^+$16] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, and Zhen Wang. Least squares generative adversarial networks. *arXiv preprint ArXiv:1611.04076*, 2016.

[MPPSD16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

[RMC15] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[Vil08] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.