Generative Adversarial Networks with Constraints

Short review of the work:

Yang, Z., Wu, J. L., Xiao, H. (2019). "Enforcing deterministic constraints on generative adversarial networks for emulating physical systems." arXiv preprint arXiv:1911.06671.

A.Demichev

April 2022

Общие замечания

- Пропускаем вводные обсуждения и предварительную информацию
- начинаем со стр. 8
- Разд. 2.2. Representation of generic physical constraints

Representation of generic physical constraints

- In view of the diverse examples physical laws enumerated above, we write the constraints in a generic form as follows:

$$\mathcal{H}[\mathbf{u}] \leq 0\,, \tag{0.1}$$

- $\mathcal{H}$ denotes an algebraic operator, which can involve nonlinearity and can be obtained from numerical discretization of the integro-differential operators in the original physics constraints.
- Typically, $\mathcal{H}$ is chosen to be a non-negative function such as norm, so requiring $\mathcal{H}[\mathbf{u}] \leq 0$ would effectively lead to $\mathcal{H}[\mathbf{u}] = 0$.
- As an example, the linear equation $\mathrm{N}\mathbf{u} = \mathbf{f}$ above can be cast into an inequality as $\|\mathrm{N}\mathbf{u} - \mathbf{f}\|^2 \leq 0$, where $\|\cdot\|$ indicates the Euclidean vector norm.
- кажется, что можно использовать и непосредственно равенство, но см. ниже - “the imprecise constraint”

Embedding constraints into the generator of GANs

- add a penalty term to the loss function of the generator based on the generic representation of constraints in the cGAN loss function:

$$V_C(\boldsymbol{D}, \boldsymbol{G}) = V(\boldsymbol{D}, \boldsymbol{G}) + \lambda C_{phys} \tag{0.2}$$

$$\text{with} \quad C_{phys} = \mathbb{E}_{Z \sim p_z(Z)} \left[\max \left(\mathcal{H}(\boldsymbol{G}(Z)),\ 0 \right) \right], \tag{0.3}$$

- λ denotes the penalty coefficient,
- V is the loss function of the baseline GANs to be constrained.
- As the physical constraint term is independent of the discriminator D, its gradient is non-zero even if the discriminator D is close to optimum.
- adding physical constraints to the generator loss function provides an alternative approach for combating the vanishing gradient problem in the training of GANs.

Implementation considerations

- The physics-constrained GANs are implemented based on the open-source software library TensorFlow.
- implementation is built upon the standard, baseline GANs implementation by Kristiadi et al. https://github.com/wiseodd/generative-models/tree/master/GAN.
- The source code for the physics-constrained GANs and the example cases presented below are publicly available in a GitHub repository https://github.com/zengyang7/ConstrainedGANs.git

Implementation considerations: part of the code

```
# WGAN

D_loss = tf.reduce_mean(D_real) - tf.reduce_mean(D_fake)

G_loss = -tf.reduce_mean(D_fake)+lam_constraint*tf.reduce_mean(penalty_log)

D_solver = (tf.train.RMSPropOptimizer(learning_rate=1e-4).minimize(-D_loss,
var_list=theta_D))

G_solver = (tf.train.RMSPropOptimizer(learning_rate=1e-4).minimize(G_loss,
var_list=theta_G))

clip_D = [p.assign(tf.clip_by_value(p, -0.01, 0.01)) for p in theta_D]
```

