Synthesizing Programs for Images using Reinforced Adversarial Learning (SPIRAL)

Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, S. M. Ali Eslami, Oriol Vinyals ;
Proceedings of the 35th International Conference on Machine Learning, 2018

Presented by: Hadi Nekoei

UNIVERSITY OF WATERLOO

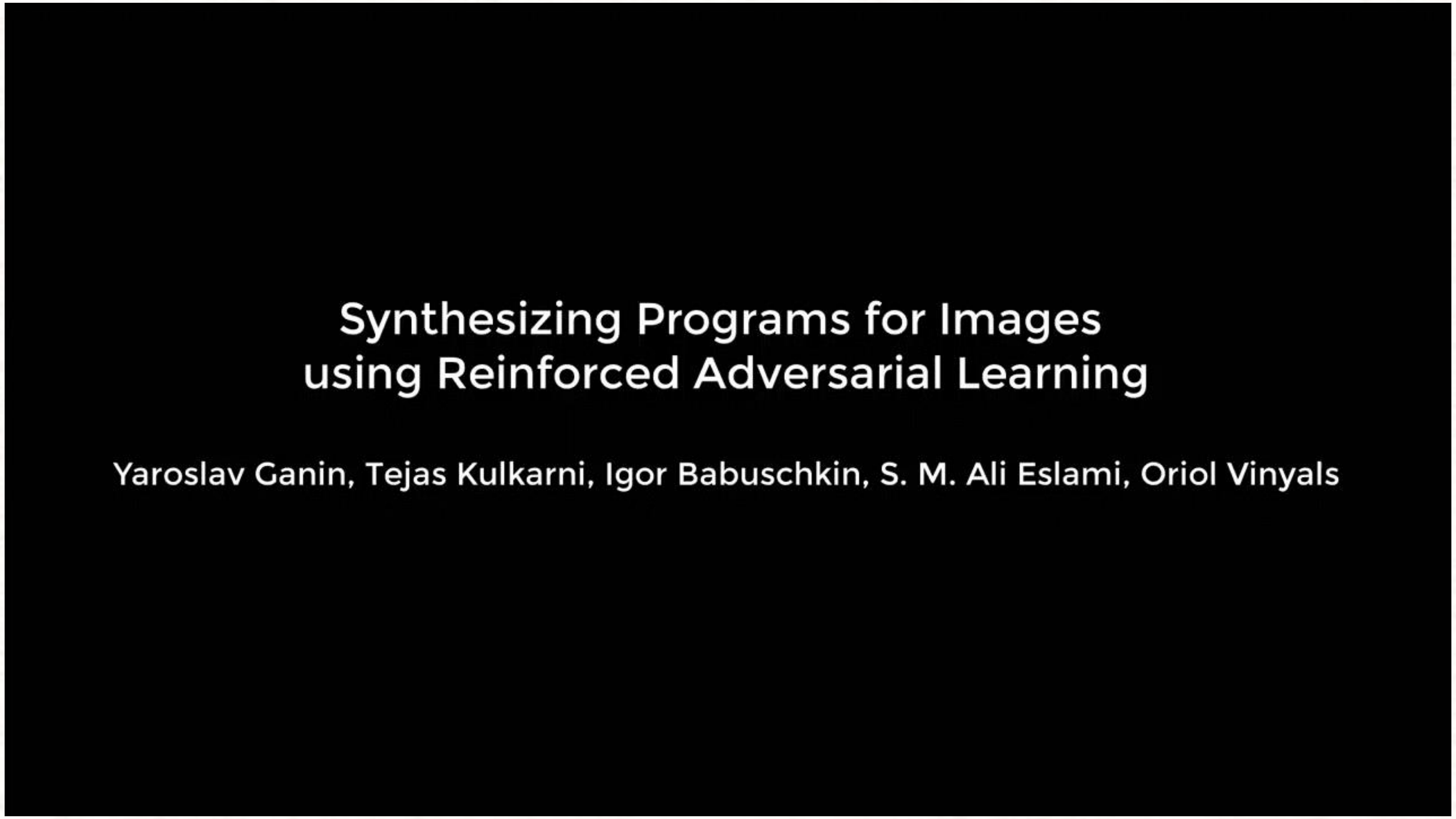
Motivation

The ability to interpret objects through the tools that created them gives us a richer understanding of the world and is an important aspect of our intelligence.

It is commonly believed that humans exploit simulations to learn this skill (Lake et al., 2017).

