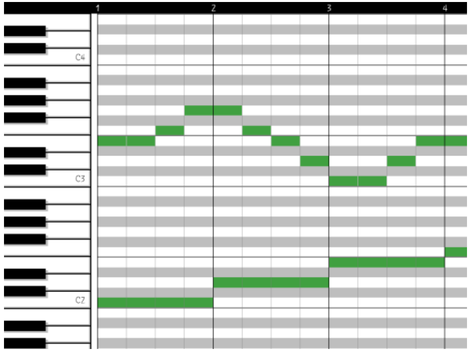
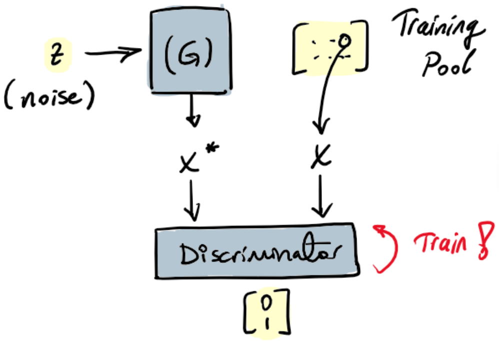
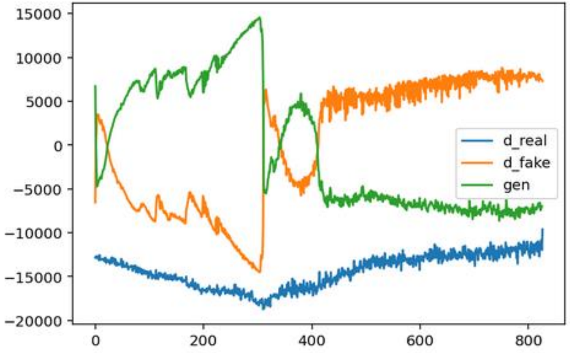
Data Driven Engineering I: Machine Learning for Dynamical Systems

Introduction to Generative Learning: VAEs and GANs

Institute of Thermal Turbomachinery
Prof. Dr.-Ing. Hans-Jörg Bauer







Generative Learning

* Generative Learning & Representation

* Latent Space ⇒ Autoencoders → M′ is easier to learn

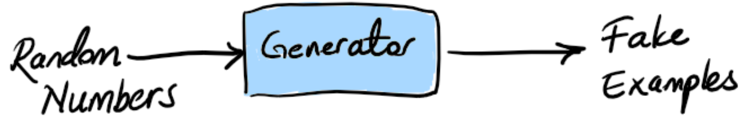
* AE + Gaussian Sampling ⇒ VAEs
- → Functional API
- → λ Layer
- → Embeddings

* Music → MIDI → VAE, LSTM

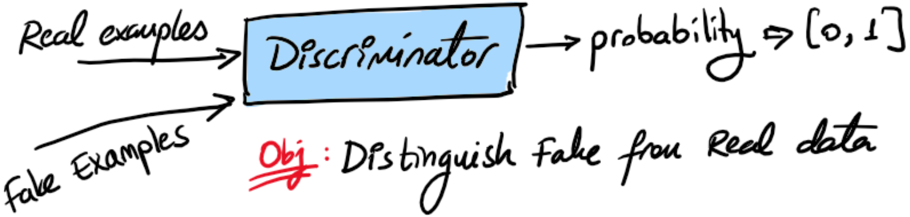
This Week: GANs for Music

Generative Adversarial Networks: GANs

- Generative ⇒ creating non-existing data
- Adversarial ⇒ Competitive dynamics (game-like)
- Network ⇒ Neural networks
- GANS (2014) → Generator, Discriminator


- Random Numbers → Generator → Fake Examples

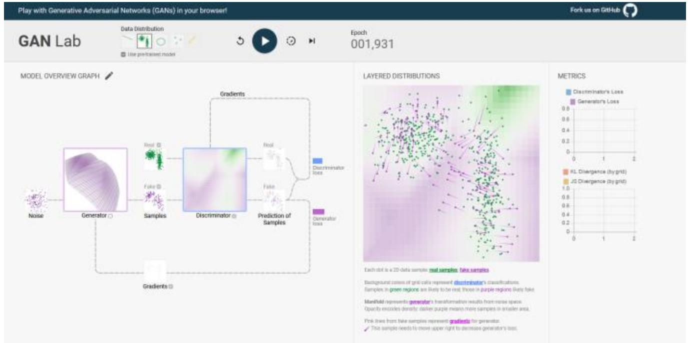
Obj: Be as realistic as possible


- Real examples → Discriminator → probability ⇒ [0, 1]

Fake Examples → Discriminator

Obj: Distinguish Fake from Real data

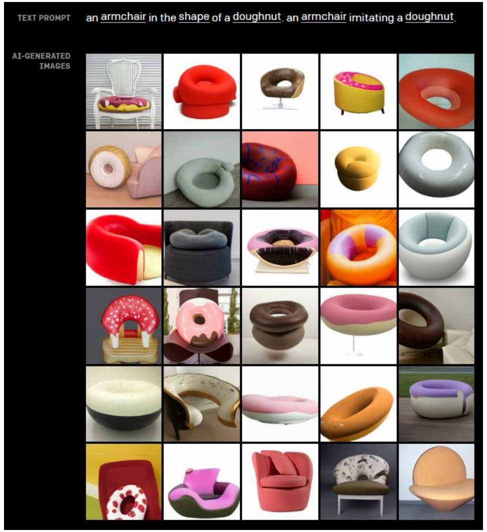
Generative Learning



Play with Generative Adversarial Networks (GANs) in your browser!

