Learning Dynamic Generative Models via Causal Optimal Transport

Beatrice Acciaio
London School of Economics

joint work with Michael Munn (Google NY), Tianlin Xu (LSE), and Kevin Li (UCL)
Idea in a nutshell

► We observe a **sample of paths or a long time series** - evolution of some process of interest (asset price process, volatility surface, LOB, claim process, audio/video data,...)

► We want to understand the **distribution** underlying the observed sample
Idea in a nutshell

- We observe a sample of paths or a long time series - evolution of some process of interest (asset price process, volatility surface, LOB, claim process, audio/video data, ...)
- We want to understand the distribution underlying the observed sample
- We want to train a generator to:
  - **generate** real-looking samples (e.g. to extend available data set for training and evaluation of trading strategies, scenario generation for risk assessment)
  - **predict** the evolution of the path given that we observe part of it (forecasting)
- For this we propose a dynamic modification of GANs
Outline

- Introduction to Generative Adversarial Networks (GANs)
- Our toolkit: Causal Optimal Transport (COT)
- Dynamic GANs via COT
- Applications
Outline

- Introduction to Generative Adversarial Networks (GANs)
- Our toolkit: Causal Optimal Transport (COT)
- Dynamic GANs via COT
- Applications
Generative Adversarial Networks (Goodfellow et al. 2014)

**Generative:** train a **Generator G** to learn data distribution from an i.i.d. sample of observations (training data)

**Adversarial:** set a **Discriminator D** against G, to stimulate G to do a better job
Generative Adversarial Networks (Goodfellow et al. 2014)

**Generative:** train a **Generator G** to learn data distribution from an i.i.d. sample of observations (training data)

**Adversarial:** set a **Discriminator D** against G, to stimulate G to do a better job

- In a loop, we train: **G** to generate real-looking samples, and **D** to recognize whether an element is real or fake (generated by G)
- G and D compete with each other, both improving, until the generated samples are indistinguishable from real data
Generative: train a **Generator G** to learn data distribution from an i.i.d. sample of observations (training data)

Adversarial: set a **Discriminator D** against G, to stimulate G to do a better job

- In a loop, we train: G to generate real-looking samples, and D to recognize whether an element is real or fake (generated by G)
- G and D compete with each other, both improving, until the generated samples are indistinguishable from real data

\[ \square \text{real} \rightarrow \square / \Diamond \]

\[ \text{latent} \rightarrow \boxed{G} \rightarrow \Diamond \text{fake} \]
Generative Adversarial Networks (Goodfellow et al. 2014)

- training data $\{x^i\}_{i=1}^N \subset \mathcal{X}$, empirical distr. $p_{data} = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^i}$
- latent space $\mathcal{Z}$, $\text{dim}(\mathcal{Z}) \ll \text{dim}(\mathcal{X})$, noise distr. $p_{noise} \in P(\mathcal{Z})$
- $g_\theta : \mathcal{Z} \to \mathcal{X}$ generates samples, $p_{\theta}^\text{gen} = g_\theta \# p_{noise} \in P(\mathcal{X})$
- $f_\varphi : \mathcal{X} \to [0, 1]$ high value if D believes input likely to be real
Generative Adversarial Networks (Goodfellow et al. 2014)

- training data \( \{x^i\}_{i=1}^{N} \subset \mathcal{X} \), empirical distr. \( p_{\text{data}} = \frac{1}{N} \sum_{i=1}^{N} \delta_{x^i} \)
- latent space \( \mathcal{Z} \), \( \dim(\mathcal{Z}) \ll \dim(\mathcal{X}) \), noise distr. \( p_{\text{noise}} \in \mathcal{P}(\mathcal{Z}) \)
- \( g_\theta : \mathcal{Z} \to \mathcal{X} \) generates samples, \( p^\theta_{\text{gen}} = g_\theta # p_{\text{noise}} \in \mathcal{P}(\mathcal{X}) \)
- \( f_\varphi : \mathcal{X} \to [0, 1] \) high value if \( D \) believes input likely to be real

**Problem formulation:**

\[
\inf_{\theta} \sup_{\varphi} \left\{ \mathbb{E}_{x \sim p_{\text{data}}} [\ln f_\varphi(x)] + \mathbb{E}_{z \sim p_{\text{noise}}} [\ln(1 - f_\varphi(g_\theta(z)))] \right\}
\]