https://deepmind.com/blog/learning-to-generate-images/

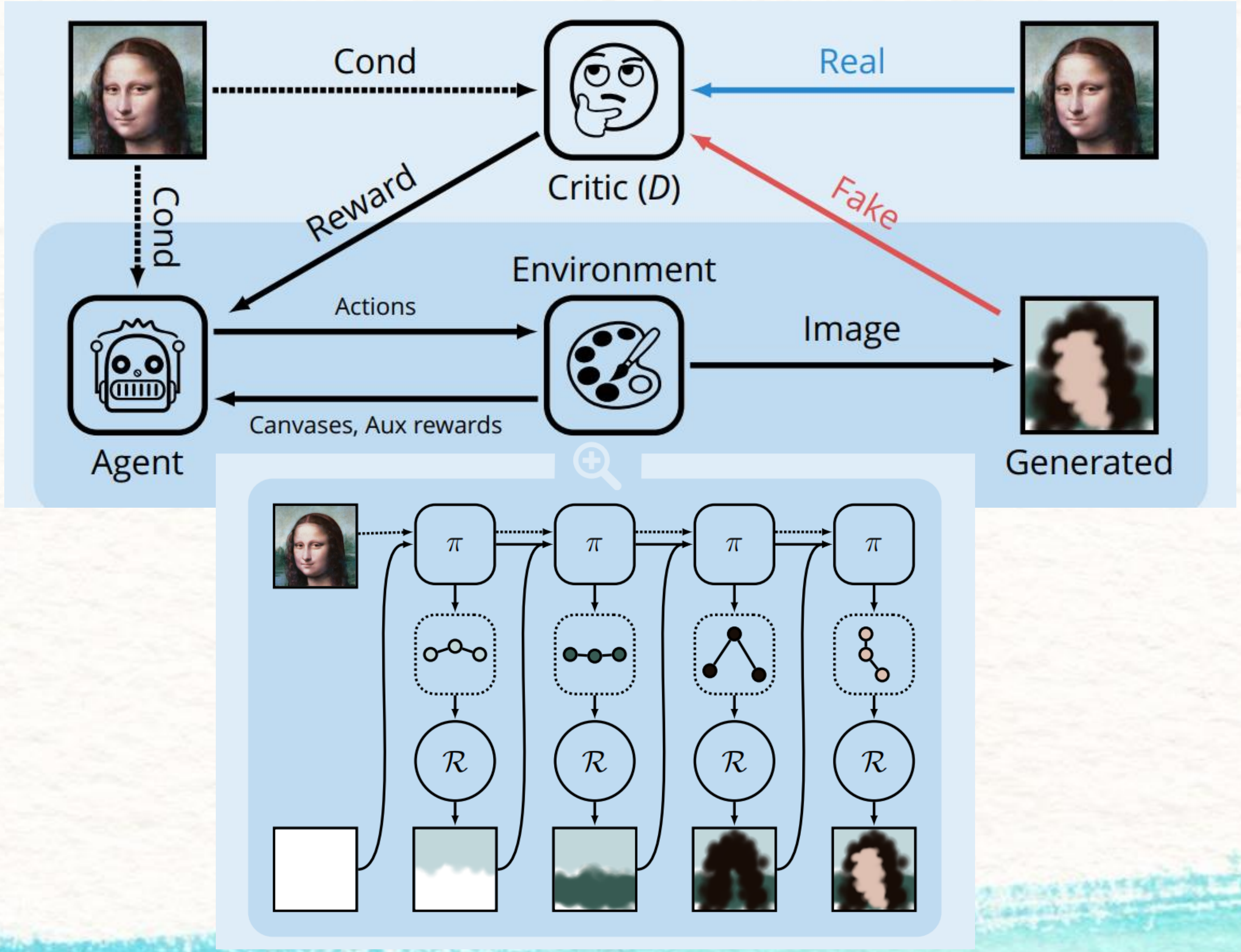
Demo

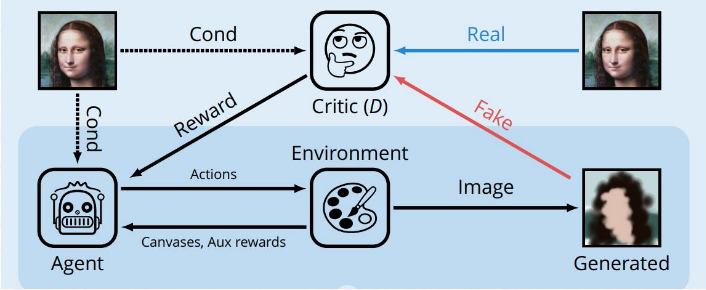


https://www.youtube.com/watch?v=iSyvwAwa7vk&feature=youtu.be

SPIRAL Architecture

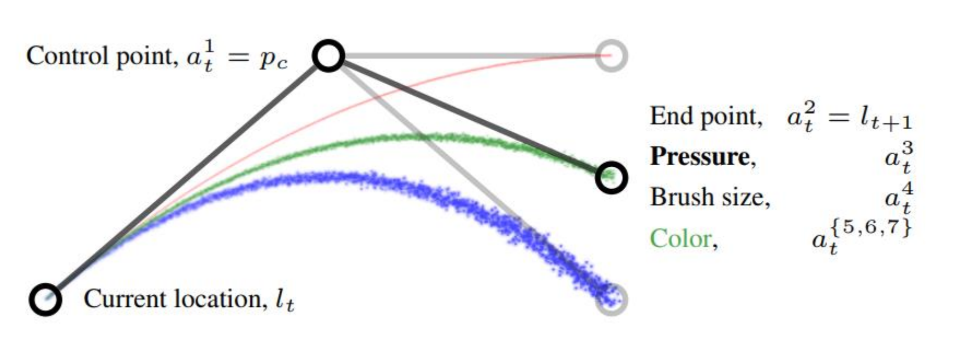
SPIRAL Architecture



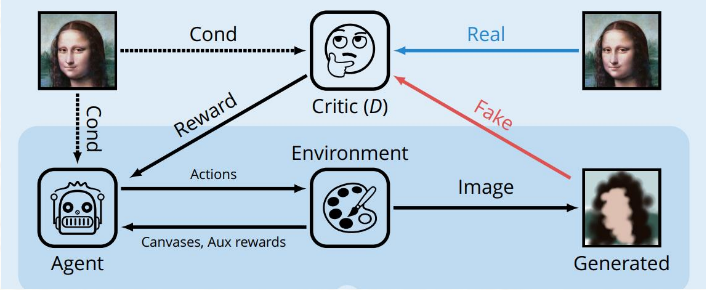


Environments

- An open-source painting library *libmypaint*
  - For MNIST, OMNIGLOT and CELEBA generation

