shorturl.at/hoA38 ← Tutorial web page

Generative Models: Part II

Advanced GAN Techniques & Application in Particle Physics

Wasserstein GANs

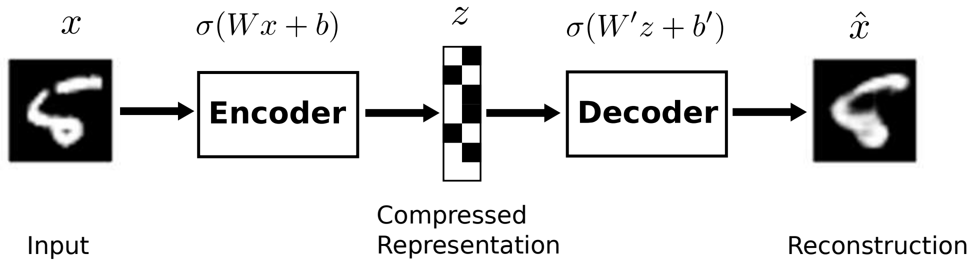
Jonas Glombitza, Martin Erdmann

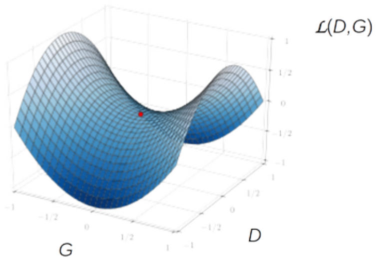
RWTH Aachen

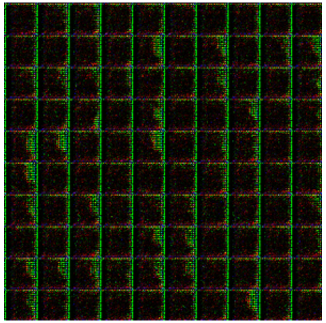
23.03.2022
Deep Learning Weeks, Uppsala

Recap – Generative Models

- **Variational Autoencoder**
  - Hidden representation follows low dimensional arbitrary prior distribution
  - Trained decoder part can be used as generator to produce new samples







- **Generative Adversarial Networks**
  - Hand-coded loss is replaced by discriminator (tries to discriminate between fake samples and real samples)
  - Adversarial training: generator and discriminator trained “against” each other
  - Generator tries to fool discriminator

Outline

Monday

I. Deep Learning Basics – short hands-on

II. Convolutional Neural Networks – short hands-on

BREAK

III. Introduction to Generative Adversarial Networks (GANs)

IV. Tutorial: Implementation of GANs

Wednesday

V. Latest developments & advanced techniques
   - Wasserstein GANs

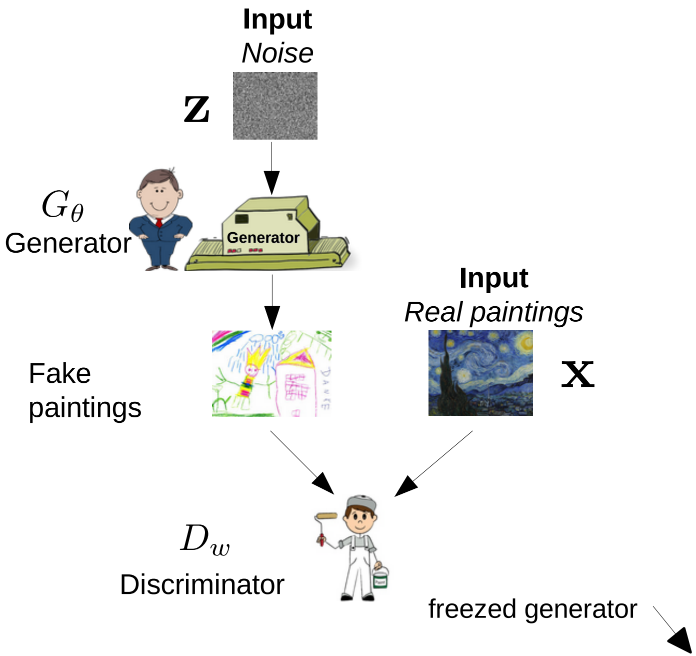
VI. Application in physics research
   - Simulation acceleration, domain adaption

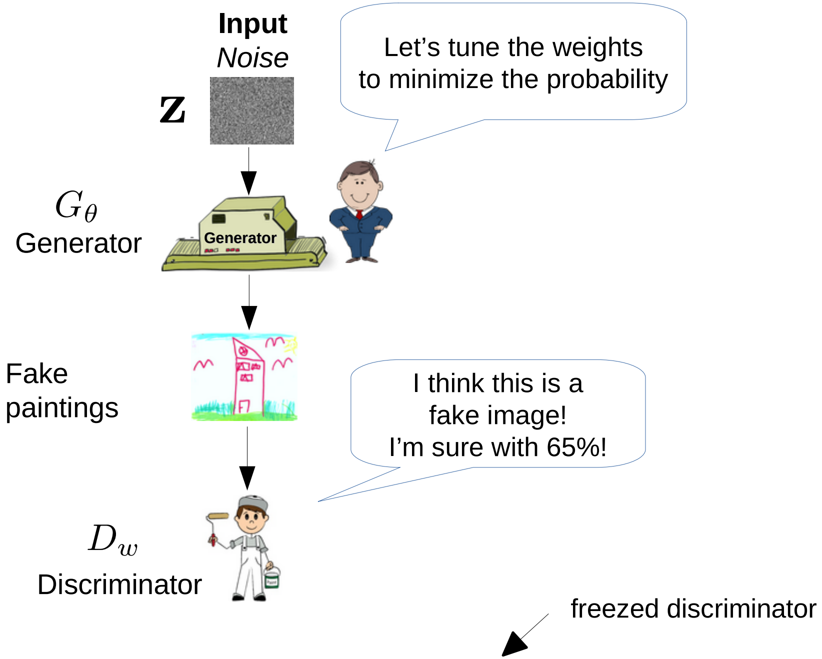
BREAK

VII. Tutorial: Implementation of Wasserstein GANs

Feel free to ask questions during the seminar!
Just “raise” your hand...

Recap – Generative Adversarial Networks





$$\mathcal{L} = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[log(D_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(1 - D_w(G(\mathbf{z})))]$$

$$\mathcal{L} = \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(1 - D(G_\theta(\mathbf{z})))]$$

Adversarial framework:

- Train generator to fool discriminator with fake samples, train discriminator to detect fake samples
- Iteratively (after each **batch** update) update generator and discriminator

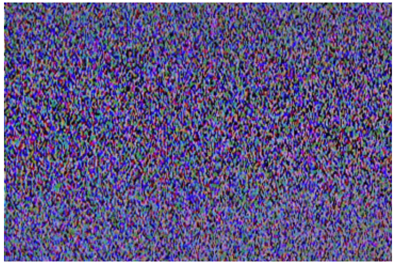
Recap - Manifold Hypothesis

Idea: Manifolds of meaningful pictures are highly concentrated with very little volume and embedded in a very high dimensional space

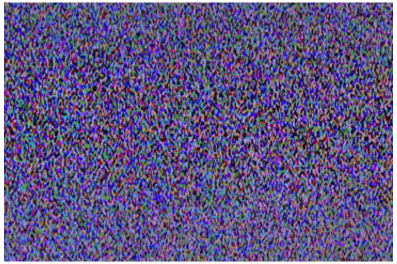
- Generation of images is a very challenging task
- Correlations / probability dimension are high dimensional

- **Example:** Try to generate images randomly:

Goal



Sample 1

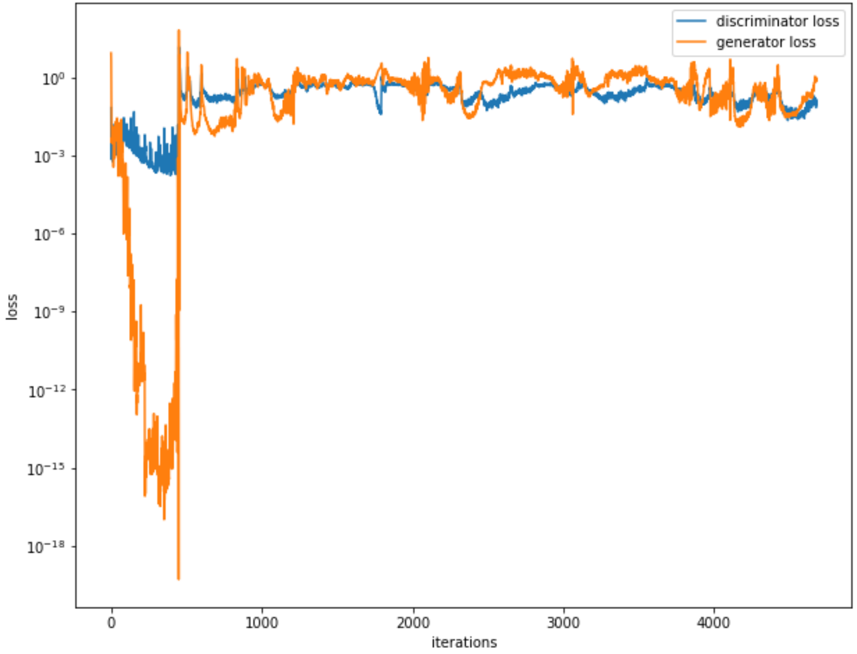


Sample 100,000

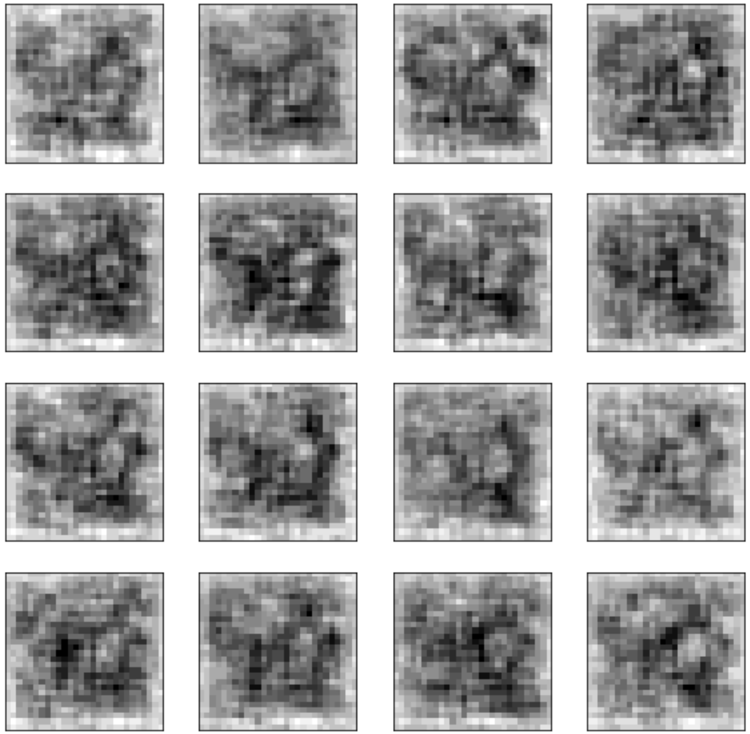
You will even never reach this “neighborhood sample”