**D:** learns \( f_\varphi \) (via NN) s.t. \( f_\varphi(\text{real}) \sim 1, f_\varphi(\text{fake}) \sim 0 \)

**G:** learns decoding map \( g_\theta \) (via NN) to maximally confuse \( D \)
Why not Maximum Likelihood Estimation?

- Density fitting: $d p^\theta_{\text{gen}}(x) = m_\theta(x) \, dx$
- MLE: $\sup_\theta \frac{1}{N} \sum_{i=1}^N \ln m_\theta(x^i) \leftrightarrow \inf_\theta H(p_{\text{data}} \mid p^\theta_{\text{gen}})$ rel. entr.
- But $p^\theta_{\text{gen}}$ has no density in $\mathcal{X}$, supports of $p^\theta_{\text{gen}}$ and $p_{\text{data}}$ may even be non-overlapping (MLE not well defined)
Why not Maximum Likelihood Estimation?

- Density fitting: \( dp^\theta_{gen}(x) = m_\theta(x)dx \)
- MLE: \( \sup_\theta \frac{1}{N} \sum_{i=1}^{N} \ln m_\theta(x^i) \leftrightarrow \inf_\theta H(p_{data}|p^\theta_{gen}) \) rel. entr.
- But \( p^\theta_{gen} \) has no density in \( \mathcal{X} \), supports of \( p^\theta_{gen} \) and \( p_{data} \) may even be non-overlapping (MLE not well defined)

\( \Rightarrow \) Need of a more flexible divergence to compare \( p^\theta_{gen} \) and \( p_{data} \)
\( \rightarrow \) GANs (and their modifications)
Problems (with original GANs):

- **Continuity** w.r.t. parameters
- **Convergence**
- **Stability**
Generative Adversarial Networks: moving on

Problems (with original GANs):
- Continuity w.r.t. parameters
- Convergence
- Stability

Some ways out:
- Gradient-based regularizations (added to the loss function)
- D calculates some other divergence between $p_{data}$ and $p_{gen}^\theta$:
  Integral Probability Metrics, Maximum Mean Discrepancy, Wasserstein distance, energy distance
**Problems** (with original GANs):

- **Continuity** w.r.t. parameters
- Convergence
- Stability

**Some ways out:**

- Gradient-based **regularizations** (added to the loss function)
- $D$ calculates some other **divergence** between $p_{data}$ and $p_{\theta}^{gen}$: Integral Probability Metrics, Maximum Mean Discrepancy, Wasserstein distance, energy distance

**Wasserstein distance:**

$$\mathcal{W}_1(\mu, \nu) = \inf \left\{ \mathbb{E}^\pi [\|x - y\|] : \pi_1 = \mu, \pi_2 = \nu \right\}$$

$$\implies \inf_{\theta} \underbrace{\mathcal{W}_1(p_{data}, p_{\theta}^{gen})}_{\text{D}}$$

Beatrice Acciaio (LSE)  Causal Generative Adversarial Networks
**Dual formulation** of the Wasserstein distance:

\[
\mathcal{W}_1(\mu, \nu) = \sup_{f \text{ Lip}_1} \{ \mathbb{E}^\mu[f] - \mathbb{E}^\nu[f] \}
\]
Dual formulation of the Wasserstein distance:

\[ \mathcal{W}_1(\mu, \nu) = \sup_{f \text{ Lip}_1} \{ \mathbb{E}^\mu[f] - \mathbb{E}^\nu[f] \} \]

→ enforce Lip constraint via gradient penalization

\[ \inf_\theta \sup_\varphi \left\{ \mathbb{E}^{p_{data}}[f_\varphi(x)] - \mathbb{E}^{p_\theta_{gen}}[f_\varphi(y)] + \text{Lip. penalization} \right\} \]
Wasserstein GANs (Arjovsky et al., Gulrajani et al. 2017)

Dual formulation of the Wasserstein distance:

$$\mathcal{W}_1(\mu, \nu) = \sup_{f \text{ Lip}_1} \{ \mathbb{E}^\mu[f] - \mathbb{E}^\nu[f] \}$$

→ enforce Lip constraint via gradient penalization

$$\inf_{\theta} \sup_{\varphi} \left\{ \mathbb{E}^{p_{\text{data}}}[f_\varphi(x)] - \mathbb{E}^{p_\theta^{\text{gen}}}[f_\varphi(y)] + \text{Lip. penalization} \right\}$$