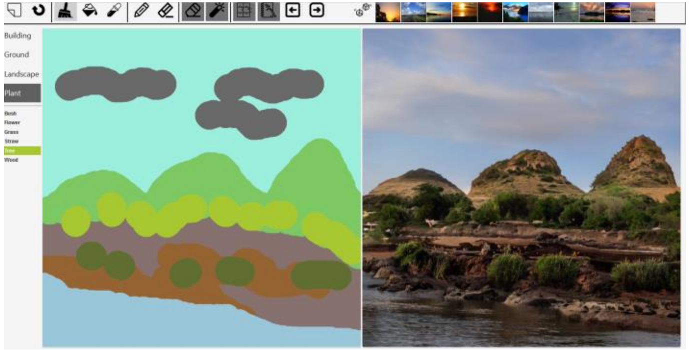
GAN Lab

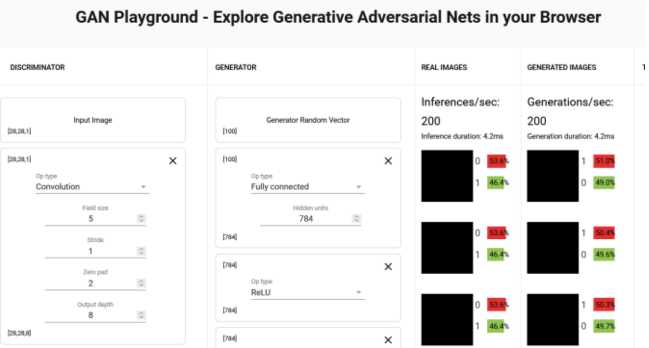
Epoch 001,931



TEXT PROMPT an armchair in the shape of a doughnut. an armchair imitating a doughnut.

AI-GENERATED IMAGES





GAN Playground - Explore Generative Adversarial Nets in your Browser

DISCRIMINATOR

Input Image [28,28,1]

[28,28,1] Op type: Convolution; Field size: 5; Stride: 1; Zero pad: 2; Output depth: 8; [28,28,8]

GENERATOR

Generator Random Vector [100]

[100] Op type: Fully connected; Hidden units: 784; [784]

[784] Op type: ReLU; [784]

[784]