“To deal with a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everyone does it.” *- G. Hinton*

Results Fashion MNIST



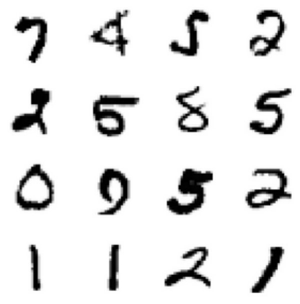
Iteration 1

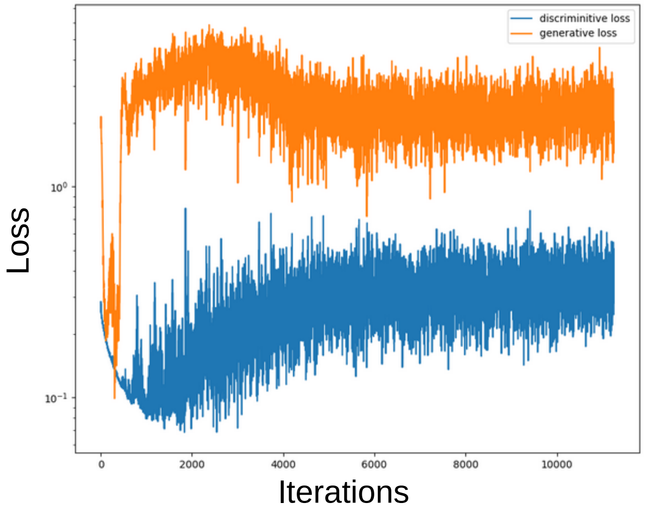


- Challenging training!
- “Weak convergence” after 3000 iterations
- Complex models show very instable training

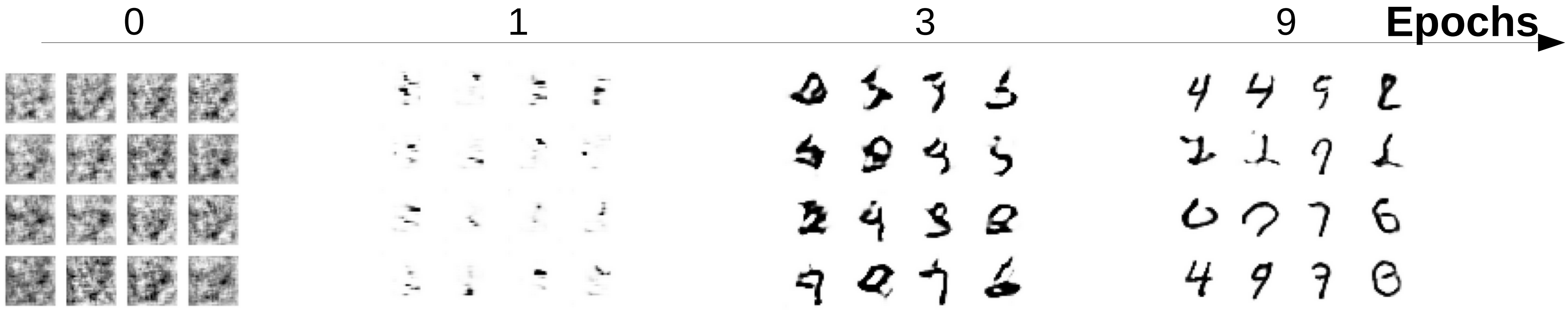
Results

Real images





- GAN produces meaningful images after ~5 epochs
- Loss do not correlate with image quality
- Not stable training



Interpreting the Adversarial Loss

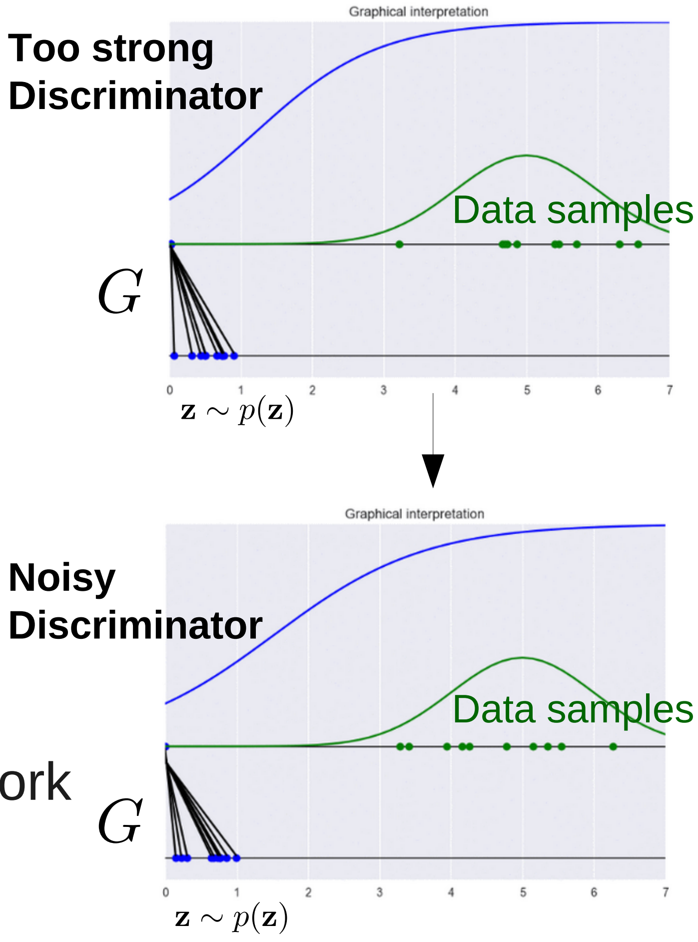
Emanuele Sansone: Tutorial on Generative Adversarial Networks (GANs) - GitHub

- GANs are hard to train → Nash equilibrium
  - generator ⟷ discriminator

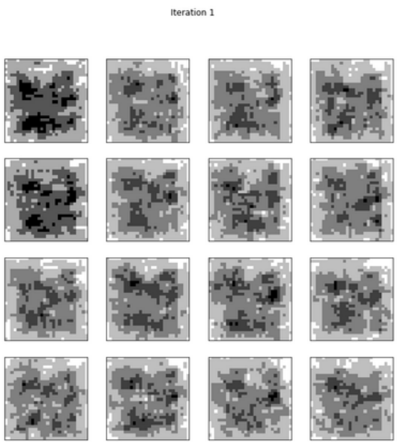
- Loss is hard to interpret (depends on discriminator)
  - no correlation with image quality

- **Strong discriminator → vanishing gradients**

➢ Best: generator and discriminator on same scale
  - Inexact noisy training → Rarely converging framework

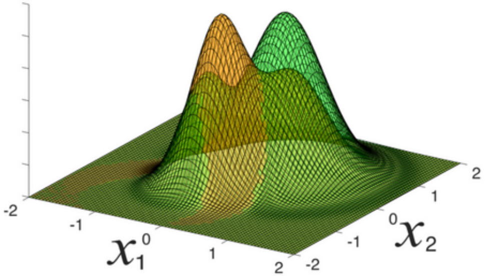


Mode Collapsing - Helvetica Scenario

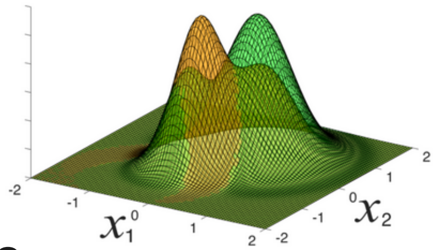


Problem: GANs often suffer from *mode collapsing*

- Many $\mathbf{z} \sim p(\mathbf{z})$ collapse towards restricted space in P_r
  - Generator produce samples of a limited phase space
  - Example: generate only digits 1 and 8
- Discriminator feedback is insensitive to complete phase-space
  - Will focus on point(s) of phase space the generator do not cover
- Discriminator will push generator to this mode → cycling behavior
- **Need different (softer) metric to address these issues!**



GAN Objective



- By fooling the discriminator the generator minimize distribution differences → $P_\theta \approx P_r$

- GAN training similar to minimizing Jensen-Shannon divergence (assume optimal discriminator)

$$\mathcal{D}_{JS}(P_r || P_\theta) = \mathcal{D}_{KL}(P_r || P_m) + \mathcal{D}_{KL}(P_\theta || P_m) \qquad P_m = \frac{1}{2}(P_r + P_\theta)$$

✔ Symmetrized and smoothed version of the Kullback-Leibler divergence

✗ Fails to provide a meaningful value when two distributions are disjoint

- In very high dimensional manifolds the distributions between generated and real samples are disjoint

Wasserstein Distance

- Also known as Earth Mover’s distance (EMD)

Ensures smallest cost

Traveling distance

$$\mathcal{D}_W(P_r||P_\theta) = \inf_{\gamma \in \Pi(P_r, P_\theta)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]$$

Transportation plans

- Describes **minimal cost** to move distribution P_θ on P_r and vice versa
  - Cost: mass * distance

Distribution Similarity - Metrics

- Kullback-Leibler divergence
  - ✗ Not finite, not symmetric

$$\mathcal{D}_{KL}(P_r||P_\theta) = \mathbb{E}_{\mathbf{x}\sim P_r} log\left(\frac{P_r}{P_\theta}\right)$$

- Jensen-Shannon divergence

$$\mathcal{D}_{JS}(P_r||P_\theta) = \mathcal{D}_{KL}(P_r||P_m) + \mathcal{D}_{KL}(P_\theta||P_m) \qquad P_m = \frac{1}{2}(P_r + P_\theta)$$

  - ✔ Symmetric

- Wasserstein distance
  - ✔ Symmetric
  - ✔ Meaningful distance measure for disjoint distributions

For disjoint distributions:

$$\mathcal{D}_{KL}(P_\theta||P_r) = \infty$$
$$\mathcal{D}_{KL}(P_r||P_\theta) = \infty$$
$$\mathcal{D}_{JS}(P_r||P_\theta) = \log(2)$$

In GAN training we are dealing with disjoint distributions!