- **Continuity**: if $\theta \mapsto g_\theta$ cont. $\Rightarrow \theta \mapsto \mathcal{W}_1(p_{\text{data}}, p_\theta^{\text{gen}})$ cont.
- **Convergence**: WGANs converge if D trained till optimality
- **WGANs outperform** MLE and MLE-NN unless exact parametric form of data is known
Instead, consider **primal formulation** with different cost functions:

\[ \mathcal{W}_f(\mu, \nu) = \inf \{ \mathbb{E}^\pi[\| f(x) - f(y) \|] : \pi_1 = \mu, \pi_2 = \nu \} \]

and let \( D \) learn the cost function:

\[ \inf_{\theta} \sup_{\varphi} \mathcal{W}_{f,\varphi}(p_{data}, p_{\theta}^{gen}) \]

\[ \Downarrow \]

\[ G \quad \rightarrow \quad \mathcal{W}_{f,\varphi} \quad \text{regularized and solved via Sinkhorn algorithm} \]

\[ \rightarrow \text{numerically more stable} \quad \text{(in the dual formulation, gradient requires differentiating the Kantorovich potential)} \]
GANs (Goodfellow et al. 2014):
continuity ×, convergence ×, stability ×

WGANs:
continuity ✓, convergence ✓, stability ×
▶ primal (Genevay, Peyré, Cuturi 2017)
continuity ✓, convergence ✓, stability ✓
Finally...what we are doing

- We consider a **dynamic framework**: we observe a sample of paths or a time series, and we want to generate/predict paths.
- We mimic primal approach by Genevay et al.
- We need **good notion of distance for sequential data**
- D will compute robust distance between sequential data, and learn the cost function.
Outline

- Introduction to Generative Adversarial Networks (GANs)
- Our toolkit: Causal Optimal Transport (COT)
- Dynamic GANs via COT
- Applications
Given Polish probability spaces \((\mathcal{X}, \mu), (\mathcal{Y}, \nu)\), move the mass from \(\mu\) to \(\nu\) minimizing the cost of transportation \(c : \mathcal{X} \times \mathcal{Y} \rightarrow [0, \infty] :\)

\[
\text{OT}_c(\mu, \nu) := \inf \{ \mathbb{E}^\pi [c(x, y)] : \pi \in \Pi(\mu, \nu) \},
\]

\(\Pi(\mu, \nu)\): probability measures on \(\mathcal{X} \times \mathcal{Y}\) with marginals \(\mu\) and \(\nu\)

e.g. \(\mathcal{X} = \mathcal{Y} = \mathbb{R}^d\), \(c(x, y) = \|x - y\| \rightarrow \text{Wasserstein distance}\)
Monge transport: all mass sitting on $x$ is transported into $y = F(x)$.
Kantorovich transport: mass can split.
→ **Dynamic framework** (e.g. \( \mathcal{X} = \mathcal{Y} = \mathbb{R}^{d \times T} \), \( d \)-dim paths long \( T \)) something that evolves in time: “move distribution of process \( (X_t)_{t=1,..,T} \) into distribution of process \( (Y_t)_{t=1,..,T} \)”
Dynamic framework (e.g. $\mathcal{X} = \mathcal{Y} = \mathbb{R}^{d \times T}$, $d$-dim paths long $T$) something that evolves in time: “move distribution of process $(X_t)_{t=1,..,T}$ into distribution of process $(Y_t)_{t=1,..,T}$”

What is a good distance in a dynamic framework?
Causal Optimal Transport

→ **Dynamic framework** (e.g. \( \mathcal{X} = \mathcal{Y} = \mathbb{R}^{d \times T} \), \( d \)-dim paths long \( T \)) something that evolves in time: “move distribution of process \((X_t)_{t=1,..,T}\) into distribution of process \((Y_t)_{t=1,..,T}\)”

→ What is a good distance in a dynamic framework?

→ **Idea**: move the mass in a non-anticipative way (\( Y \) is \( X \)-adapted, modulo external randomization)
→ **Dynamic framework** (e.g. $\mathcal{X} = \mathcal{Y} = \mathbb{R}^{d \times T}$, $d$-dim paths long $T$) something that evolves in time: “move distribution of process $(X_t)_{t=1,\ldots,T}$ into distribution of process $(Y_t)_{t=1,\ldots,T}$”

→ What is a **good distance in a dynamic framework**?