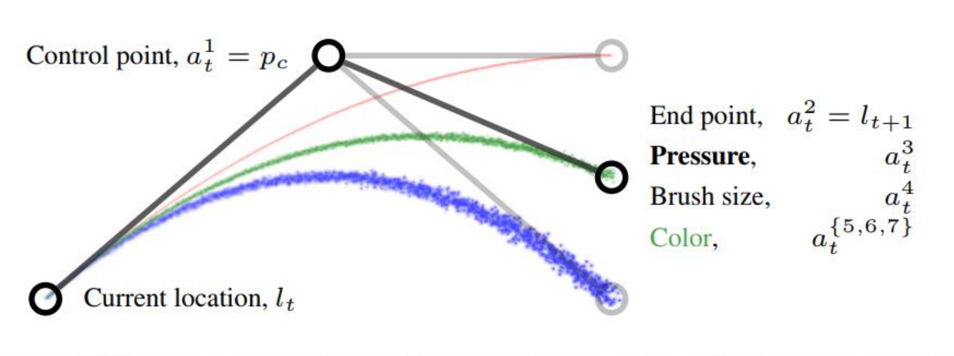


$$a_t = (a_t^1, a_t^2, a_t^3, \ldots, a_t^8)$$



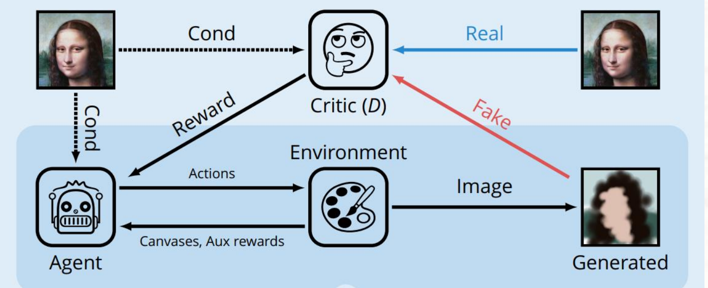
Environments

- An open-source painting library *libmypaint*
  - For MNIST, OMNIGLOT and CELEBA generation

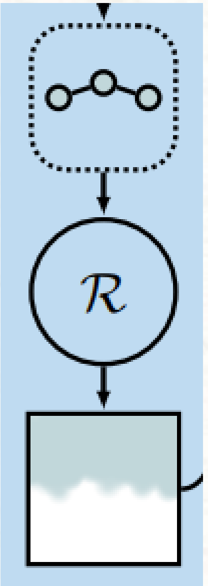


$$a_t = (a_t^1, a_t^2, a_t^3, \dots, a_t^8)$$

- MuJoCo-based environment (Todorov et al., 2012)
  - MUJOCO SCENES experiment

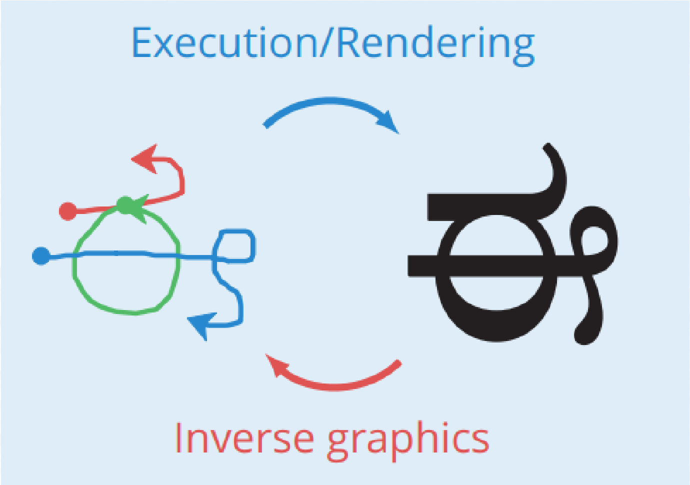


Programs

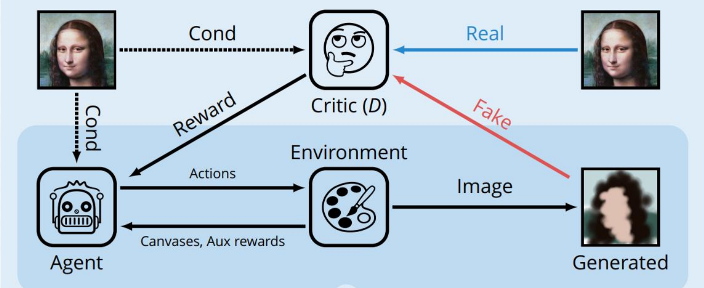


Modelling data distribution in the space of visual programs (**not in the pixel-space**)