Kantorovich-Rubinstein Duality

- Wasserstein distance (formal definition very intractable) can be expressed

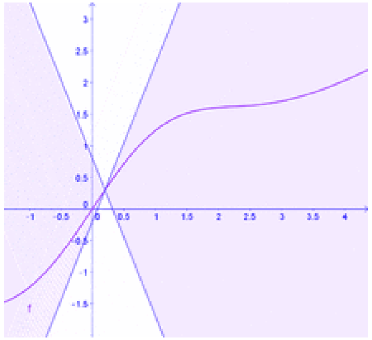
$$\mathcal{D}_W(P_r||P_\theta) = \sup_{f \in Lip_1} \mathbb{E}_{x\sim P_r}[f(x)] - \mathbb{E}_{\tilde{x}\sim P_\theta}[f(\tilde{x})]$$

- supremum = least upper bound
- f = Set of 1-Lipschitz functions

- $\mathbb{E}_{x\sim P_r}[f(x)]$ - Expectation value when applying set of 1-Lipschitz functions on samples from real samples

> Approximate $f \approx f_w$ with neural network

1-Lipschitz functions



Slope everywhere less equal 1!

The WGAN Concept

$$\mathcal{D}_W(P_r||P_\theta) = \sup_{f\in Lip_1} \mathbb{E}_{x\sim P_r}[f_w(x)] - \mathbb{E}_{\tilde{x}\sim P_\theta}[f_w(\tilde{x})]$$

Real samples

Generated samples $\tilde{x} = G_\theta(z)$

f_w = neural network (discriminator → critic)

- Neural network carries the Lipschitz continuity constraint
  - Critic network estimate Wasserstein distance between generate and real samples

To paint more realistic images: Just change your brush!

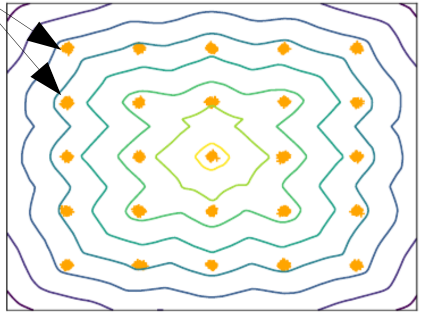
Discriminator

Critic

Gradient Penalty

- Implement Lipschitz constraint
  - Build up space for meaningful discriminator feedback
- Without Lipschitz constrain
  - Critic will not converge → **No Wasserstein!**

Targets



Value surfaces of critic

Extend objective with additional term:

- Penalize gradients being different from 1

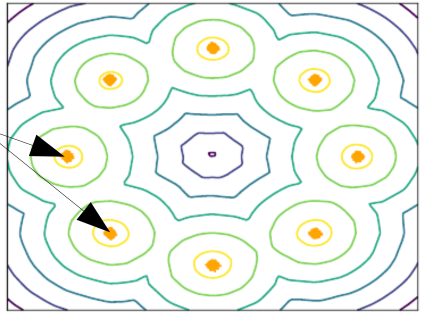
$$\mathcal{L}_{GP} = \lambda\, \mathbb{E}_{\hat{u}\sim P_{\hat{u}}}[(||\nabla_{\hat{u}} f_w(\hat{u})||_2 - 1)^2]$$

hyperparameter

- Sample gradients along line between event mixture $\hat{u}$

$$\hat{u} = \epsilon x + (1-\epsilon)\tilde{x} \qquad 0 \leq \epsilon \leq 1$$

Targets



Value surfaces of critic

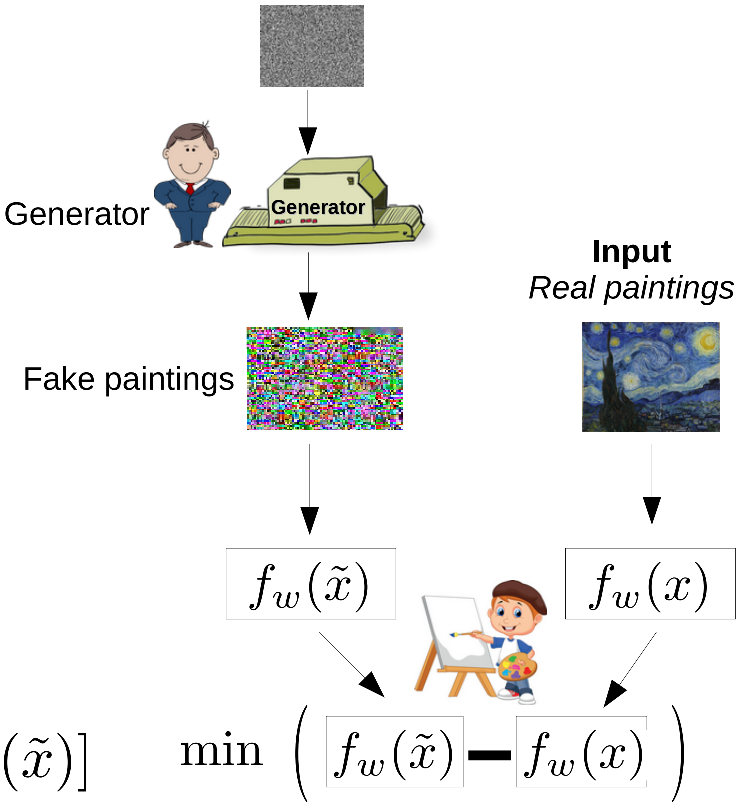
Critic

- **Critic** approximates Wasserstein distance
  - Carries Lipschitz constraint
  - Ensures meaningful and stable gradients

- No explicit formulation of loss function
  - Approximate loss function itself
  - Maximize difference $|f_w(\tilde{x}) - f_w(x)|$

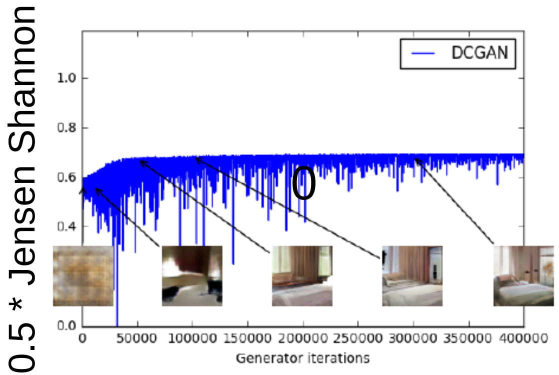
- Critic should be always trained to convergence
  - Usually ~ 5 – 10 iterations

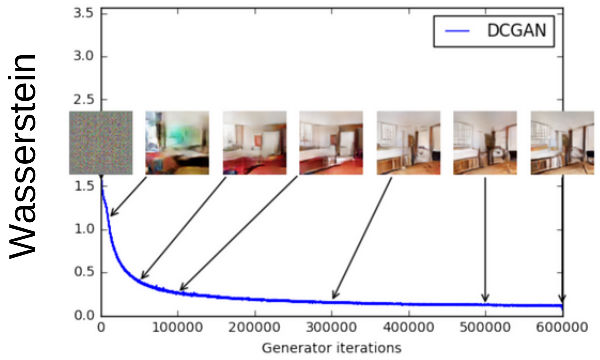
$$\mathcal{D}_W(P_r||P_\theta) = \sup_{f \in Lip_1} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{\tilde{x} \sim P_\theta}[f_w(\tilde{x})]$$

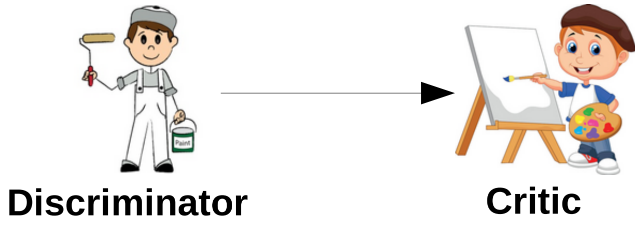


Advantages WGAN vs. GAN

- Train critic to convergence – ensure quality gradients
- Insensitive to mode collapsing
- Meaningful metric / objective → allow for easy hyperparameter search
  - Convergence correlates with generation quality
- Change from Jensen Shannon divergence to Wasserstein-1
- We get feedback from an art expert!

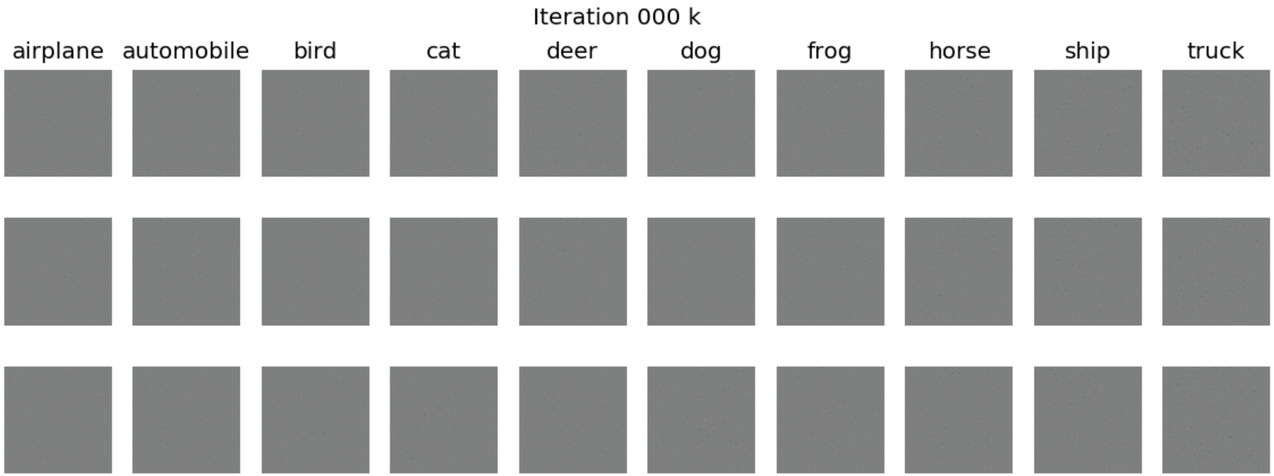






Results

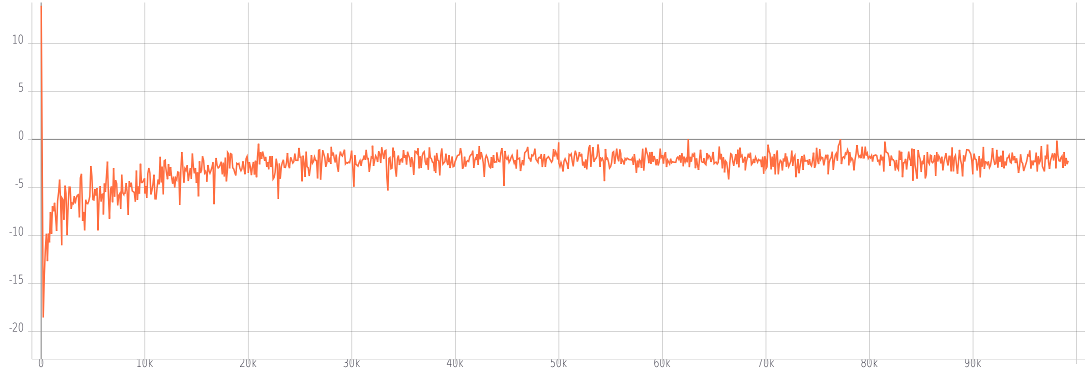
- WGAN generates images with much better quality
- Critic loss converges
- Loss correlates with images quality