REAL IMAGES

Inferences/sec: 200

Inference duration: 4.2ms

0	53.6%
1	46.4%
0	53.6%
1	46.4%
0	53.6%
1	46.4%

GENERATED IMAGES

Generations/sec: 200

Generation duration: 4.2ms

1	51.0%
0	49.0%
1	50.4%
0	49.6%
1	50.3%
0	49.7%

Everybody can dance now

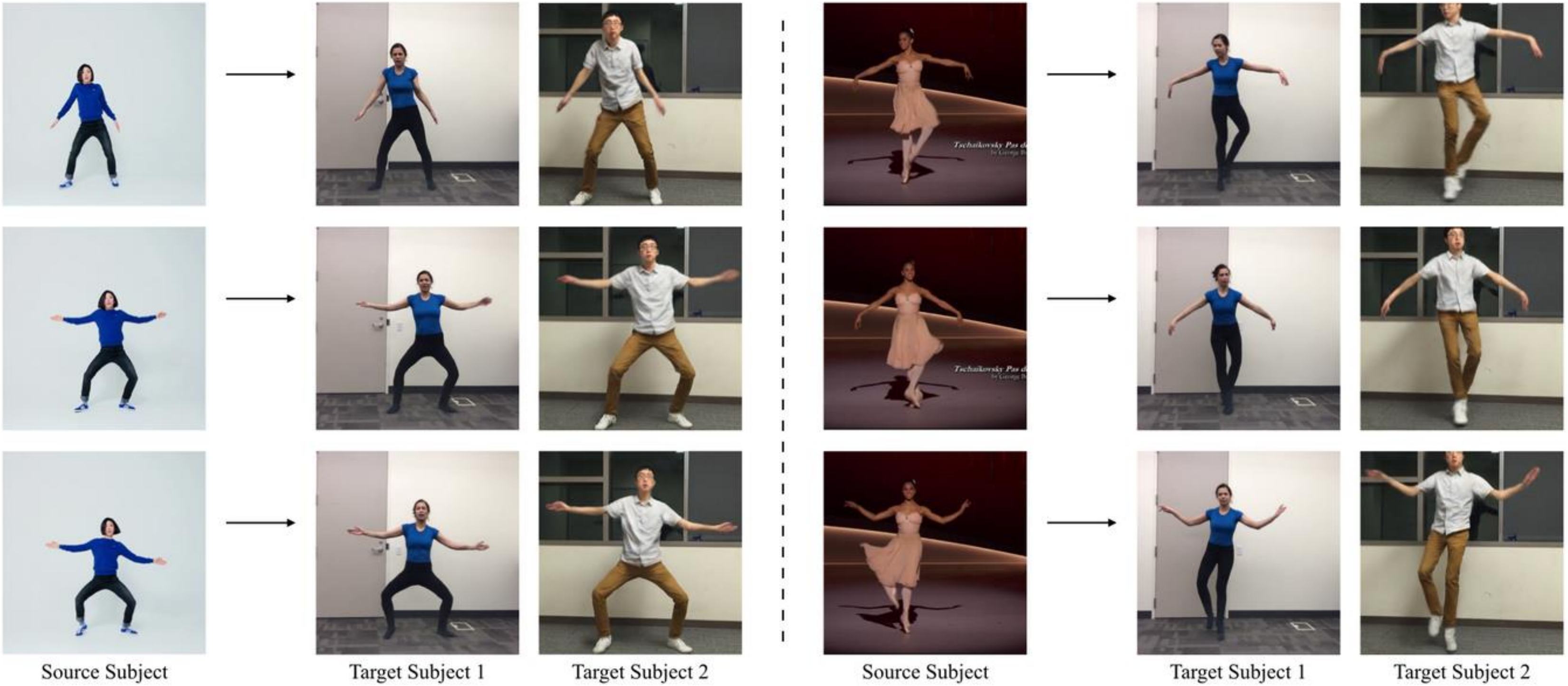
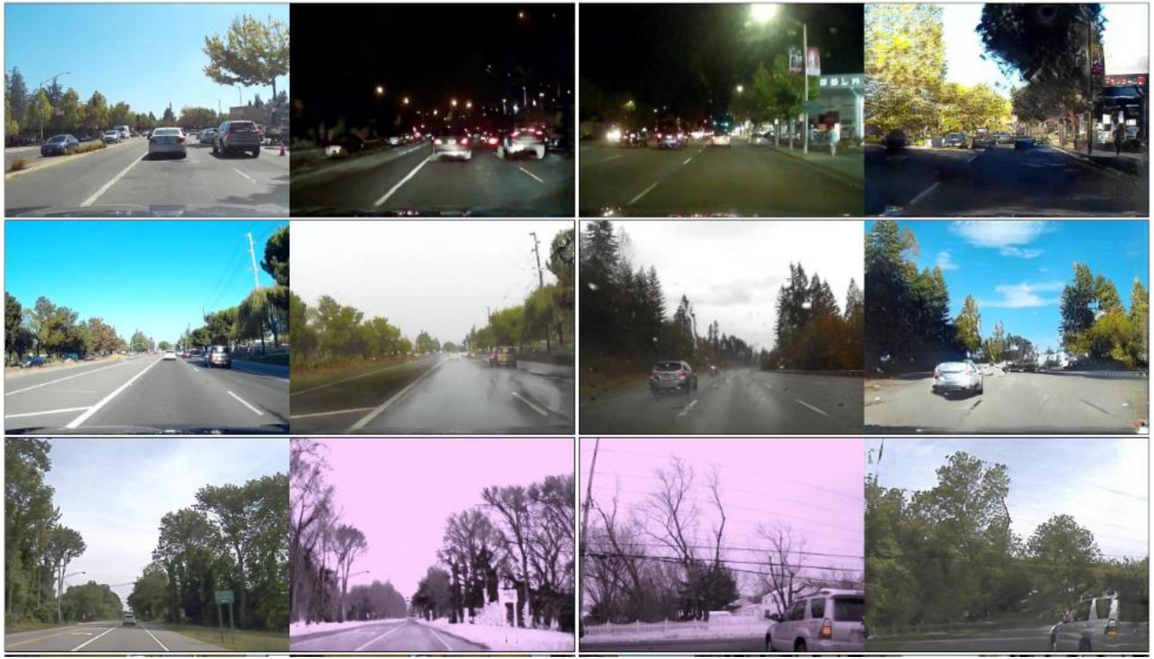
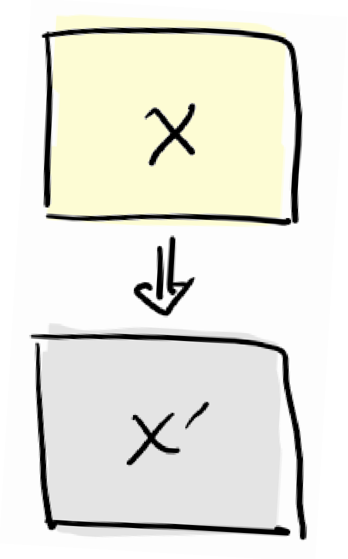


Image Translation

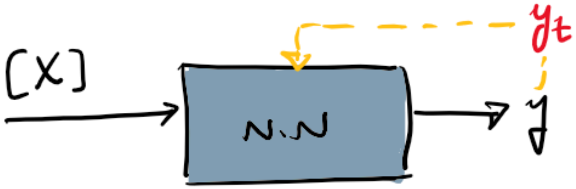


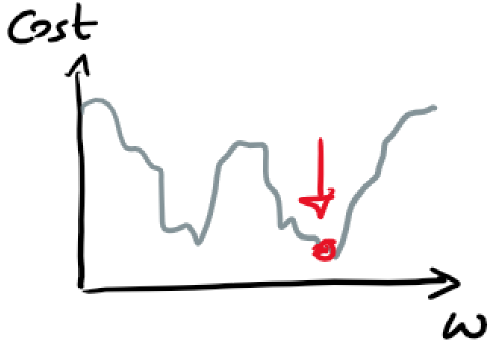


Training GANs

- In MLP, we have a clear goal & measure

  Eg Minimize Cross-entropy loss.