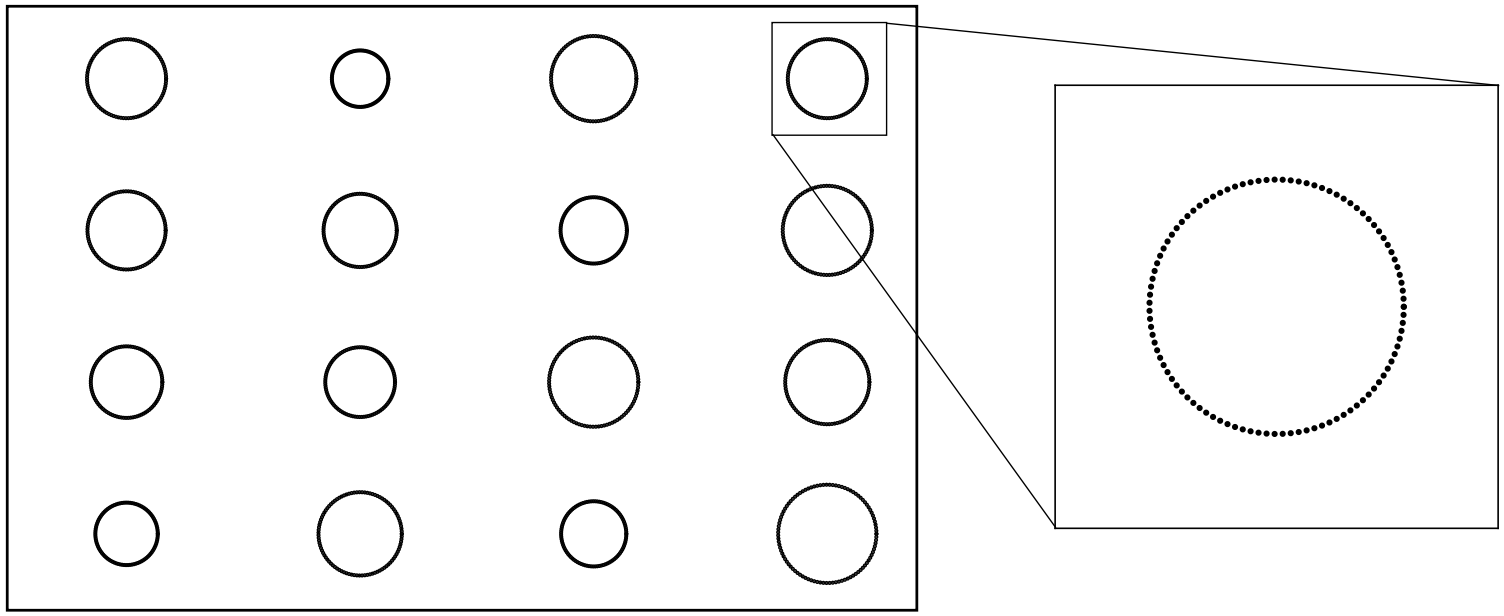
Geometrical constraint: generating circles. Problem description

- The training samples are the circles with different radii. The parametric constrained function of circles is given as

$$\begin{cases} x = r\cos(\psi) \\ y = r\sin(\psi) \end{cases} \tag{0.4}$$

- $[x^{(i)}, y^{(i)}]^\top$ denotes the points on the circle in Cartesian coordinates, where $i \in \{1, 2, ..., 100\}$ is the index of point of a generated sample.
- The training samples consist of $N = 5000$ circles whose radius is drawn from a uniform distribution between 0.4 and 0.8.
- The goal is to first train GANs with the training dataset of circles and then use the trained GANs to generate samples.
- train **standard GANs, standard cGANs, and constrained cGANs** proposed here
  - нужно ли cGANs, если constrained

Representative examples from the training samples



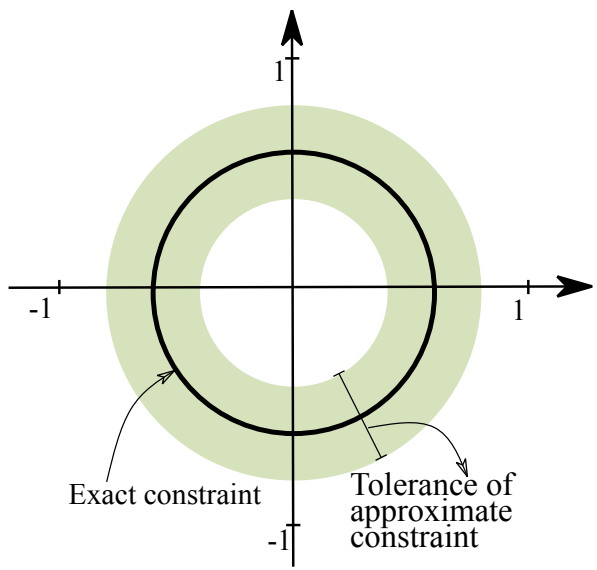
- Each sample consists of 100 points as is evident from the zoom-in view in the right panel (albeit appearing as solid lines above due to compact spacing in the left panel).
- These points are evenly distributed along the circumferential direction of a circle.
- The radius r of each circle is randomly drawn from a uniform distribution in the range $[0.4, 0.8]$.