Wasserstein GANs

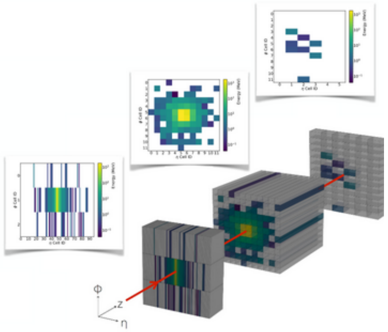
- Allow stable training of GANs
  - Train critic to convergence
  - Precise feedback for generator
- Prevent mode collapsing
- Provide meaningful loss

Critic loss

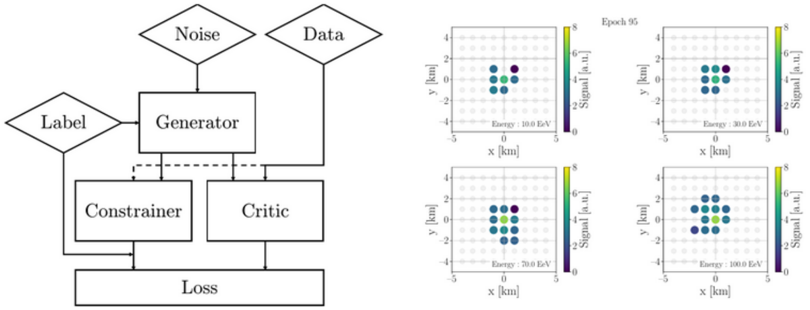


Application in Particle Physics

- Detector simulation are very time consuming
- Replace simulation programs like Geant4 with generative model
  - Reach speed-up of factor $10^3 – 10^5$
- Add constrainer networks to condition the generation process
  - Generator needs dependence (energy, particle type...)
  - Samples must comply with physics laws



Paganini, Oliviera, Nachman - https://arxiv.org/abs/1712.10321

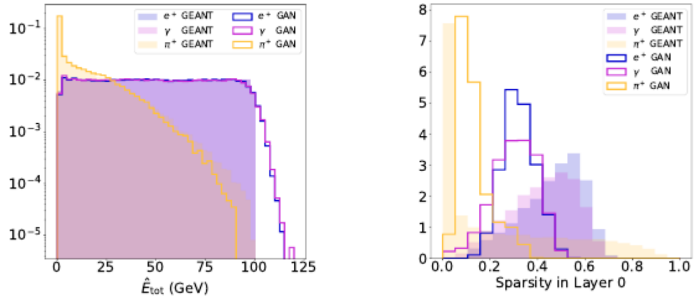


Erdmann, Geiger, Glombitza, Schmidt - https://arxiv.org/abs/1802.03325

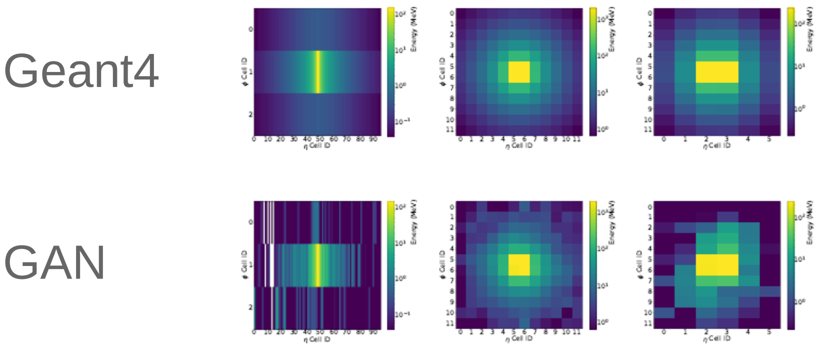
Generation of Calorimeter Images

- Quality of images is crosschecked using physics observables
- Challenges: Sparsity, logarithmic intensity distribution

Traditional GAN

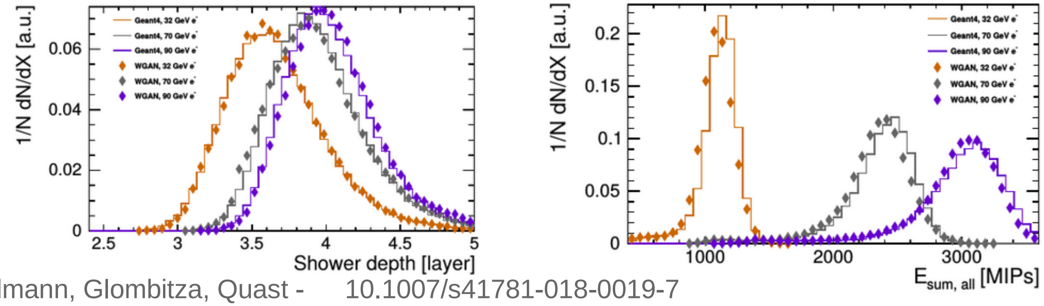


Layer

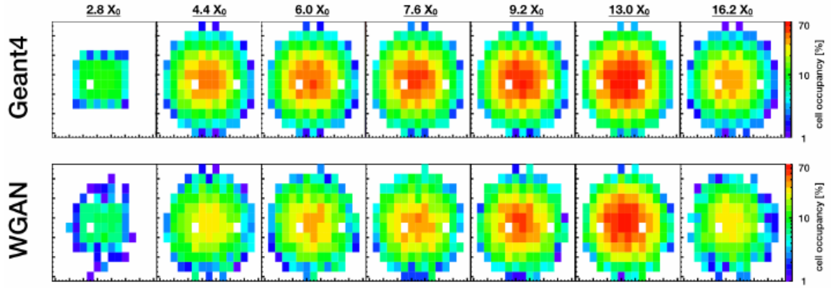


Paganini, Oliviera, Nachman - https://arxiv.org/abs/1712.10321

Wasserstein GAN



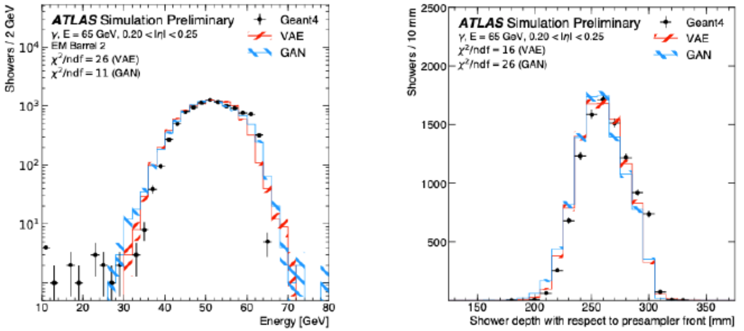
Layer



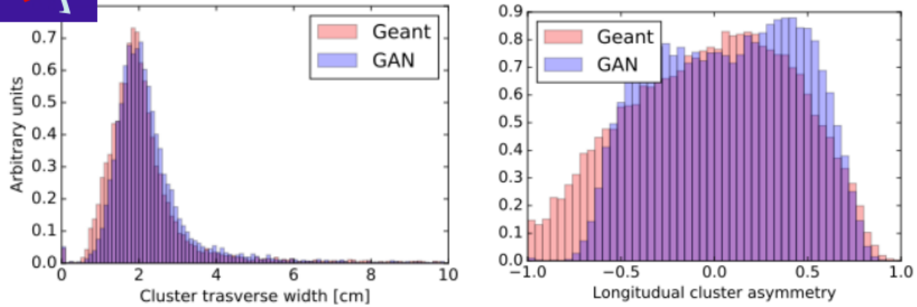
Erdmann, Glombitza, Quast - 10.1007/s41781-018-0019-7

GAN applications for fast calorimeter simulation in other experiments

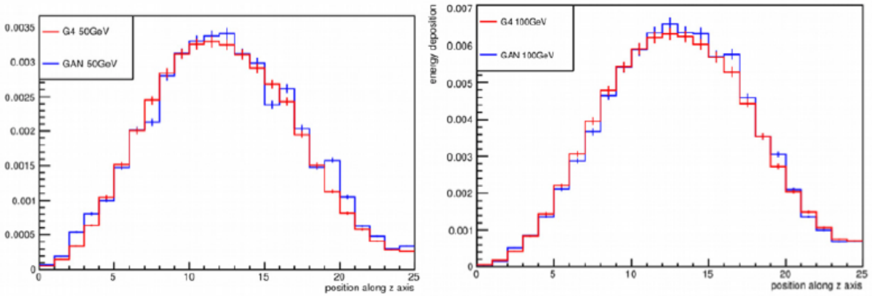
Deep generative models for fast shower simulation in ATLAS, http://cds/2680531