- In GANs, two networks have competing obj. !

  (G) ↑ ; (D) ↓ // (D) ↑ ; (G) ↓

Training GANs

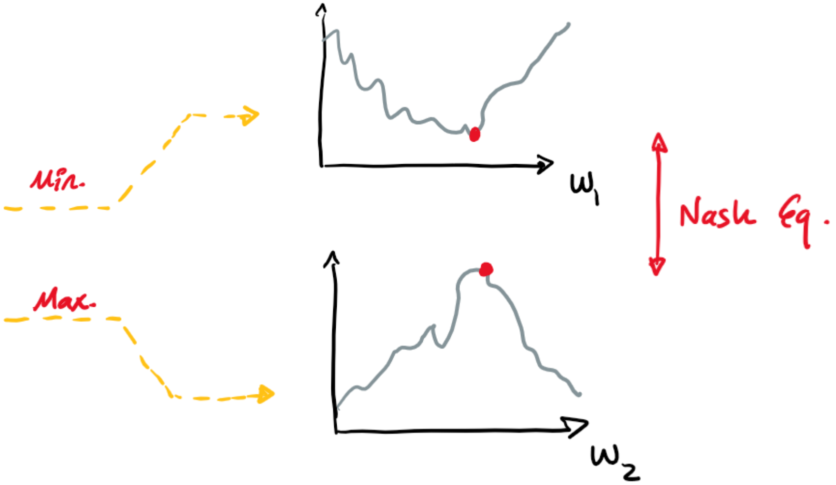
Minmax Game

* Adversarial ML

↳ Game Theory

⊛ Player 1

⊛ Player 2



Training GANs

! Nash Equilibrium := Point where neither "player" can improve their situation

- (G) := Fakes are indistinguishable from Real data
- (D) := at best random guess (F/R ⇒ 1)

! In practice ; ~ impossible to achieve Nash Eq.

it still works ...

Training GANs

Loss Function Likelihood ⇒ Classification problem

$$\text{Value}(G, D, X, y) = \underbrace{E\left(\log D(X)\right)}_{\text{log. probability D correctly predicts reals}} + \underbrace{E\left(1 - D(G(\overset{\text{noise}}{z}))\right)}_{\text{log. prob. D correctly predicts fakes are fakes}}$$

Discriminator ⇒ max. accuracy. of D

Generator ⇒ min. accuracy of D

Training Algorithm:

For each training do:

Train (D):

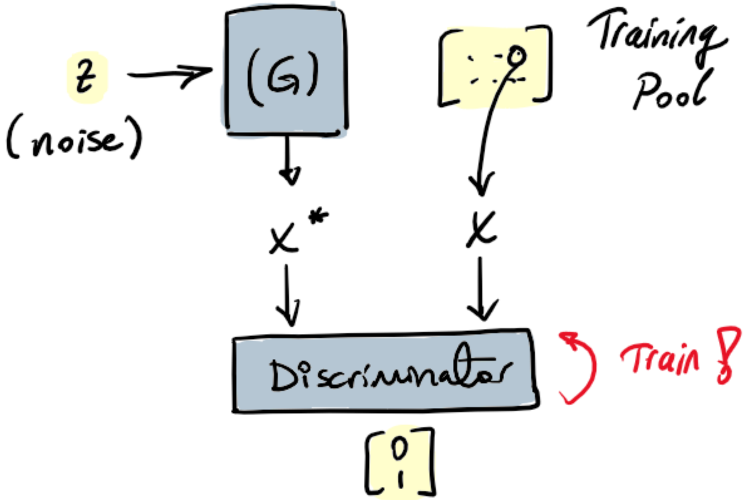
(1) Take a random real example from Training data, x

(2) Get a fake example from Generator, x^*

(3) Use Discriminator to classify x & x^*.

(4) Compute the class. error.

(5) Backprop. error & update Discriminator trainable parameters.