→ **Idea**: move the mass in a non-anticipative way ($Y$ is $X$-adapted, modulo external randomization)

→ **Mathematically**: $\pi \in \mathcal{P}(\mathbb{R}^{d \times T} \times \mathbb{R}^{d \times T})$ is **causal** if

$$
\pi(dy_t|dx_1, \cdots, dx_T) = \pi(dy_t|dx_1, \cdots, dx_t) \quad \forall t
$$
Dynamic framework (e.g. $\mathcal{X} = \mathcal{Y} = \mathbb{R}^{d \times T}$, $d$-dim paths long $T$) something that evolves in time: “move distribution of process $(X_t)_{t=1,..,T}$ into distribution of process $(Y_t)_{t=1,..,T}$”

What is a good distance in a dynamic framework?

Idea: move the mass in a non-anticipative way ($Y$ is $X$-adapted, modulo external randomization)

Mathematically: $\pi \in \mathcal{P}(\mathbb{R}^{d \times T} \times \mathbb{R}^{d \times T})$ is causal if

$$\pi(dy_t|dx_1, \cdots, dx_T) = \pi(dy_t|dx_1, \cdots, dx_t) \quad \forall t$$

E.g. causal Monge transports:

$$(x_1, ..., x_T) \mapsto (F_1(x_1), F_2(x_1, x_2), ..., F_T(x_1, ..., x_T))$$
Causal Optimal Transport problem:

\[ \text{COT}_c(\mu, \nu) := \inf \left\{ \mathbb{E}^\pi [c(X, Y)] : \pi \in \Pi^{\text{causal}}(\mu, \nu) \right\}, \]

where \( \Pi^{\text{causal}}(\mu, \nu) = \{ \pi \in \Pi(\mu, \nu) : \pi \text{ causal} \} \)

- Nested distance (Pflug and Pichler 2012) for multistage optimization problems
- Adapted Wasserstein distance (Backhoff et al. 2019) for stability in mathematical finance
One equivalent characterization of causality, useful for us:

$$\pi \text{ causal } \iff \mathbb{E}^{\pi}[s(x, y)] = 0 \quad \forall s \in \mathcal{S},$$

where $\mathcal{S}$ is some well-defined linear space of functions.
One equivalent characterization of causality, useful for us:

\[ \pi \text{ causal } \iff \mathbb{E}^\pi[s(x, y)] = 0 \quad \forall s \in \mathbb{S}, \]

where \( \mathbb{S} \) is some well-defined linear space of functions

This allows to rewrite the causal optimal transport problem as

\[ \text{COT}_c(\mu, \nu) = \sup_{s \in \mathbb{S}} \text{OT}_{c+s}(\mu, \nu) \]
Outline

- Introduction to Generative Adversarial Networks (GANs)
- Our toolkit: Causal Optimal Transport (COT)
- Dynamic GANs via COT
- Applications
We want to train a generator to produce sequential data

We build a modification of GAN where $D$ computes the distance between the real distribution $p_{data}$ and the generated distribution $p_{gen}^\theta$ of paths via causal optimal transport.
We want to train a generator to produce sequential data.

We build a modification of GAN where $D$ computes the distance between the real distribution $p_{data}$ and the generated distribution $p_{gen}^\theta$ of paths via causal optimal transport.

\[
\inf_{\theta} \ COT_c(p_{data}, p_{gen}^\theta) = \inf_{\theta} \ inf_{s \in S} OT_{c+s}(p_{data}, p_{gen}^\theta)
\]

OT problems regularized and solved via Sinkhorn algorithm.
Entropic regularization:

\[ \text{OT}_c^\varepsilon(\mu, \nu) := \inf_{\pi \in \Pi(\mu, \nu)} \left\{ \mathbb{E}^\pi \left[ c(x, y) \right] + \varepsilon H(\pi) \right\} \xrightarrow{\varepsilon \to 0} \text{OT}_c(\mu, \nu) \]

\[ \text{COT}_c^\varepsilon(\mu, \nu) := \inf_{\pi \in \Pi^\text{causal}(\mu, \nu)} \left\{ \mathbb{E}^\pi \left[ c(x, y) \right] + \varepsilon H(\pi) \right\} \xrightarrow{\varepsilon \to 0} \text{COT}_c(\mu, \nu) \]
Entropic regularization:

\[
\text{OT}_c^\varepsilon(\mu, \nu) := \inf_{\pi \in \Pi(\mu, \nu)} \left\{ \mathbb{E}^\pi [c(x, y)] + \varepsilon H(\pi) \right\} \xrightarrow{\varepsilon \to 0} \text{OT}_c(\mu, \nu)
\]

\[
\text{COT}_c^\varepsilon(\mu, \nu) := \inf_{\pi \in \Pi^{\text{causal}}(\mu, \nu)} \left\{ \mathbb{E}^\pi [c(x, y)] + \varepsilon H(\pi) \right\} \xrightarrow{\varepsilon \to 0} \text{COT}_c(\mu, \nu)
\]