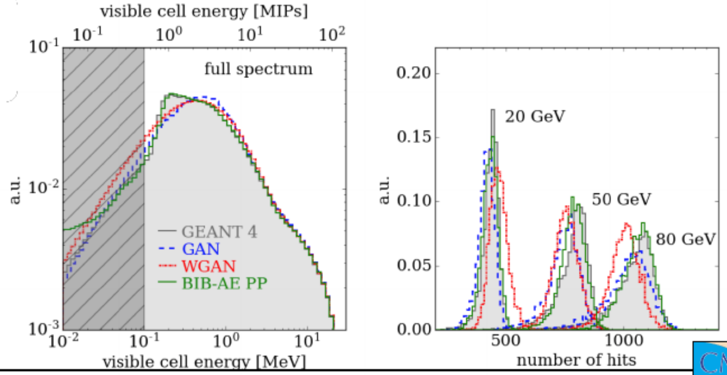
Generative Models for Fast Calorimeter Simulation - LHCb case, [1812.01319](1812.01319)



3D convolutional GAN for fast simulation, https://doi.org/10.1051/epjconf/201921402010



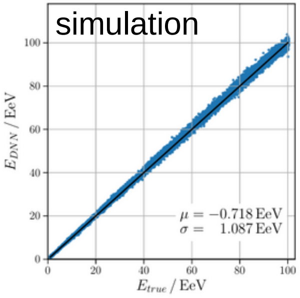
Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed, [2005.05334](2005.05334)



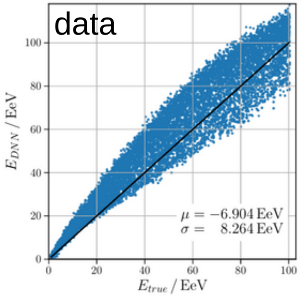
Simulation Refinement

- Simulation data mismatches
  - Predictive models can be sensitive to artifacts / mismatches existing in simulation

DNN evaluated on simulation



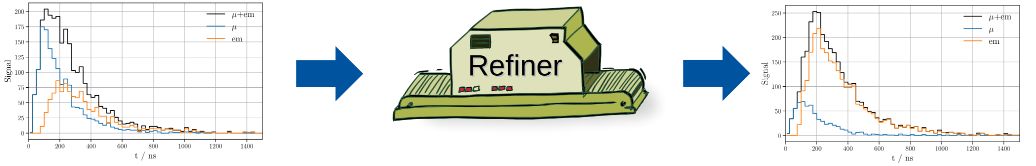
DNN evaluated on data



- Can lead to reconstruction errors

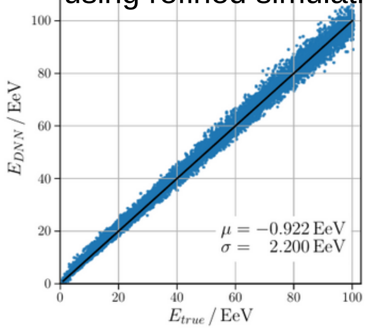
- Use adversarial networks with refinement constraint
- Train *refiner* to refine simulations

Simulated time trace | Refined time trace



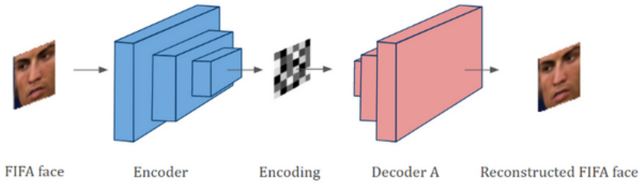
Erdmann, Geiger, Glombitza, Schmidt - https://arxiv.org/abs/1802.03325

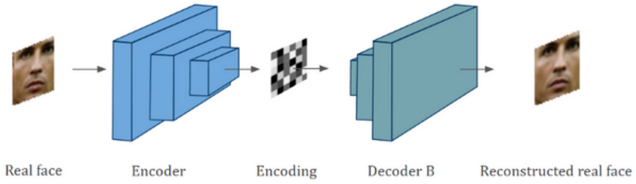
DNN evaluated on data using refined simulation



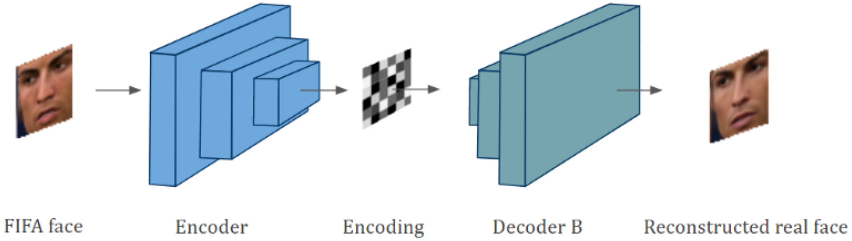
Simulation Refinement

- Use auto encoder set up to mitigate data / simulation differences





- Simulation and data share encoder but different decoder (similar representation)
- After training: refine simulation with decoder trained to reconstruct data



Chintan Trivedi: Using Deep Learning to improve FIFA 18 graphics – Towards Data Science

Spectral Normalization for GANs

- **Gradient penalty / regularization is most important for training GANs!**
- WGAN-GP is state of the art
- Adapt Lipschitz constraint using Gradient “normalization” (penalty)
  - Also *standard (NS-GAN)* with *gradient penalty* performs well!

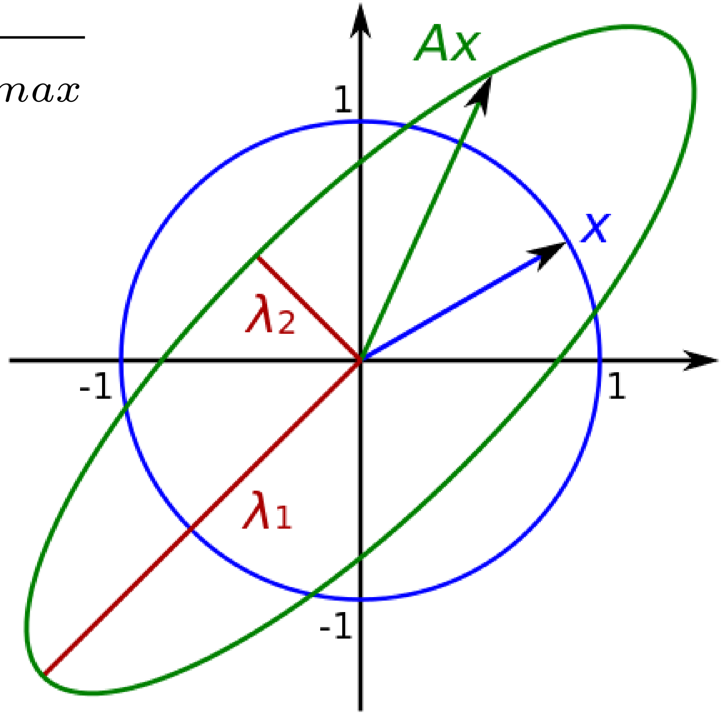
- Adapt Lipschitz constraint in the weights using the *spectral norm*

- Critic in WGAN-GP needs many iterations → slow training
- Spectral norm can be fast approximated using power iteration method
- ➢ Increased stability (high learning rates, high momentum rates)

Spectral Norm

- Spectral norm: “natürliche Matrixnorm”

$$||\mathbf{W}||_2 = \max_{x \neq 0} \frac{||\mathbf{W}x||_2}{||x||_2} = \max_{||x||_2=1} ||\mathbf{W}x||_2 = \sqrt{\lambda_{max}}$$



- Maximum stretch factor of unit vector after multiplication with matrix

- $\lambda_1 = \lambda_{max}$ = highest singular value (“Singulärwert”) of the matrix

Spectral Normalization for GANs

- $D(x)$ = discriminator
- Adapt WGAN-GP constraint (gradient wrt. x real and fake samples)
  - Use **spectral normalization** in each layer!

- Basic idea:

$$||D(x)||_{\text{Lip}} = \sup_x \sigma(\nabla_x D(x)) = \sup_x \sigma(\nabla_x W x) = \sigma(W) \longrightarrow W_{\text{norm}} = \frac{W}{\sigma(W)}$$

- Cover Lipschitz constraint by normalizing the weights

- **Gradient update:**
  - Gradient penalizes updates in direction of highest singular value (in each layer)

SNGAN

- **discriminator**
  - discriminator weights cover Lipschitz constraint due to spectral norm
  - “regularize” gradients → mode collapsing unlikely
  - discriminator loss still meaningless → no critic / distance measure

- **generator**
  - Also spectral normalization in generator improves stability
    - enforce harmless mapping

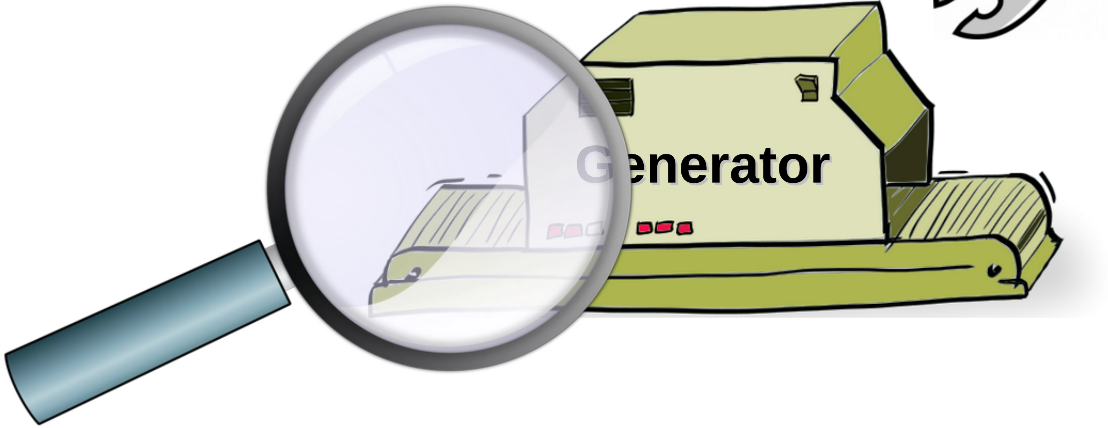
- Framework trained with 1:1 discriminator / generator update ratio

Game: Which face is real?