Likelihood ⇒ Classification problem

Training Algorithm:

Train (G):

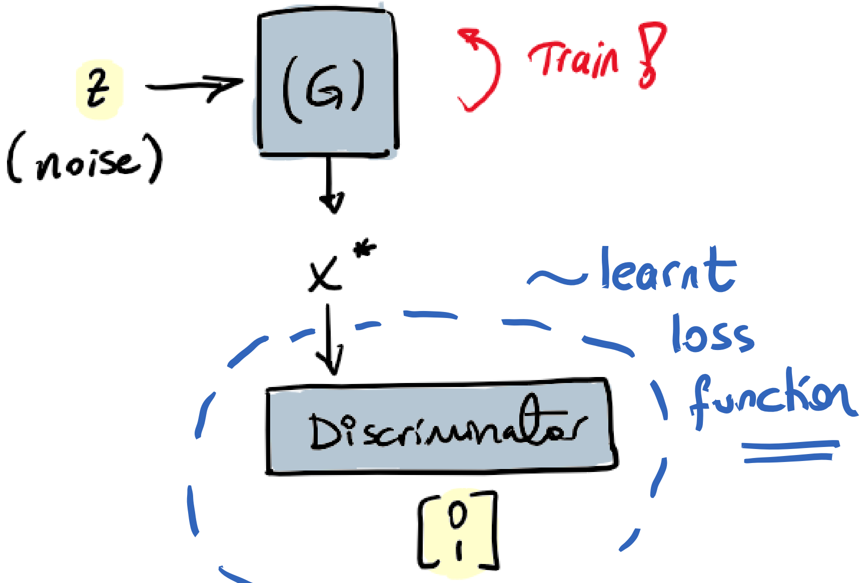
(6) Generate a new fake x^*.

(7) Use Discriminator to classify x^*.

(8). Compute the error.

(9). Update Generator 's trainable parameters via backprop.

end for



Likelihood → Classification problem

Putting it together...

```
for i steps of traning, do:
    for j steps of D traning, do:
        → Sample mini-batch of real $X_i$; $y=1$
        → Create mini-batch of fake $X_i^*$; $y=0$
        → Udpate D Model
              $\frac{1}{N}\sum^{N}\left[\log D(x) + \log(1 - D(G(z)))\right]$
    → Create a set of fake examples $X^z$, $y=1$
    → Update G model
              $\frac{1}{N}\sum^{N}\log(1 - D(G(z)))$
```

Likelihood ⇒ Classification problem

Training GANs

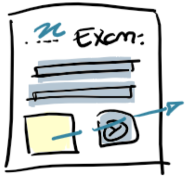
* Training is very difficult

- ■ Disc. usually wins ⇒ make the game unfair ?
- ■ Training with more Epoch. can make it worse.
- ■ More data ⇒ more confusion ⇒ may worsen
- ■ Cross-domain architectures may fail

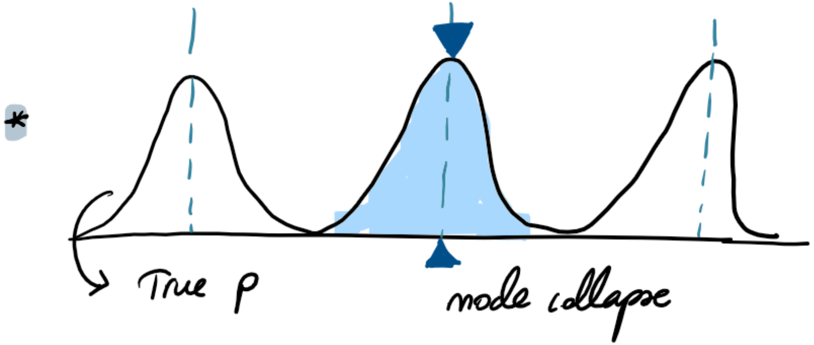
Baby sitting ?

Training GANs

Training problems #1: Mode Collapse

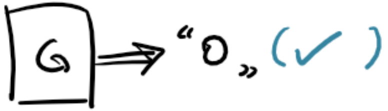


"Abracadabra" ~ "I create as I speak"



$$\begin{bmatrix} -4 \\ -2 \\ 0 \\ +2 \\ +4 \end{bmatrix}$$

True values



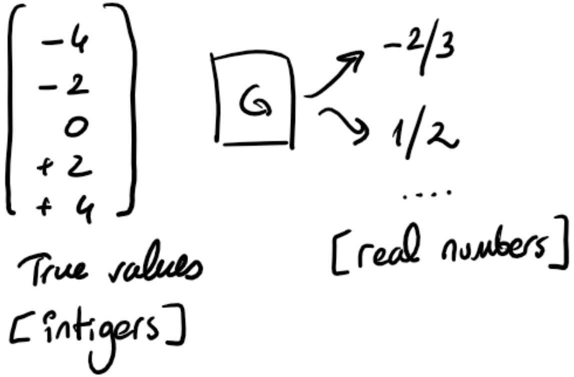
~~[-4, -2, 2, 4]~~

Training GANs

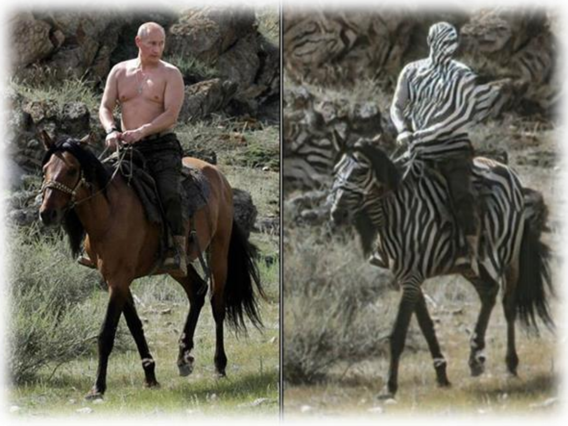
Training Problem #2 : Over generalization

- * Mods that should not exist, do exist.