The imprecise constraint

- The imprecise constraint is defined as follow:

$$C^{approx} = \mathbb{E}_{Z \sim p_z(Z)} \left[\sum_i \max \left(\left(\|\mathbf{x}^{(i)}\| - \zeta \right)^2, \varepsilon^2 \right) \right], \quad (0.5)$$

- $\|\cdot\|$ indicate Euclidean norm, ζ denotes the specified radius, and ε denotes the tolerance of the specified constraint.
- The imprecise constraint is schematically illustrated:



The metrics

- three metrics are used to evaluate the generated samples
  - 1st: the magnitude of the original loss function
  - 2d: the deviation of generated samples from the circle with an optimal radius $\bar{r}$ obtained from least-square fitting:

$$\mathcal{E}_{\text{dev}} = \frac{1}{n}\sum_{i=1}^{n}\left|(x_i^2 + y_i^2) - \bar{r}^2\right| \tag{0.6}$$

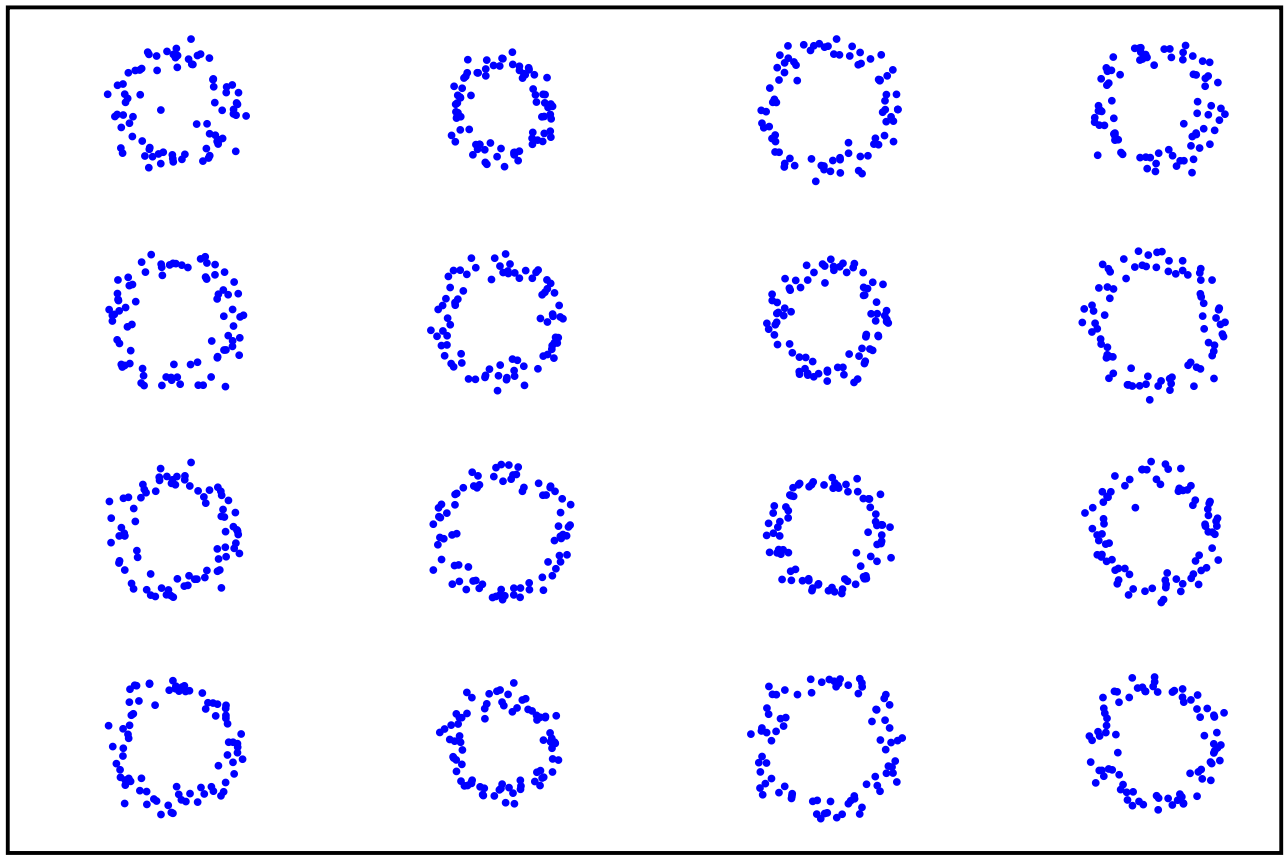
  - 3d: the deviation from the specified radius:

$$\mathcal{E}_{\text{bias}} = \left|\bar{r} - \zeta\right|, \tag{0.7}$$

- $\langle\cdot\rangle$ below indicates ensemble averaging

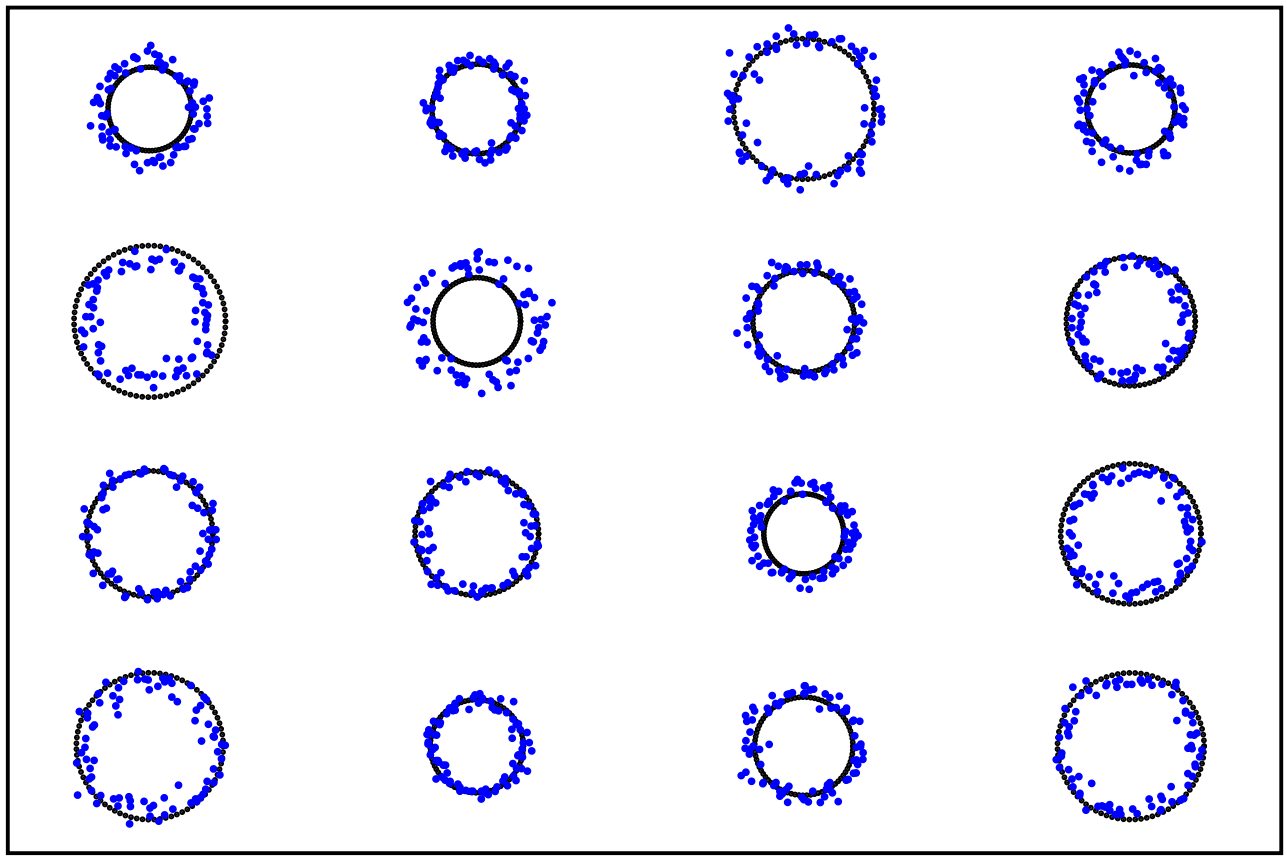
Results for $\varepsilon = 0$ (1)