http://www.whichfaceisreal.com/index.php

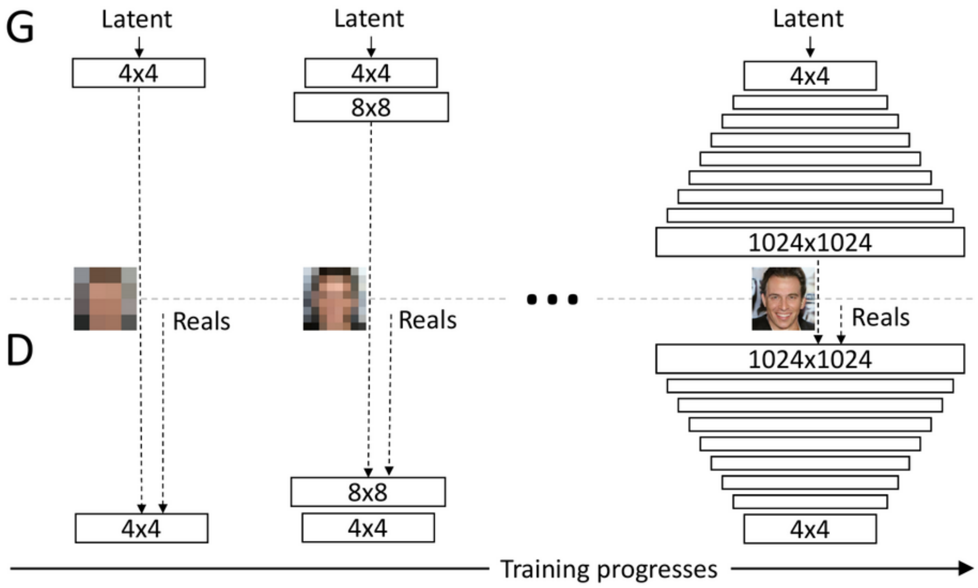
   

Changing the Generator



Progressive Growing

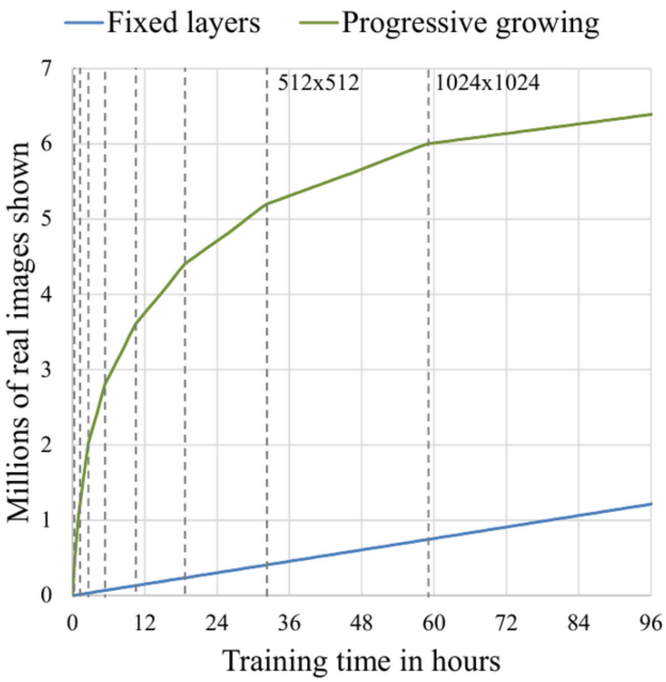
- Separation of training process in several steps
- Increase image resolution stepwise
- Beginning: (low resolution) data set has only few modes
  - small differences to be learned

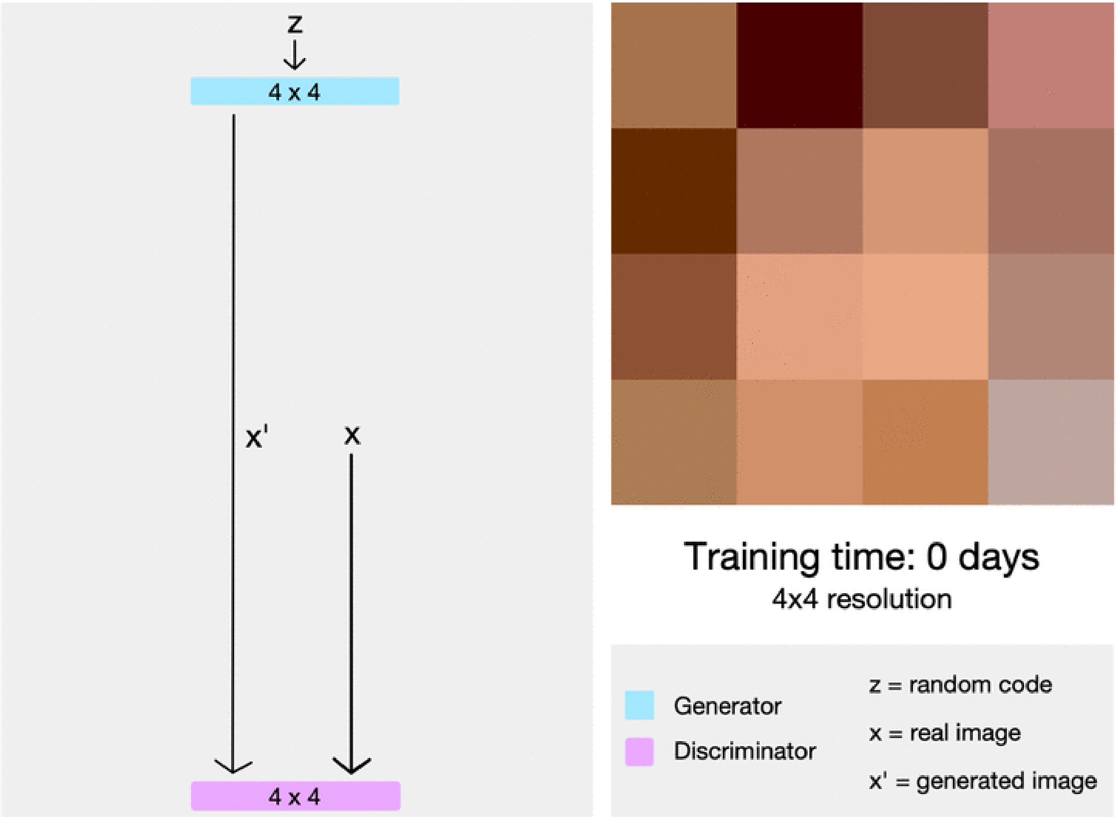


- low resolution
  - learn large scale structure
- High resolution
  - learn fine details

- Speed up
  - Most iteration in the beginning

Progressive Training





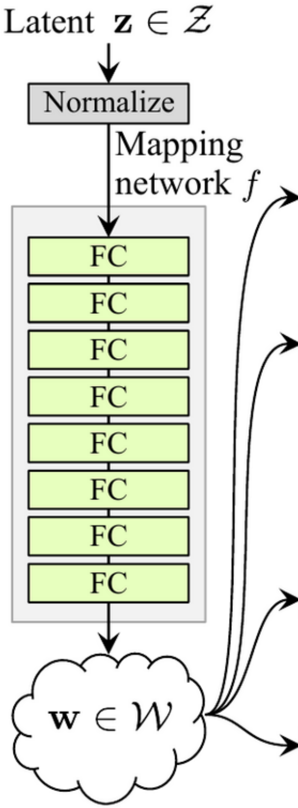
Training on 8 Tesla V100!

StyleGAN

- Image contains of several style levels
- ➢ Change structure of generator
  - disentangle styles in architecture
    - Coarse (pose, face shape)
    - Medium (facial features, eyes closed)
    - Fine (finer hair details, exes)
  - ➢ Learning of high-level attributes
- Add additional noise

- Use medium representation of latent space
  - Use mapping network

High resolution: 1024 x 1024 pixels



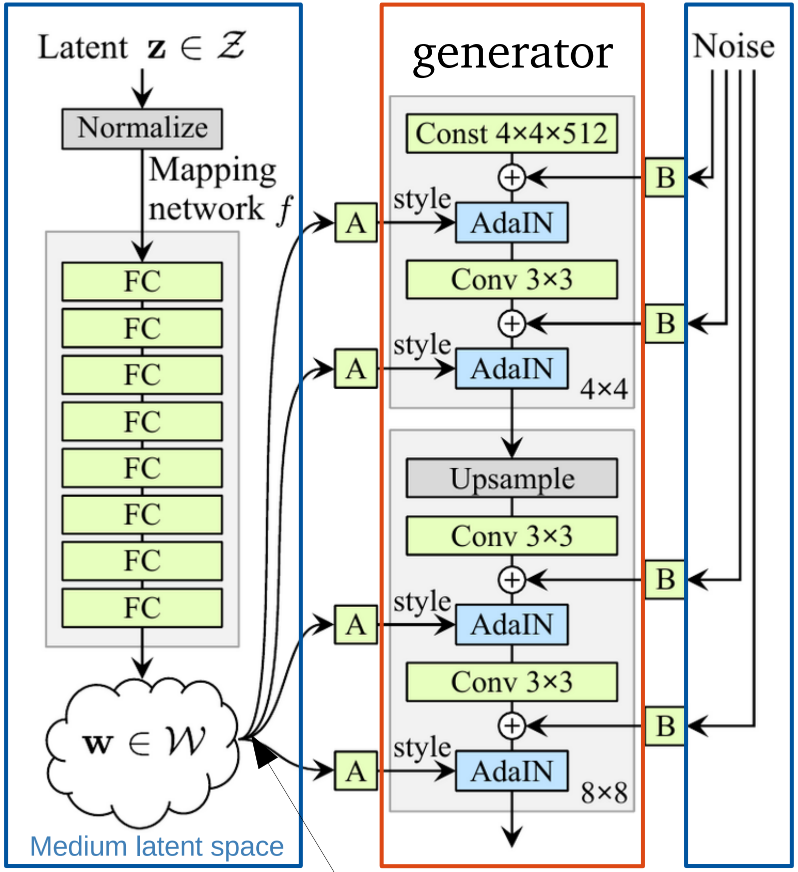
StyleGAN

- Mapping network learns “medium” latent space
  - perceptually smoother latent space
- Generator input:
  - Styles control adaptive instance normalization

$\mathbf{x}_i$ = feature map

$$AdaIN(\mathbf{x}_i, \mathbf{y}) = \mathbf{y}_{s,i} \frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}$$

  - Styles **re-scale features maps** in generator at each representation level
- Add noise at each level of representation
  - Small image variations (natural images)
- Sample from restricted (truncated) phase space
  - Loss off variation → better quality



styles $\hat{=}\ \mathbf{y} = (\mathbf{y}_s, \mathbf{y}_b)$

Video: A Style based Generator

https://www.youtube.com/watch?time_continue=370&v=kSLJriaOumA

Summary

GANs

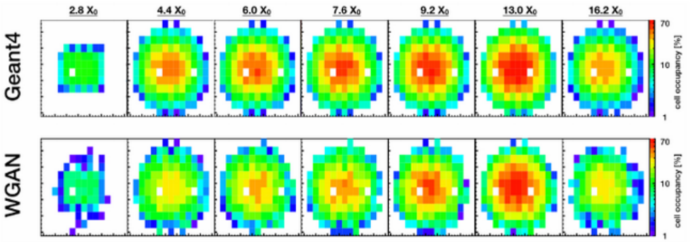
- Delicate, hard to train (mode collapsing, vanishing gradients, meaningless loss)