*

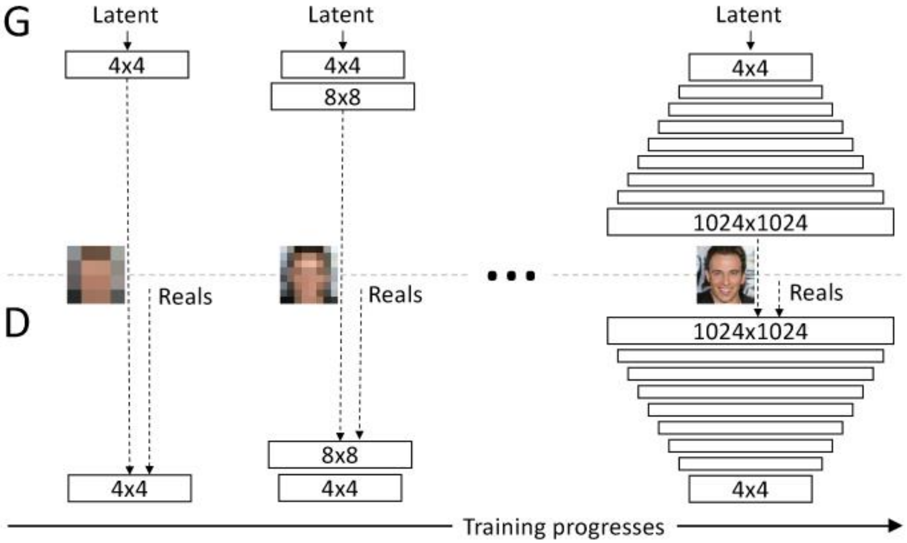


* Image generation



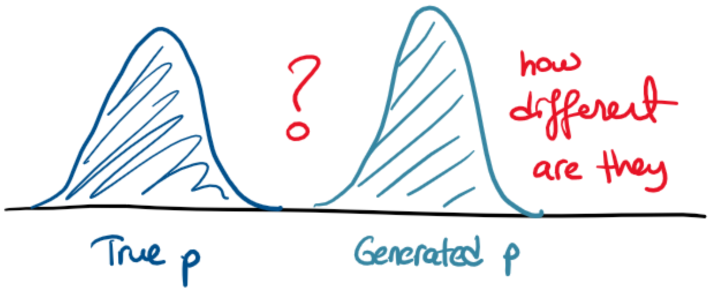
Possible Remedies

① Growing the network gradually

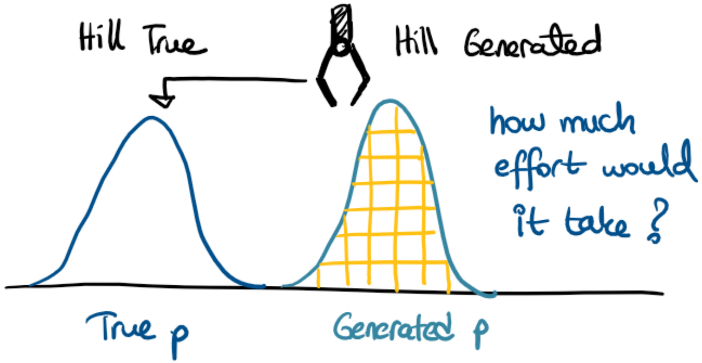


Possible Remedies

② Alternative loss definitions ⇒ Wasserstein GAN



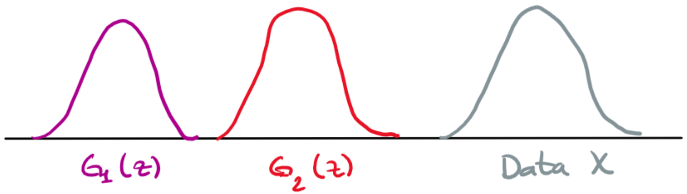
* Distance (Similarity) := (TV distance, KL divergence, JS divergence, ...) ⇒ Earth-mover Distance



Possible Remedies

② Alternative loss definitions ⇒ Wasserstein GAN

Obj: Learn correct representation ⇨ "p"



How similar they are?