Standard GANs



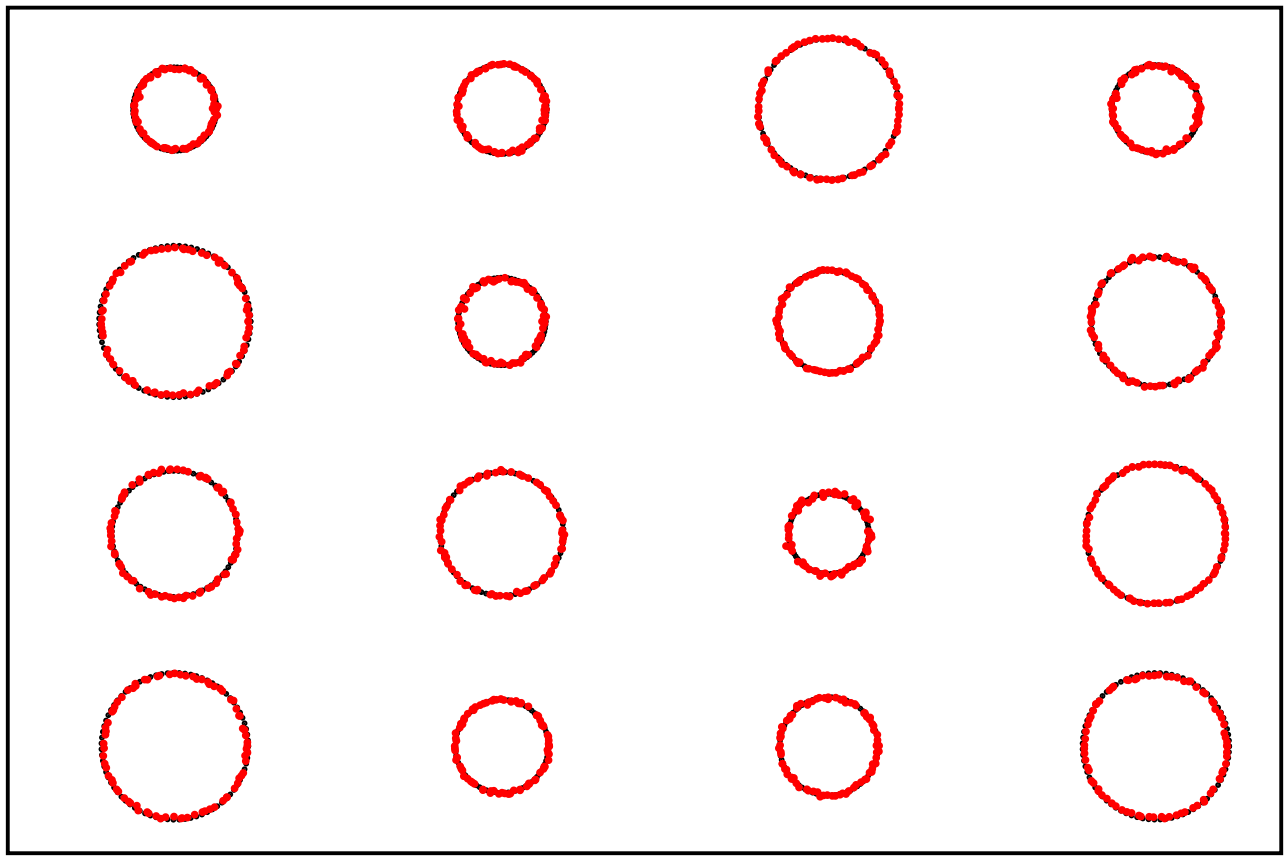
Results for $\varepsilon = 0$ (2)

Standard cGANs



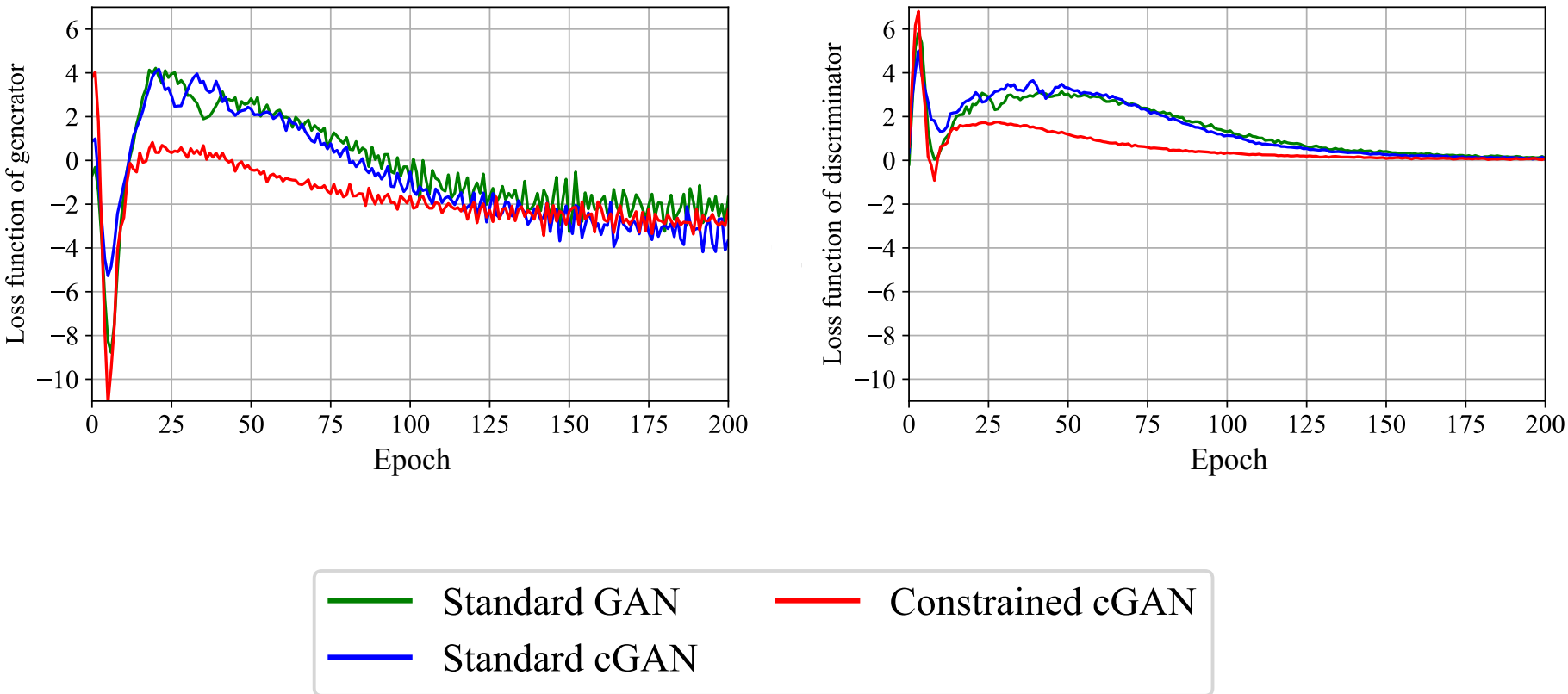
Results for $\varepsilon = 0$ (3)