Goal: Finding p_{prog} such that *execute* $(p_{prog}) \approx p_{data}$



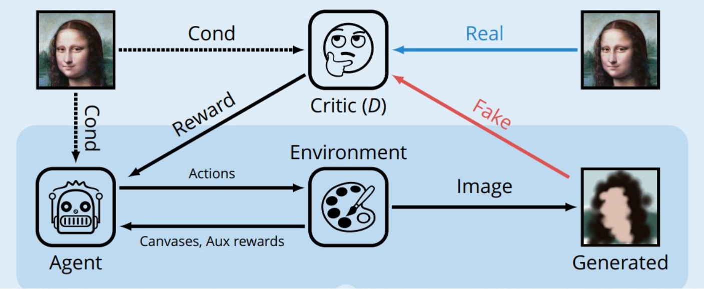
Discriminator Objective:



$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x}\sim p_d}\left[D(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}\sim p_g}\left[D(\mathbf{x})\right] + R$$

- R is a regularization term

Discriminator Objective:



$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x}\sim p_d}[D(\mathbf{x})] + \mathbb{E}_{\mathbf{x}\sim p_g}[D(\mathbf{x})] + R,$$

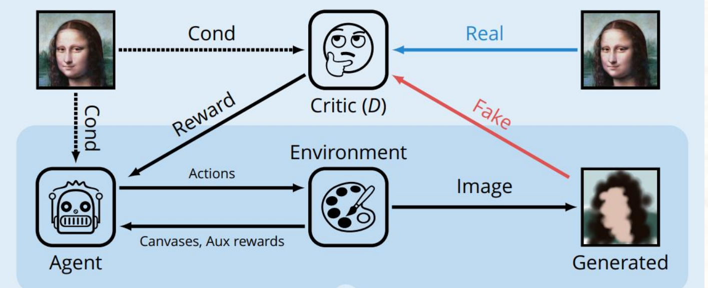
- R is a regularization term
- In the context of this model, optimizing minimax objective form is hard

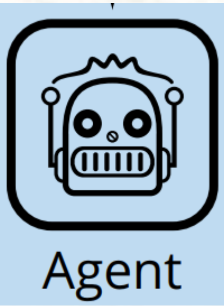
WGAN

Wasserstein distance (Earth Mover's distance) as a measure of divergence between distributions

$$W(p_r, p_g) = \inf_{\gamma\sim\Pi(p_r,p_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]$$

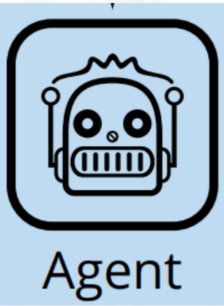
Generator (Policy) Objective:



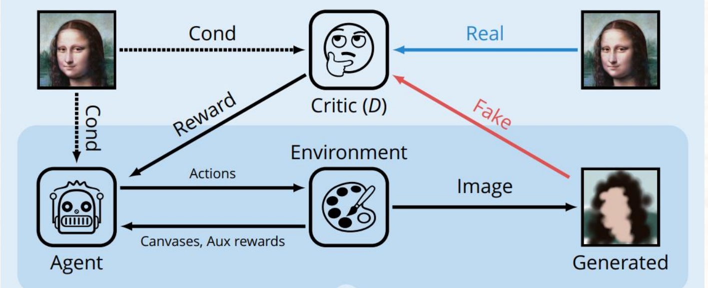


$$\mathcal{L}_G = -\mathbb{E}_{\mathbf{x}\sim p_g}\left[D(\mathbf{x})\right]$$

Since the generator is an arbitrary non-differentiable function



Generator (Policy) Objective:



$$\cancel{\mathcal{L}_G = -\mathbb{E}_{\mathbf{x}\sim p_g}[D(\mathbf{x})]}$$

Since the generator is an arbitrary non-differentiable function

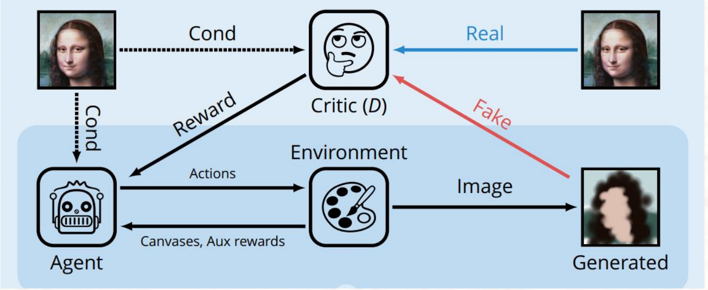
Advantage actor-critic (A2C): maximization of the expected return:

$$\mathcal{L}_G = -\sum_t \log \pi(a_t \,|\, s_t; \theta)\,[R_t - V^\pi(s_t)]$$

where V^π is an approximation to the value function which is considered to be independent of θ, and $R_t = \sum^N r_t$ is a 1-sample Monte-Carlo estimate of the return.

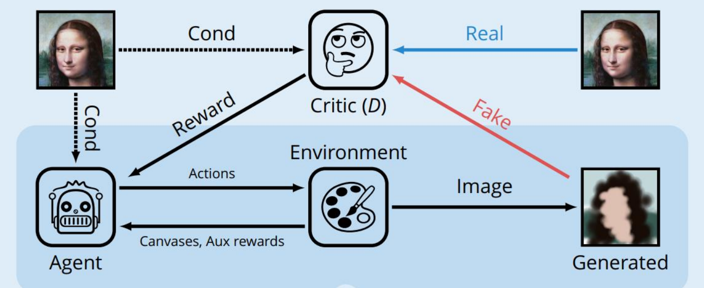
$$r_t = \begin{cases} 0, & t < N, \\ D(\mathcal{R}(a_1, a_2, \ldots, a_N)), & t = N. \end{cases}$$