Advanced techniques – WGAN, ProGAN, SNGAN

- Stabilize training process using smooth metric → meaningful distance measure
- Use regularization: penalize gradient, enforce spectral norm of the weights
- Imply prior on generator architecture (progressive growing, hierarchy control)

Generative Models in Physics Research

- Speed up simulations by factor $10^3 – 10^5$
- Reduce data / simulation mismatches



References & Further Reading

- M. Erdmann, J. Glombitza, G. Kasieczka, U. Klemradt, Deep Learning for Physics Research, World Scientific, 2021, www.deeplearningphysics.org/
- Goodfellow et al.: Generative Adversarial Networks - https://arxiv.org/abs/1406.2661
- Arjovsky, Chintala, Bottou: Wasserstein GAN - https://arxiv.org/abs/1701.07875
- Gulrajani et al.: Improved Training of Wasserstein GANs - https://arxiv.org/abs/1704.00028
- Paganini, Oliveira, Nachman: CaloGAN - https://arxiv.org/abs/1712.10321
- Erdmann, Geiger, Glombitza, Schmidt - https://arxiv.org/abs/1802.03325
- Erdmann, Glombitza, Quast: Calorimeter WGAN - T. Comput Softw Big Sci (2019) 3: 4
- C. Trivedi: Using Deep Learning to improve FIFA 18 graphics - Towards Data Science
- Emanuele Sansone: https://github.com/emsansone/GAN
- Miyato et al.: SNGAN- https://arxiv.org/abs/1802.05957
- T. Karras, S. Laine, T. Aila: A Style-Based Generator - https://arxiv.org/abs/1812.04948

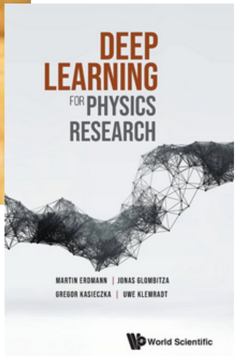
Tryout
Deep Learning
Yourself!

Find many physics examples at:

http://www.deeplearningphysics.org/

For example:

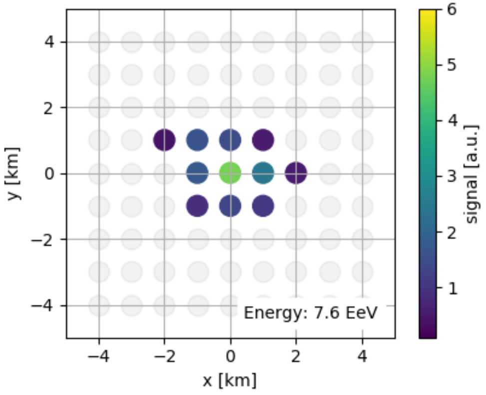
- CNNs, RNNs, GCNs
- GANs and WGANs
- Anomaly detection, Denosing AEs
- Visualization & introspection and more

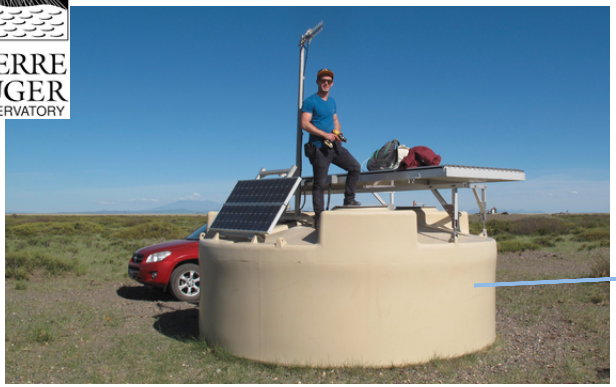


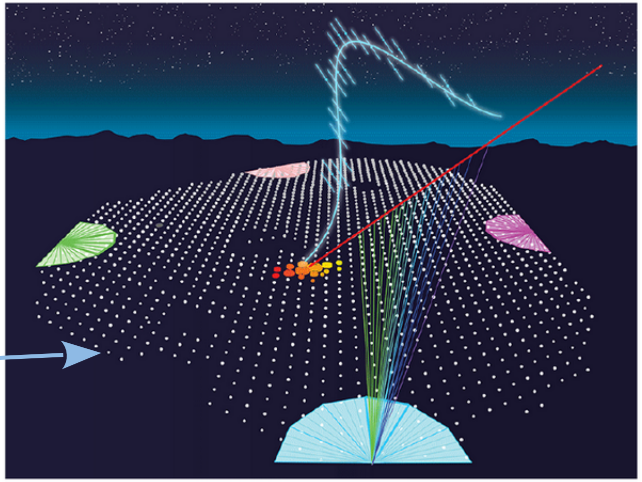
Generate Air Shower Footprints

- Measurement of cosmic ray induced air showers
- **Pierre Auger Observatory: Fluorescence (FD)** and **Surface Detector (SD)**
  - FD: Telescopes measure light of excited nitrogen
  - SD: Water Cherenkov stations detect passage of charged particles
  - Simulation: 2D image sequence, Cartesian grid, 1-100 EeV protons

Open notebook:
http://shorturl.at/hoA38

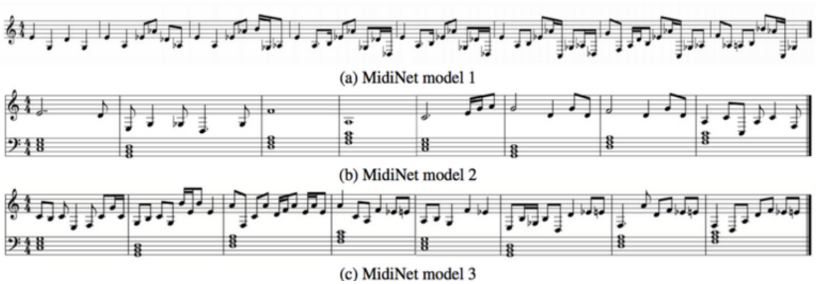






https://physics.aps.org/articles/v9/125

Stay tuned...



Yang, Chou, Yang - https://arxiv.org/abs/1703.10847

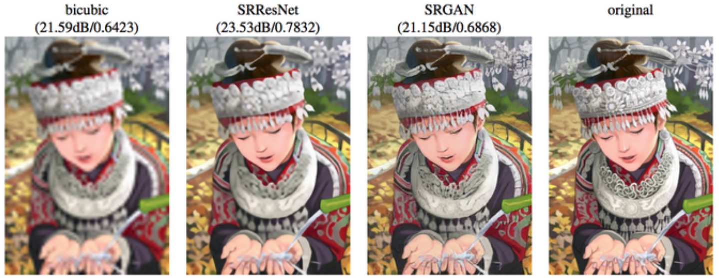
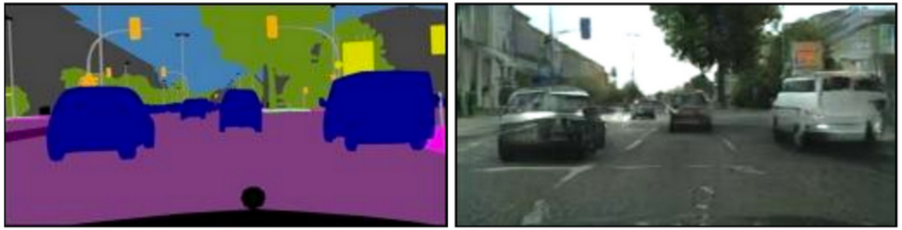


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Ledig et. al. - https://arxiv.org/abs/1609.04802

Zhu, Park, Isola, Efros - https://arxiv.org/abs/1703.10593



input output

Isola, Zhu, Zhou, Efros - https://arxiv.org/abs/1611.07004

… there is much more going on!

The WGAN-GP Algorithm

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

```
while θ has not converged do
    for t = 1, ..., n_critic do
        for i = 1, ..., m do
            Sample real data x ~ P_r, latent variable z ~ p(z), a random number ε ~ U[0, 1].
            x̃ ← G_θ(z)
            x̂ ← εx + (1 − ε)x̃
            L^(i) ← D_w(x̃) − D_w(x) + λ(‖∇_x̂ D_w(x̂)‖_2 − 1)^2
        end for
        w ← Adam(∇_w (1/m) Σ_{i=1}^m L^(i), w, α, β_1, β_2)
    end for
    Sample a batch of latent variables {z^(i)}_{i=1}^m ~ p(z).
    θ ← Adam(∇_θ (1/m) Σ_{i=1}^m −D_w(G_θ(z)), θ, α, β_1, β_2)
end while
```

With:

$$\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$$

$$\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1-\epsilon)\tilde{\boldsymbol{x}}$$

$$L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$$

$$w \leftarrow \text{Adam}\left(\nabla_w \frac{1}{m}\sum_{i=1}^m L^{(i)}, w, \alpha, \beta_1, \beta_2\right)$$

$$\theta \leftarrow \text{Adam}\left(\nabla_\theta \frac{1}{m}\sum_{i=1}^m -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2\right)$$

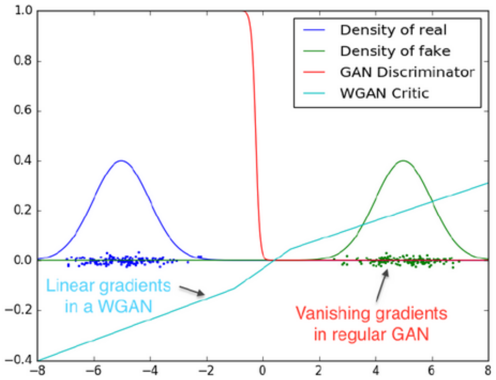
Gulrajani et al.: Improved Training of Wasserstein GANs - https://arxiv.org/abs/1704.00028