Constrained cGANs



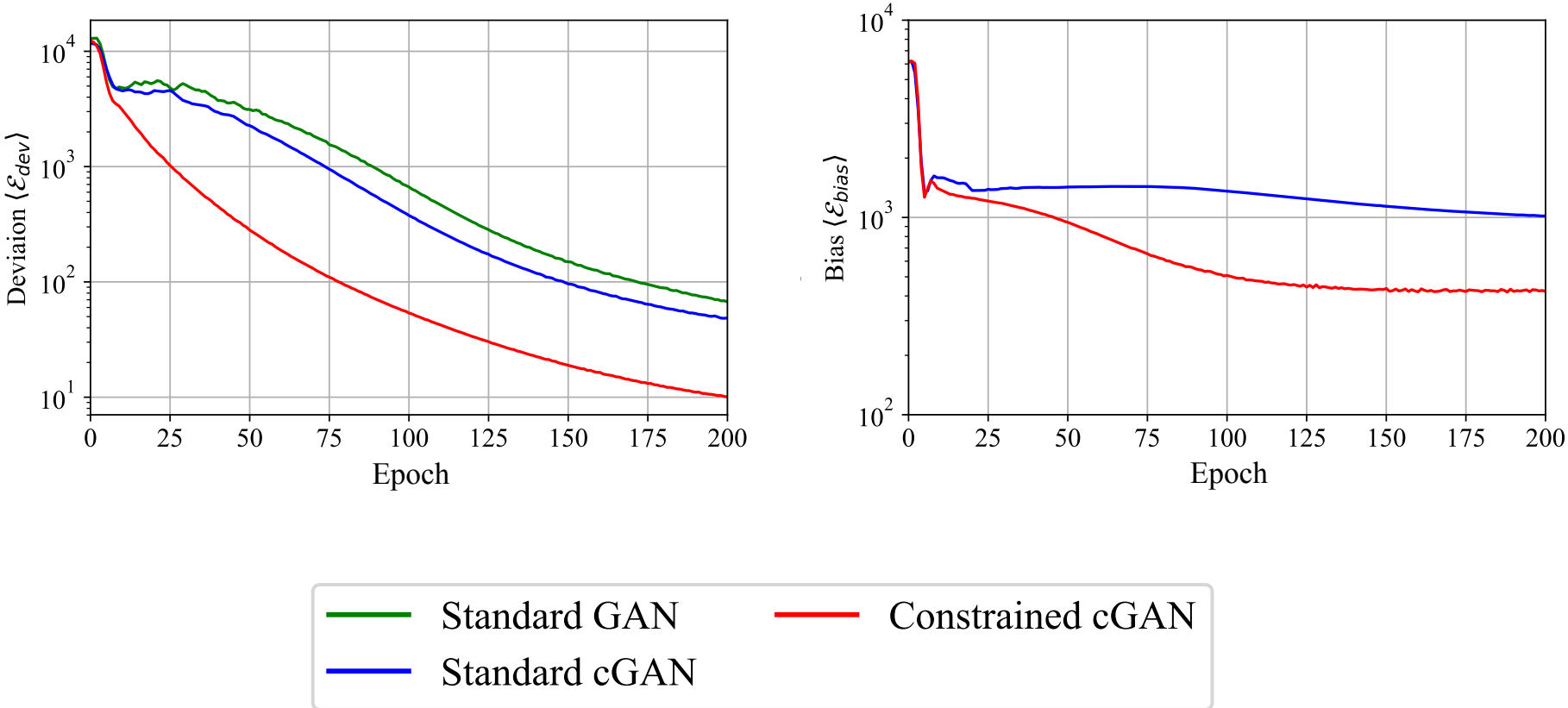
Results for $\varepsilon = 0$ (4): loss function