By dualizing the causality constraint:

\[
\text{COT}_c^\varepsilon(\mu, \nu) = \sup_{s \in \mathbb{S}} \text{OT}_c^{\varepsilon + s}(\mu, \nu)
\]
Dynamic Generative Adversarial Networks via COT

Entropic regularization:

\[ \text{OT}^\varepsilon_c(\mu, \nu) := \inf_{\pi \in \Pi(\mu, \nu)} \left\{ \mathbb{E}^\pi [c(x, y)] + \varepsilon H(\pi) \right\} \xrightarrow{\varepsilon \to 0} \text{OT}_c(\mu, \nu) \]

\[ \text{COT}^\varepsilon_c(\mu, \nu) := \inf_{\pi \in \Pi_{\text{causal}}(\mu, \nu)} \left\{ \mathbb{E}^\pi [c(x, y)] + \varepsilon H(\pi) \right\} \xrightarrow{\varepsilon \to 0} \text{COT}_c(\mu, \nu) \]

- By dualizing the causality constraint:

  \[ \text{COT}^\varepsilon_c(\mu, \nu) = \sup_{s \in \mathcal{S}} \text{OT}^\varepsilon_{c+s}(\mu, \nu) \]

- We remove the bias \( \rightarrow \) Sinkhorn divergence:

  \[ \hat{\text{OT}}^\varepsilon_{c+s}(\mu, \nu) := \text{OT}^\varepsilon_{c+s}(\mu, \nu) - \frac{1}{2} \text{OT}^\varepsilon_{c+s}(\mu, \mu) - \frac{1}{2} \text{OT}^\varepsilon_{c+s}(\nu, \nu) \]
Causal Wasserstein GAN:

\[
\inf_{\theta} \sup_{\varphi} \hat{\text{OT}}^\varepsilon_{c_{\varphi}}\left(p_{\text{data}}, g_{\theta} \# p_{\text{noise}}\right)
\]

→ D learns \( c_{\varphi} \), i.e. the best cost function - 2 networks

→ G learns \( g_{\theta} \), i.e. the best generating function - 1 network
Causal Wasserstein GAN:

\[
\inf_{\theta} \sup_{\varphi} \widehat{\text{OT}}^\varepsilon_{c_{\varphi}}(p_{\text{data}}, g_\theta \neq p_{\text{noise}})
\]

→ D learns \( c_{\varphi} \), i.e. the best cost function - 2 networks

→ G learns \( g_\theta \), i.e. the best generating function - 1 network

- the cost functions \( c_{\varphi} \) are of the form appearing in the dualization of causality

- parameters \( \varphi \) ad \( \theta \) learned through "dynamic architectures": Recurrent Neural Networks, Convolutional Neural Networks,...
Training architecture: example

Recurrent Neural Network

Basic RNN: \( h_t = \sigma_1(Az_t + Bh_{t-1} + a), \quad y_t = \sigma_2(Cs_t) \)

LSTM: more involved structure in the repeating module for \( h_t \)
The algorithm

To solve the min-max problem:

- sample **mini-batches** from real data and from latent space $\to$ empir. distr. $\hat{x}$, $\hat{y}_\theta$
- calculate $c(\hat{x}, \hat{y}_\theta)$ using random projection to reduce dim.
- penalize cost functions $c_\varphi = c + s$ for which $s \notin S$
- compute $\inf_{\Pi(\hat{x}, \hat{y}_\theta)} \left\{ \mathbb{E}_{\pi} \left[ c_\varphi \right] + \epsilon H(\pi) \right\}$ by **Sinkhorn algo** (Cuturi 2013, fast & stable), with pre-determined $\#$ iterations

$\Rightarrow \quad \mathcal{W}_{c_\varphi}^\epsilon (\hat{x}, \hat{y}_\theta)$ (adjusted loss function)
The algorithm

To solve the min-max problem:

- sample mini-batches from real data and from latent space → emp. distr. $\hat{x}, \hat{y}_\theta$
- calculate $c(\hat{x}, \hat{y}_\theta)$ using random projection to reduce dim.
- penalize cost functions $c_\varphi = c + s$ for which $s \notin S$
- compute $\inf_{\Pi(\hat{x}, \hat{y}_\theta)} \left\{ \mathbb{E}^\pi [c_\varphi] + \epsilon H(\pi) \right\}$ by Sinkhorn algo (Cuturi 2013, fast & stable), with pre-determined # iterations