Conditional Generation:



- So far, we have described the case of unconditional generation!
- One might be interested in finding a specific program that generates a given image x_{target}.

$$\mathcal{L}_D = -\mathbb{E}_{\mathbf{x}\sim p_d}\left[D(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}\sim p_g}\left[D(\mathbf{x})\right] + R,$$



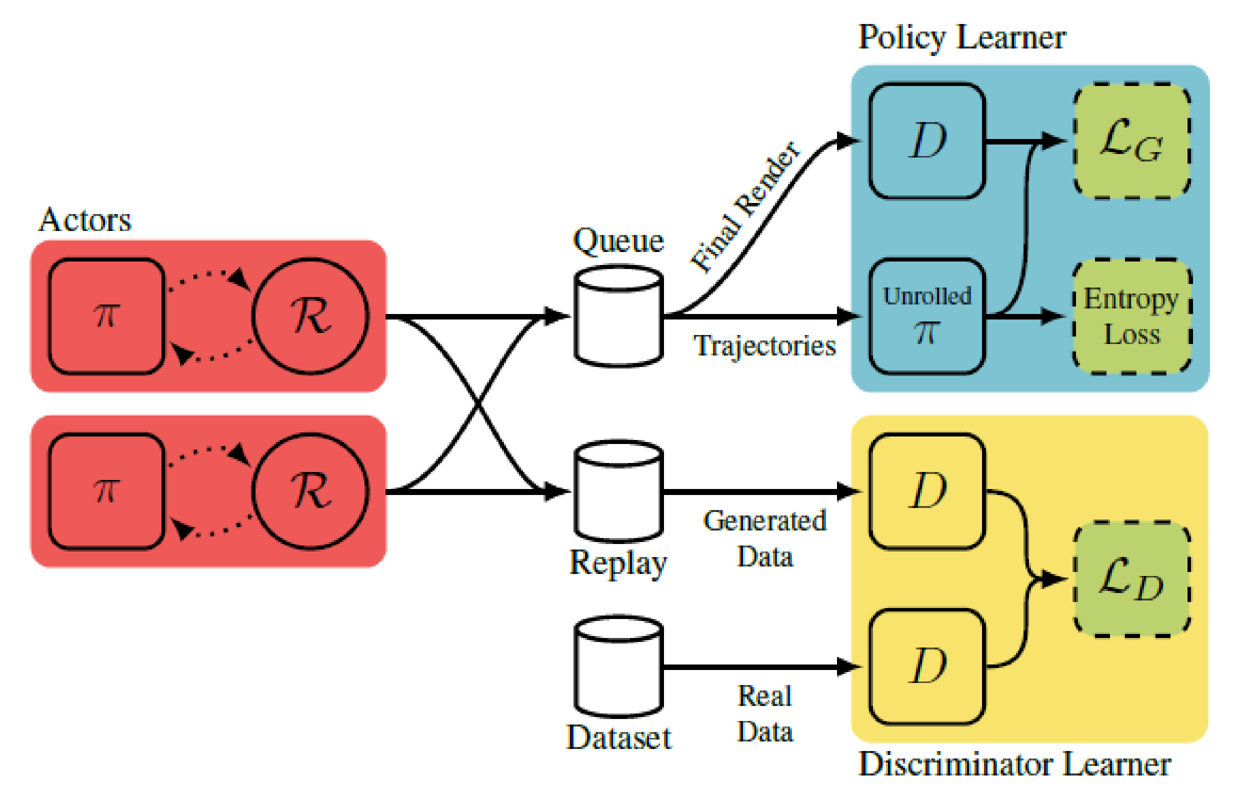
Conditional Generation:

- So far, we have described the case of unconditional generation!
- One might be interested in finding a specific program that generates a given image x_{target}.

$$\mathcal{L}_D = -D(\mathbf{x}_{\text{target}} \mid \mathbf{x}_{\text{target}}) + \mathbb{E}_{\mathbf{x}\sim p_g}\left[D(\mathbf{x}|\mathbf{x}_{\text{target}})\right]$$

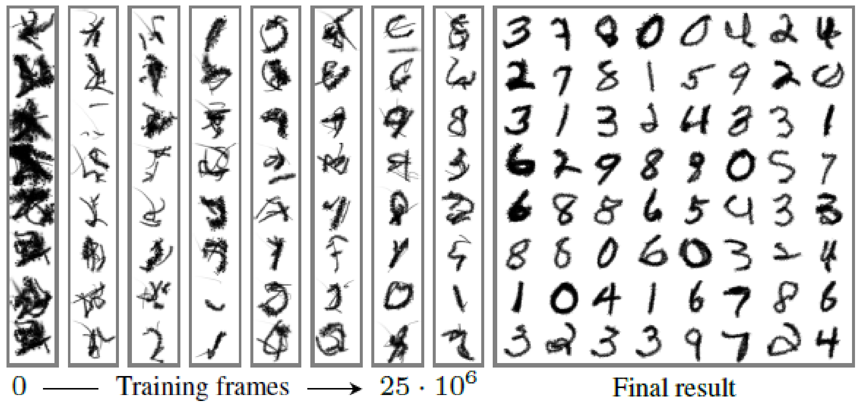
Distributed Learning

- **Actors:** Up to 64
- **A policy learner:**
- **discriminator learner:**:



Experiments:

- MNIST (LeCun et al., 1998):
  - It contains 70,000 examples of handwritten digits, of which 10,000 constitute a test set. Each example is a 28×28 grayscale image.
  - To encourage the agent to draw a digit in a single continuous motion of the brush, we provide a small negative reward for starting each continuous sequence of strokes.