The generator (left) and the discriminator (right)



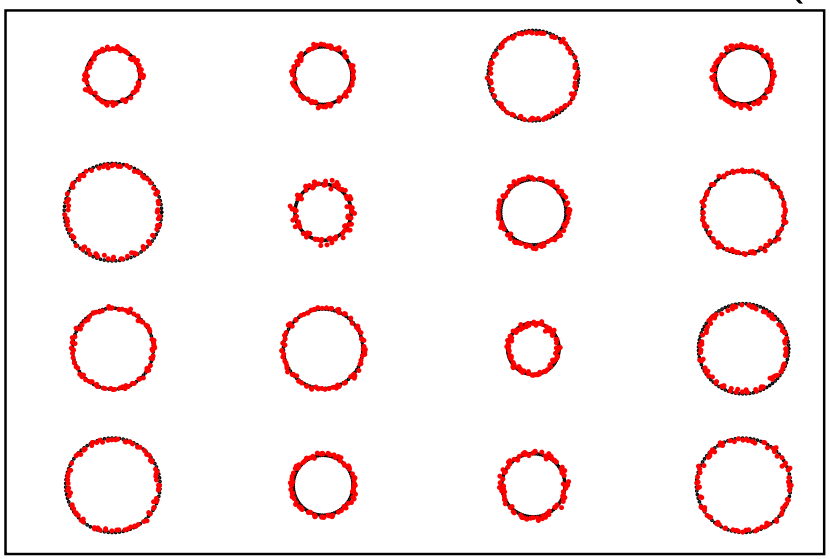
Results for $\varepsilon = 0$ (5)

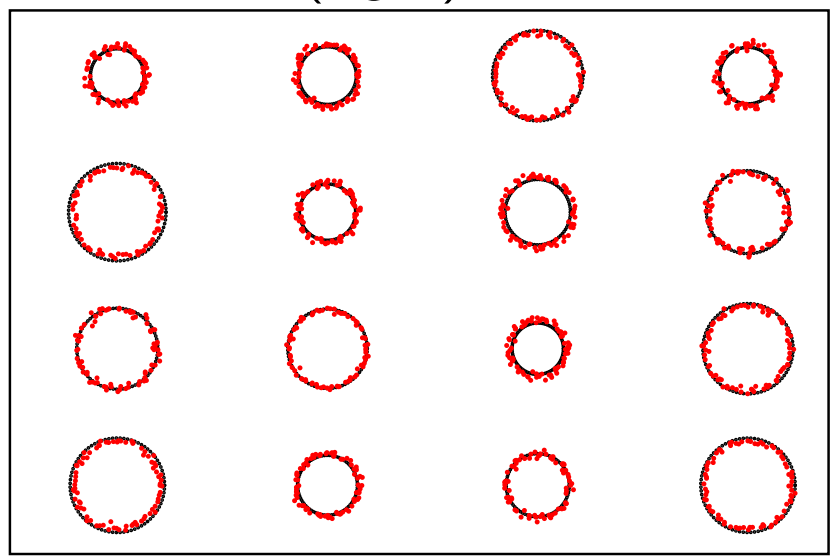
Deviations $\langle \mathcal{E}_{\text{dev}} \rangle$ (left) and Bias $\langle \mathcal{E}_{\text{bias}} \rangle$ (right)



Results for $\varepsilon \neq 0$ (1)