$\Rightarrow \hat{\mathcal{W}}^\varepsilon_{c_\varphi}(\hat{x}, \hat{y}_\theta)$ (adjusted loss function)

Use stochastic Gradient Ascent/Descent to update parameters:

$\varphi_{n+1} = \varphi_n + \alpha \nabla_\varphi \hat{\mathcal{W}}^\varepsilon_{c_\varphi}(\hat{x}, \hat{y}_\theta)$

$\theta_{n+1} = \theta_n - \alpha \nabla_\theta \hat{\mathcal{W}}^\varepsilon_{c_\varphi}(\hat{x}, \hat{y}_\theta)$
Pseudo-code

**Data:** \( \theta_0, \varphi_0, \{x_i\}_{i=1}^N, p_{\text{noise}}, \epsilon, \# \text{ Sinkhorn iter.}, \# \text{ terms in cost}, \) batch size \( m \), learning rate \( \alpha \), critic iter. \( n_c \)

**Result:** \( \theta, \varphi \)

\( \theta \leftarrow \theta_0, \varphi \leftarrow \varphi_0 \)

for \( k = 1, 2, \ldots \) do

  for \( l = 1, 2, \ldots, n_c \) do

    Sample: \( \{x_i\}_{i=1}^m \) from real data, and \( \{z_i\}_{i=1}^m \) from \( p_{\text{noise}} \)
    \(
y^i \leftarrow g_{\theta}(z^i)
    \)
    \(
    \varphi \leftarrow \varphi + \alpha \nabla \varphi \left( \hat{W}^{\epsilon}_{c\varphi}(\hat{x}, \hat{y}_\theta) \right)
    \)

  end

  Sample: \( \{x_i\}_{i=1}^m \) from real data, and \( \{z_i\}_{i=1}^m \) from \( p_{\text{noise}} \)
  \(
y^i \leftarrow g_{\theta}(z^i)
  \)
  \(
  \theta \leftarrow \theta - \alpha \nabla \theta \left( \hat{W}^{\epsilon}_{c\varphi}(\hat{x}, \hat{y}_\theta) \right)
  \)

end
→ Causal Wasserstein GANs: learn how to generate real-looking evolutions given an observed dataset.

→ In progress: conditional modification of the algorithm, for time-series trend prediction, so that we feed the beginning of a sequence and the generator produces some reasonable continuation.

- Mathematically: non-expensive modification
- But requires different choice of architectures
Outline

• Introduction to Generative Adversarial Networks (GANs)
• Our toolkit: Causal Optimal Transport (COT)
• Dynamic GANs via COT
• Applications
Initial testing:

- We have tested for easy-to check features on synthetic data, e.g. reproducing ARMA models and periodic curves.
- We have tested on standard datasets, such as MNIST.
- We are testing video data (sequence of pictures).
Initial testing:

- We have tested for easy-to check features on synthetic data, e.g. reproducing ARMA models and periodic curves
- We have tested on standard datasets, such as MNIST
- We are testing video data (sequence of pictures)

Applications in finance and insurance: data-driven robust (model-independent) analysis

- Market generation for robust pricing and hedging
- Prediction of volatility, LOB,...
- Scenario generation for risk evaluation of insurance companies
MNIST (yes, I know, not truly sequential..)

1 iteration (0.14 sec)  300 iterations (33 sec)  15k iterations (36min)  40k iterations (1h45’)

batch 32, critic 1, $\epsilon = 0.8$, Sinkhorn iter. 30, learning rate 0.0001
Now something sequential but very basic, so we can still use our eyes to judge...

\[ \sim 15k \text{ iterations}, 6 \text{ iterations per sec for GPC and its ‘sequential’ modification, 4 iterations per sec for our CWGAN} \]

- Default, critic 1, \( \epsilon = 0.8 \), Sinkhorn iter. 30, learning rate 0.0001
Acciaio, Backhoff, Jia: Cournot-Nash equilibrium and optimal transport in a dynamic setting, 2020

Arjovsky, Chintala, Bottou: Wasserstein GAN, 2017

Cuturi: Sinkhorn distances: Lightspeed computation of OT, 2013

Genevay, Peyré, Cuturi: Learning Generative Models with Sinkhorn Divergences, 2017

Goodfellow et al.: Generative Adversarial Networks, 2014

Gulrajani et al.: Improved Training of Wasserstein GANs, 2017
Thank you for your attention!