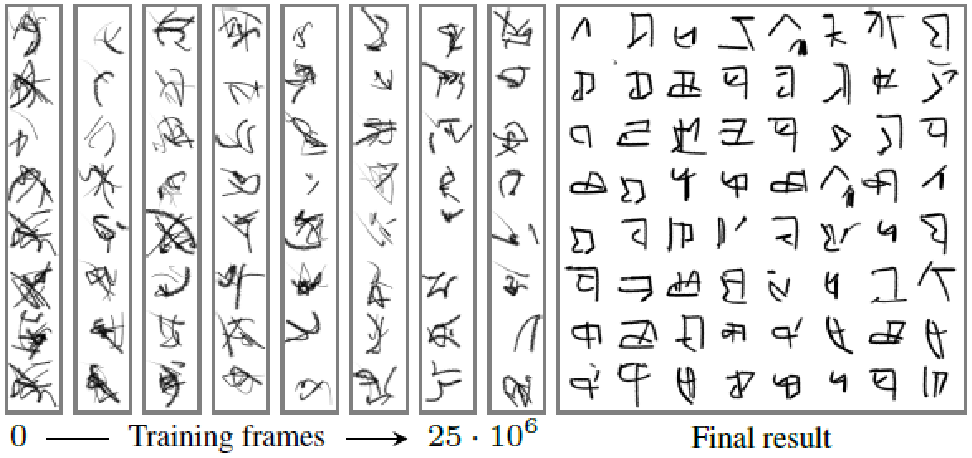


(a) MNIST unconditional generation


(b) MNIST reconstruction

Experiments:

- OMNIGLOT (Lake et al., 2015):
  - Comprises 1623 handwritten characters from 50 alphabets.
  - Compared to MNIST, this dataset introduces three additional challenges:
    - higher data variability

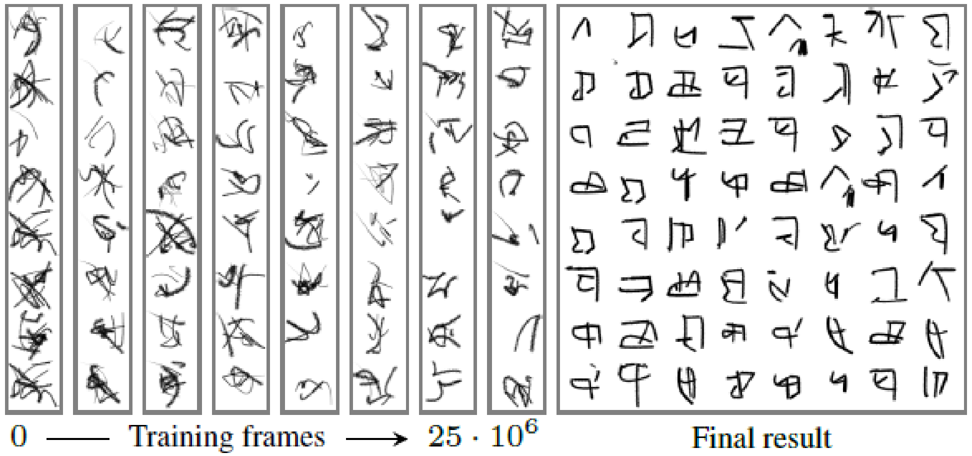


(a) Omniglot unconditional generation


(b) Omniglot reconstruction

Experiments:

- OMNIGLOT (Lake et al., 2015):
  - Comprises 1623 handwritten characters from 50 alphabets.
  - Compared to MNIST, this dataset introduces three additional challenges:
    - higher data variability
    - higher complexity of symbols (e.g., disjoint subcurves) and

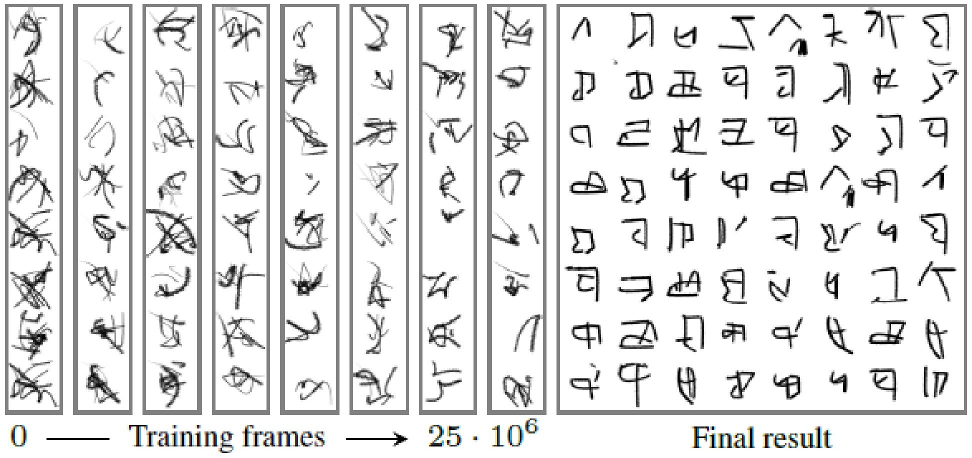


(a) Omniglot unconditional generation

(b) Omniglot reconstruction

Experiments:

- OMNIGLOT (Lake et al., 2015):
  - Comprises 1623 handwritten characters from 50 alphabets.
  - Compared to MNIST, this dataset introduces three additional challenges:
    - higher data variability
    - higher complexity of symbols (e.g., disjoint subcurves) and
    - fewer (only 20) data points per symbol class.