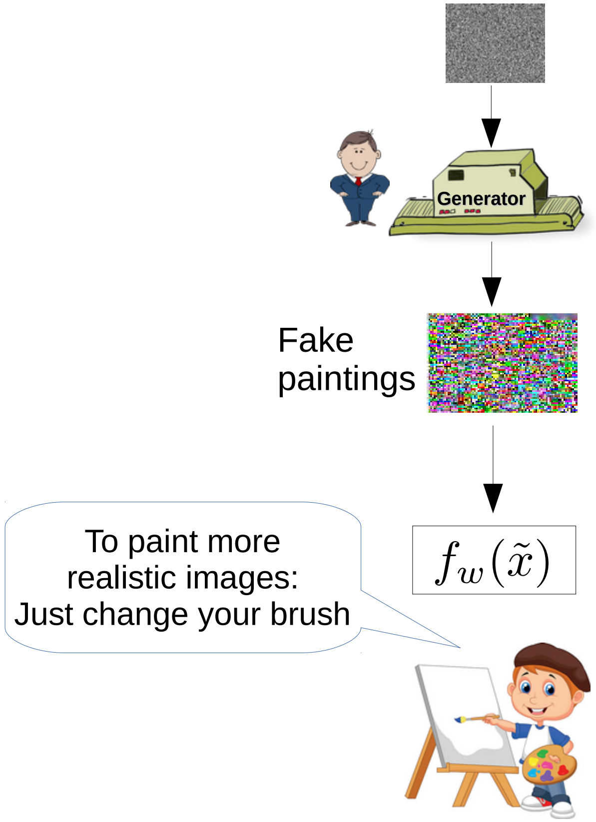
Generator

- Use critic feedback to increase generation quality
  - Minimize $\mathcal{D}_W(P_r, P_\theta)$ using gradient descent

$$\nabla_\theta \mathcal{D}_W(P_r, P_\theta) = -\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}[\nabla_\theta f(G_\theta(\mathbf{z}))]$$

- No vanishing gradients for the generator





Non Saturation GAN (NS-GAN)

- Use **label switching** to avoid vanishing gradients in discriminator
- Standard loss: *minimize*

$$Loss = \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(1 - D(G_\theta(\mathbf{z})))]$$

- But gradients vanish for $D(G_\theta(\mathbf{z})) \to 0$ (good discriminator)

- Replace loss and minimize instead

$$Loss = -\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(D(G_\theta(\mathbf{z})))]$$

  - No vanishing gradient but very instable update
- But new loss has strange update behavior:

$$\mathbb{E}_{z\sim p(z)}\left[-\nabla_\theta \log D^*(g_\theta(z))|_{\theta=\theta_0}\right] = \nabla_\theta \left[KL(\mathbb{P}_{g_\theta}\|\mathbb{P}_r) - 2JSD(\mathbb{P}_{g_\theta}\|\mathbb{P}_r)\right]|_{\theta=\theta_0}$$

Distribution Similarity - Metrics

$\theta = 0$

$$\mathcal{D}_{KL} = 0$$

$$\mathcal{D}_{JS} = 0$$

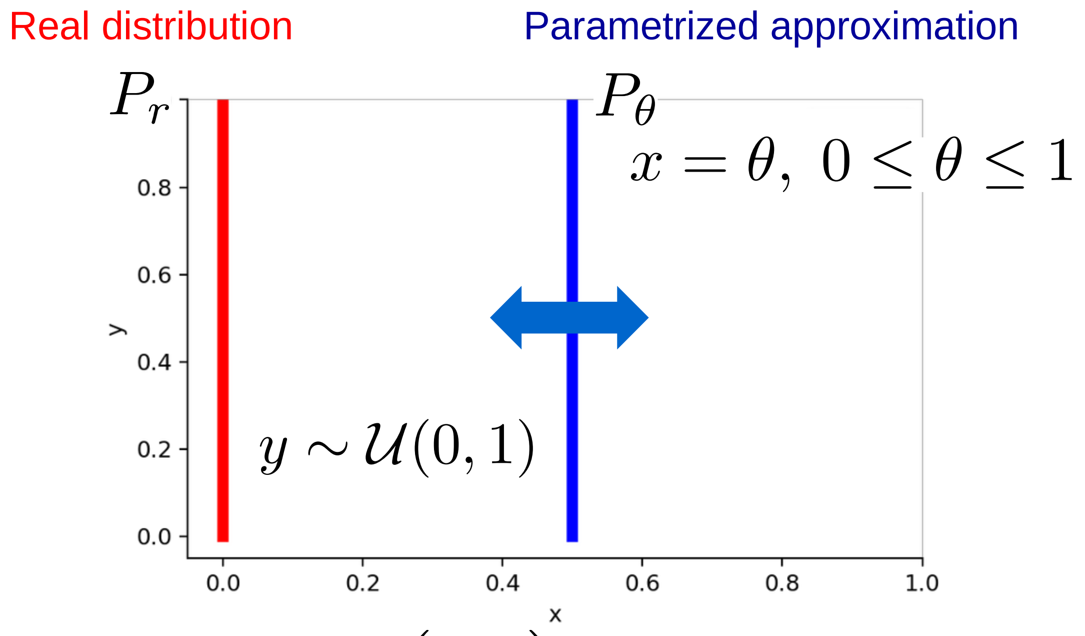
$$\mathcal{D}_{W} = 0$$

$\theta \neq 0$

$$\mathcal{D}_{KL} = \sum_{x=\theta, y\sim\mathcal{U}(0,1)} 1 \cdot \log\left(\frac{1}{0}\right) = \infty$$

$$\mathcal{D}_{JS} = \sum_{x=\theta, y\sim\mathcal{U}(0,1)} 0 \cdot \log\left(\frac{1}{1/2}\right) + \sum_{x=\theta, y\sim\mathcal{U}(0,1)} 1 \cdot \log\left(\frac{1}{1/2}\right) = log(2)$$

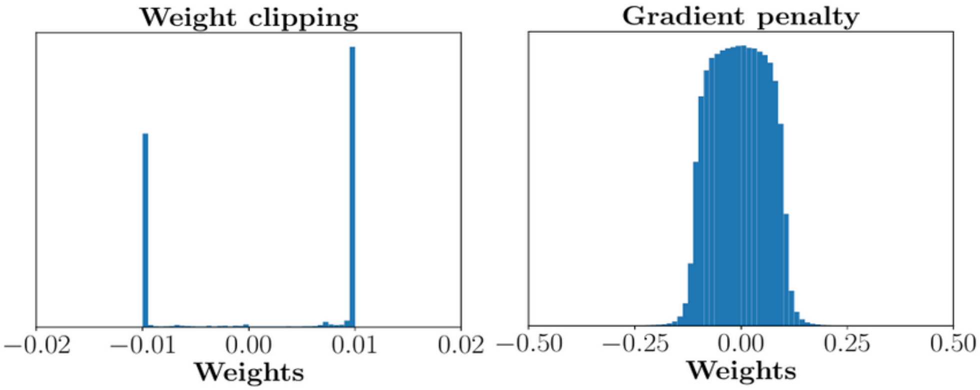
$$\mathcal{D}_{W} = |\theta|$$



➢ Only $\mathcal{D}_W$ provides meaningful distance measure even for disjoint distributions!

Weight Clipping vs. WGAN-GP

- Weight Clipping:
  - Constraints the weights to lie on a compact space
  - Clip weights after each gradient update eg. to [-0,001; 0,001]

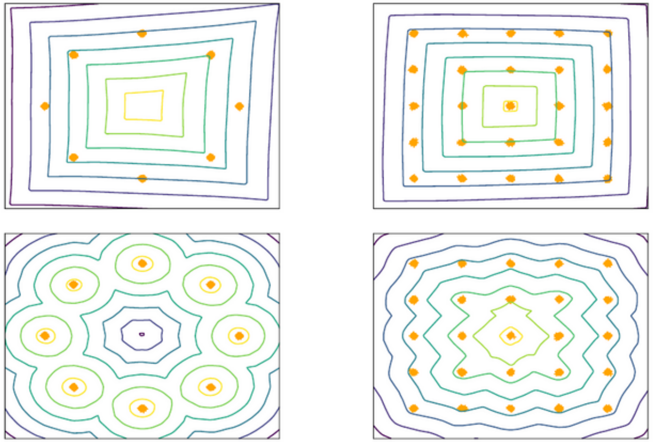


- Heavily constraints the discriminator

- Gradient Penalty allows for a much more complex approximation

Weight clipping

Gradient Penalty

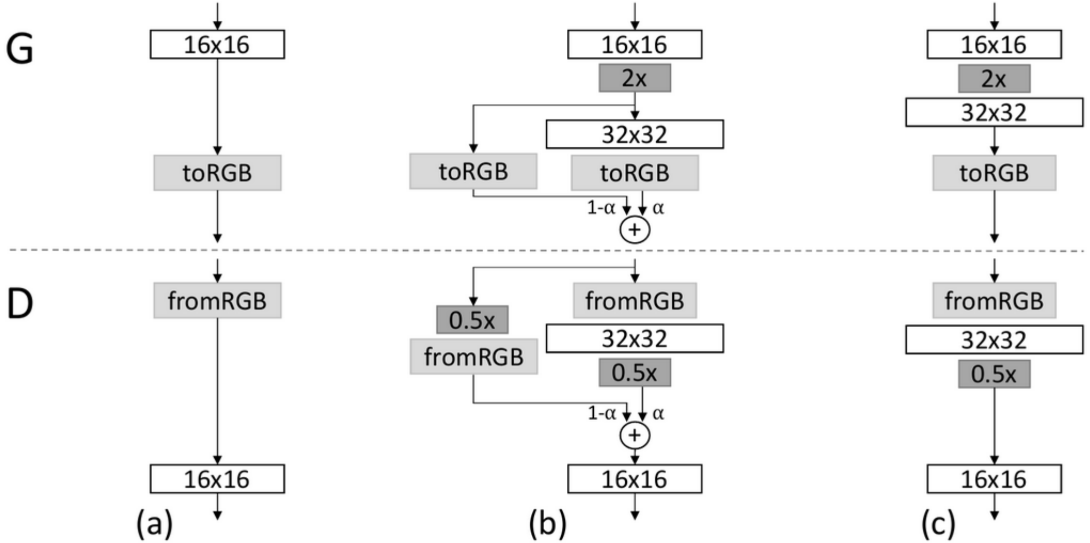


Progressive Growing

- RGB Layer: project feature maps to RGB colors (1 x 1 convolution)
- Upscaling using nearest-neighbor interpolation

- New layer act as "residual block"
  - In generator & discriminator

- During transition:
  - New block is slowly faded in
  - α increases linear from 0 to 1



- Training samples:
  - down-scaled and interpolated during transition between resolutions