- KL Divergence ⇒ no overlap ⇒ ∞
- JS Divergence ⇒ no overlap ⇒ finite, same value
- W Distance ⇒ "G_2 is better than G_1".

$$E(\log D(x)) + E(\log(1 - D(G(\overset{\text{noise}}{z}))))$$

$$\Downarrow$$

$$E(D(x)) - E(D(G(\overset{\text{noise}}{z})))$$

Score ⇒ Regression Problem

Possible Remedies

② Alternative loss definitions ⇒ Wasserstein GAN

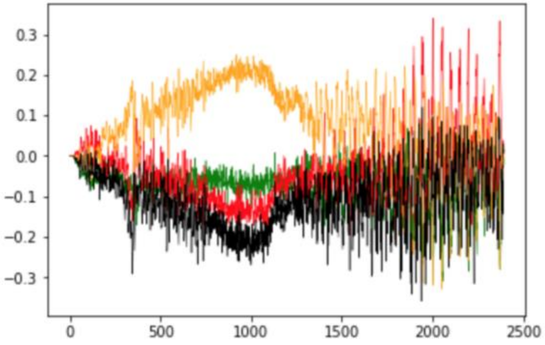
* Class. ⇒ [0, 1]

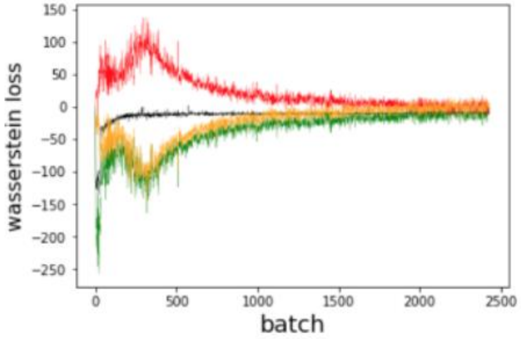
↳ Train... ⇒ 0.9971 ‖ 0.0004
0.9965 ‖ 0.00013
0.9981 ‖ 0.00021
...

⇓

vanishing gradient problem

* Stabilized learning





I/O in Music

① Symbolic Representation

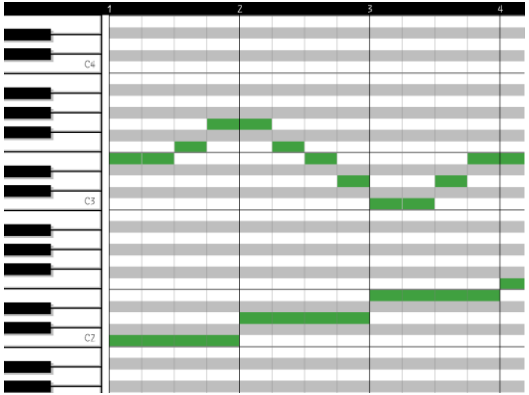
* Piano Rolls

↓

"image like"

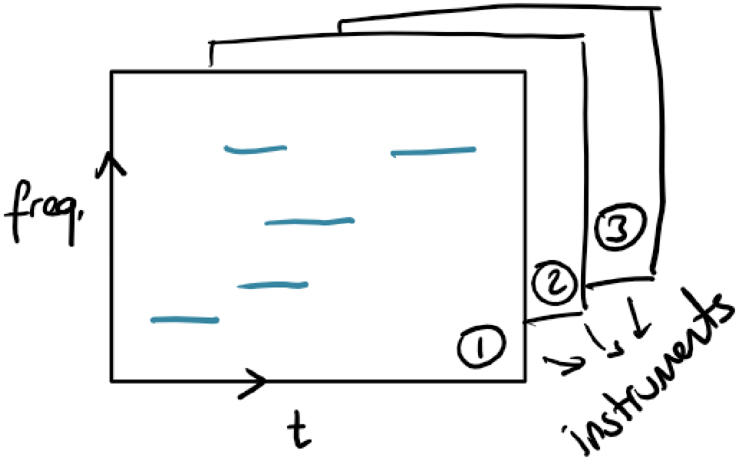
Eg:

- MuseGAN
- MidiNET



⇒

② Audio



Possible Remedies

To Do List

(i) Create D model

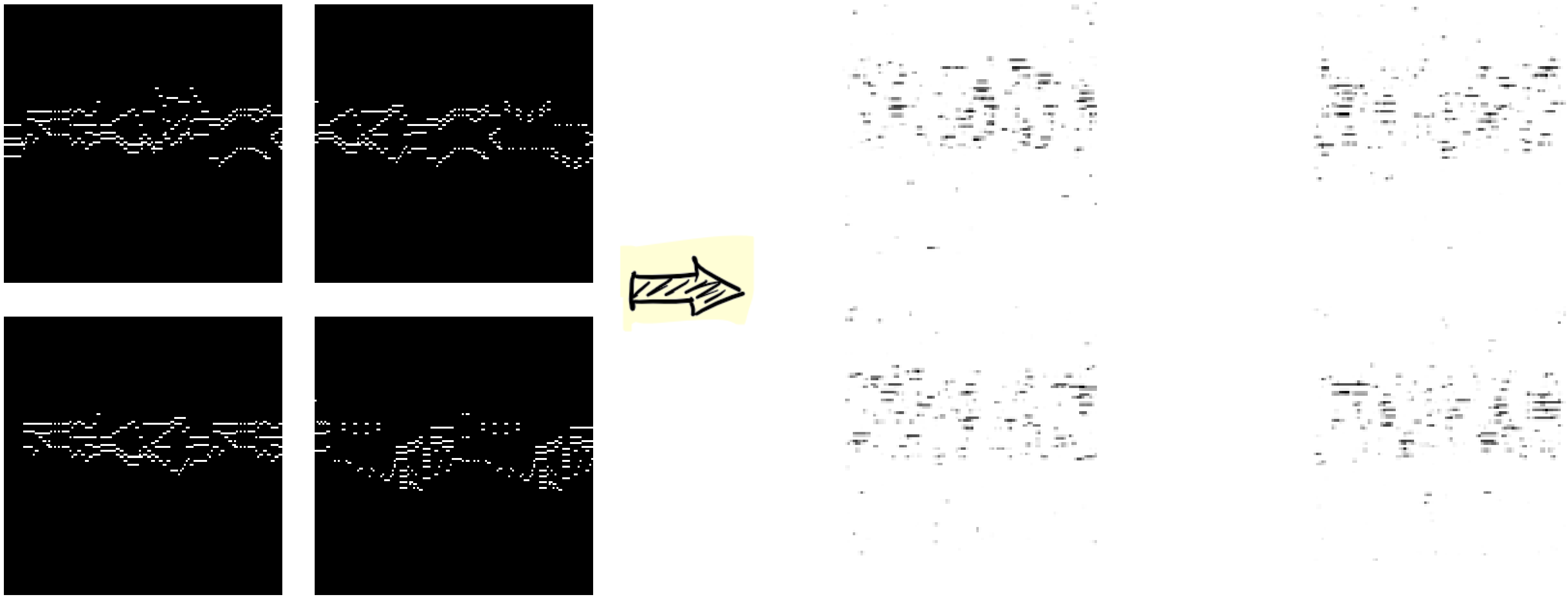
- W ⇒ linear act. function

(ii) Create G model

- Final layer → tanh [-1, 1]

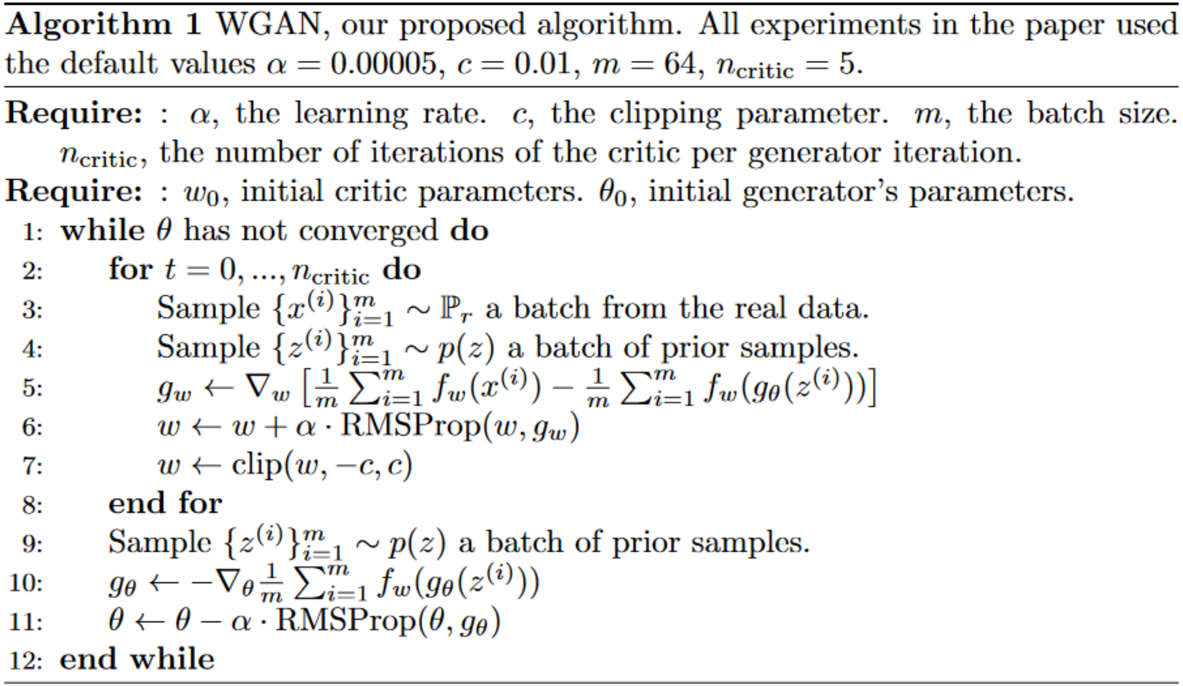
(iii) Add weight clipping in D for W

I/O in Music



Ateliers & Saveurs in Montreal

colab



Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

```
Require: : $\alpha$, the learning rate. $c$, the clipping parameter. $m$, the batch size.
    $n_{\text{critic}}$, the number of iterations of the critic per generator iteration.
Require: : $w_0$, initial critic parameters. $\theta_0$, initial generator's parameters.
while $\theta$ has not converged do
    for $t = 0, ..., n_{\text{critic}}$ do
        Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_r$ a batch from the real data.
        Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
        $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right]$
        $w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w)$
        $w \leftarrow \text{clip}(w, -c, c)$
    end for
    Sample $\{z^{(i)}\}_{i=1}^m \sim p(z)$ a batch of prior samples.
    $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^m f_w(g_\theta(z^{(i)}))$
    $\theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta)$
end while
```