(a) Omniglot unconditional generation

(b) Omniglot reconstruction

Experiments:

- CELEBA (Liu et al., 2015):
  - contains over 200,000 color headshots of celebrities with large variation in poses, backgrounds and lighting conditions.
  - The SPIRAL agent reconstructs human faces in 20 strokes.

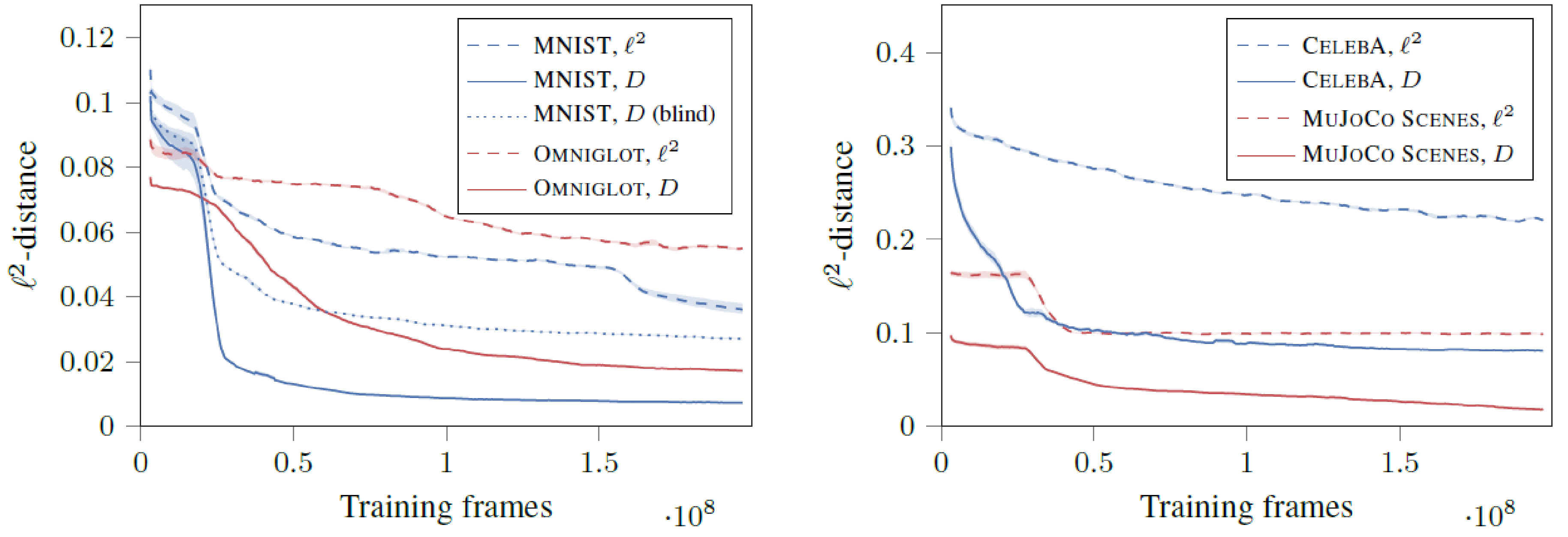
Experiments:

- MUJOCO SCENES dataset:
  - Consisting of renders of simple 3D primitives (up to 5 objects) scattered around a square platform. The training set is comprised of 50,000 RGB images generated by means of the MuJoCo environment
  - Only considering the case of conditional generation

ℓ^2-distance between reconstructions and ground truth

Based on two kinds of rewards:

- Discriminator score (D)
- ℓ^2- distance



(a) MNIST and OMNIGLOT (b) CELEBA and MUJOCO SCENES

Discussion and Critiques:

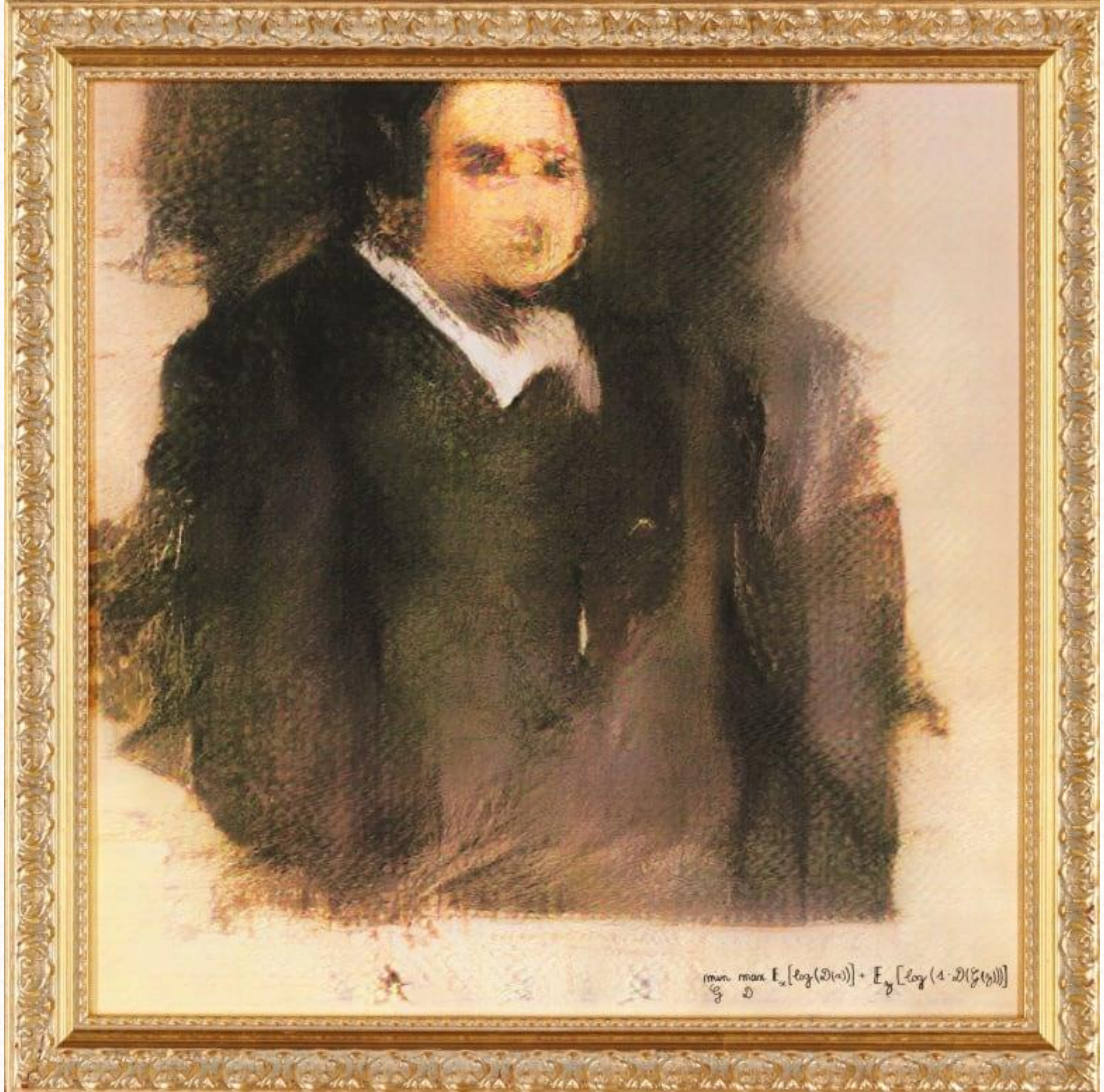
- Sophisticated search algorithms for policy improvement. For instance, Monte Carlo Tree Search can be used, analogous to AlphaGo Zero (Silver et al., 2017).
- Future work should explore different parameterizations of action spaces. For instance, the use of two arbitrary control points is perhaps not the best way to represent strokes.
- Using a joint image-action discriminator similar to BiGAN/ALI(Donahue et al., 2016; Dumoulin et al., 2016) could result in a more meaningful learning signal, since D will be forced to focus on the semantics of the image.

“Artwork created with artificial intelligence fetches more than $400K US at major auction” !!!

Thanks for your attention!

Any Questions?

https://www.cbc.ca/news/entertainment/ai-artwork-sells-for-400k-auction-1.4877945



Appendix

Even when two distributions are located in lower dimensional manifolds without overlaps, **Wasserstein** distance can still provide a meaningful and smooth representation of the distance in-between.

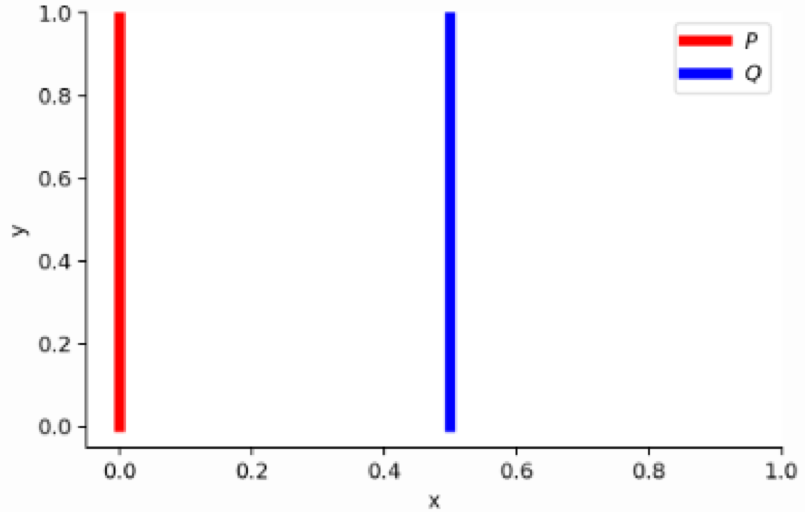


Fig. 8. There is no overlap between P and Q when $\theta \neq 0$.

When $\theta \neq 0$:

$$D_{KL}(P\|Q) = \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q\|P) = \sum_{x=\theta, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{JS}(P,Q) = \frac{1}{2}\Big(\sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2}\Big) = \log 2$$

$$W(P,Q) = |\theta|$$

But when $\theta = 0$, two distributions are fully overlapped:

$$D_{KL}(P\|Q) = D_{KL}(Q\|P) = D_{JS}(P,Q) = 0$$

$$W(P,Q) = 0 = |\theta|$$

https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#wasserstein-gan-wgan