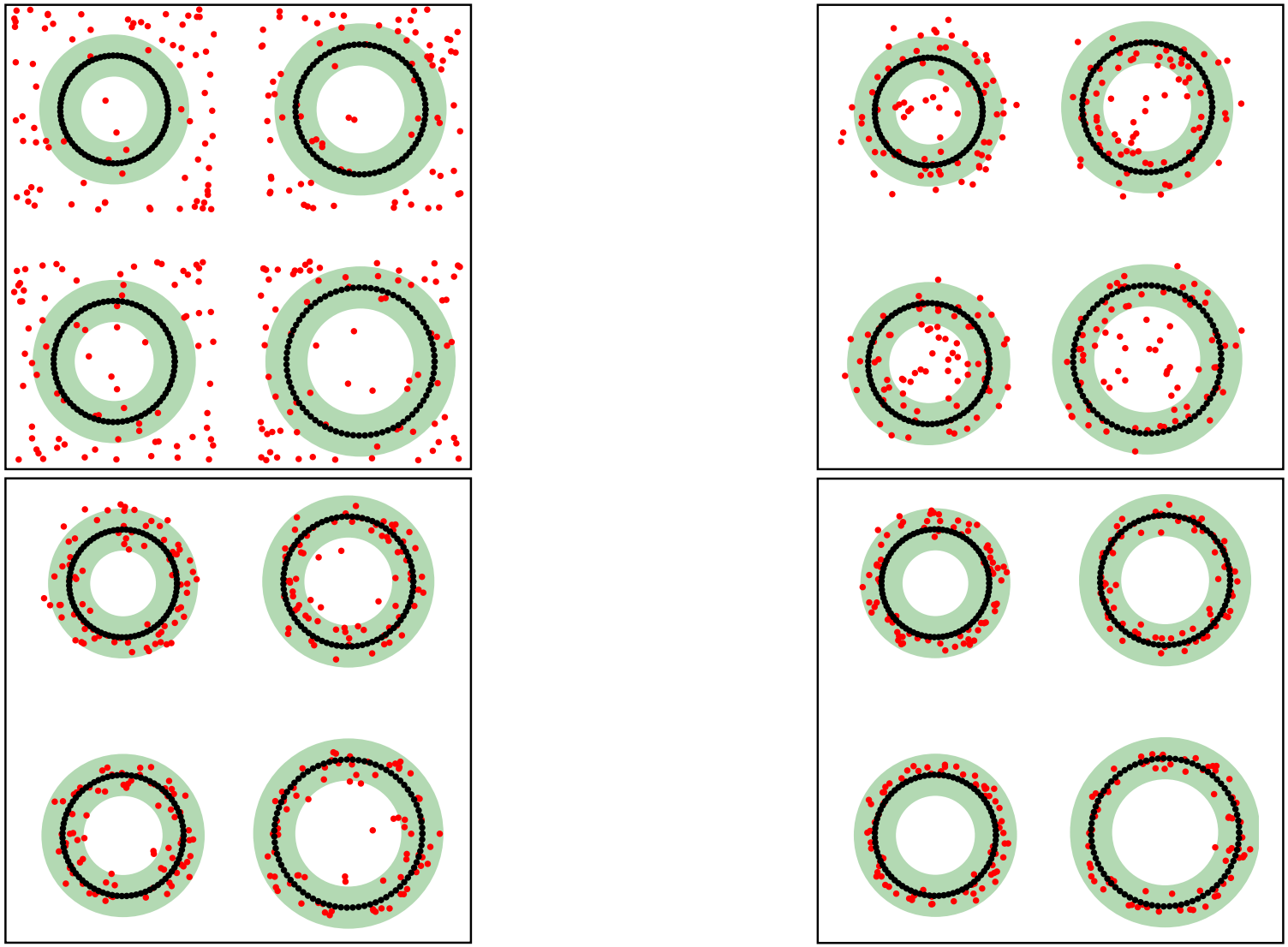
Constrained cGANs, $\varepsilon = 0.1$ (left) and $\varepsilon = 0.2$ (right)





Results for $\varepsilon \neq 0$ (2)

The generated data at different epochs (1, 21, 41, and 61), $\varepsilon = 0.4$



Instead of a conclusion

- More complex example:generate velocity fields of turbulent flows with zero divergence $\equiv$ constraint
- a general approach is proposed of embedding constraints in GANs for emulating physical systems.
- The physical constraints are embedded to the loss function of the generator to help the weaker party in the two-player game, which help the training to achieve equilibrium faster.
- This is motivated by the observation that the constraints reduce the effective dimension of the search space for the weight optimization and thus accelerate the training convergence.
- Moreover, there is the extension of the proposed framework of constrained GANs to incorporate imprecise constraints. This is motivated by the fact that many physical constraints are not known exactly in practical applications.