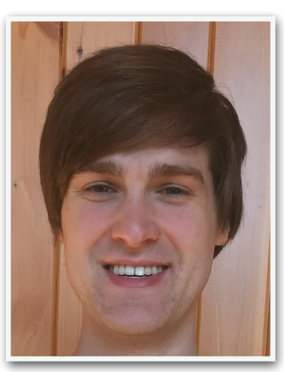
Detecting Overfitting of Deep Generative Networks *via* Latent Recovery

Mathematics of Imaging Workshop, March 2019
"Statistical Modeling for Shapes and Imaging", IHP, Paris

Julien Rabin, Normandie Univ., EnsiCaen, CNRS, GREYC

with



Ryan Webster

Loïc Simon

Frédéric Jurie

Context

- **Deep Learning** is a powerful framework for computer vision and signal processing fields

- **Convolutional Neural Networks** (CNNs) allows for extremely **fast** execution and state-of-the-art **performance** in many applications

- Example: impressive results in image classification VGG **[Simonyan'14]**, Inception Network **[Szegedy'15]**

- In this talk: focus on **Image Generation**

Generative Networks

- **Objective**: synthesize images that are perceptually similar (yet different) from examples from a **dataset**

- **Principle**: train a CNN G to generate such image samples*;* Various (unsupervised) strategies during training **to compare samples with examples from the dataset**

- **Synthesis**: generated images $G(z)$ are obtained by feeding the network with (random) **latent codes** z

- *Remark: no optimization required during synthesis (only ~20 ms with a GPU, <10 s with a CPU)*

GAN

- A recent and popular approach is **Generative Adversarial Networks** **[Goodfellow'14]**, or GANs

- Incredible improvement in the last few years !

2014

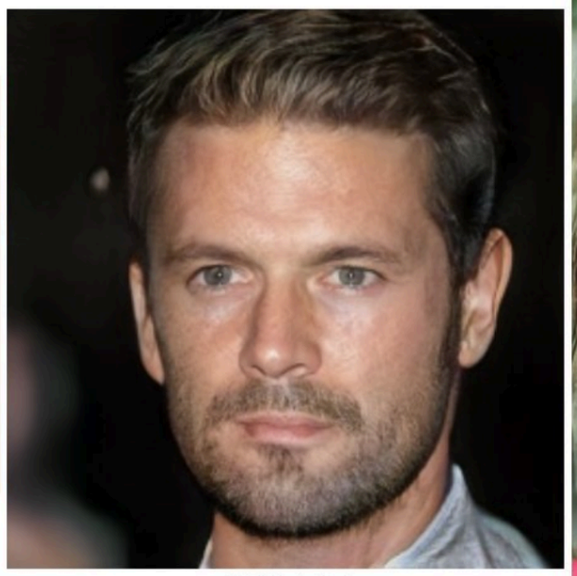
GAN

2015

DC-GAN

2016

Co-GAN



2017

PG-GAN

2018

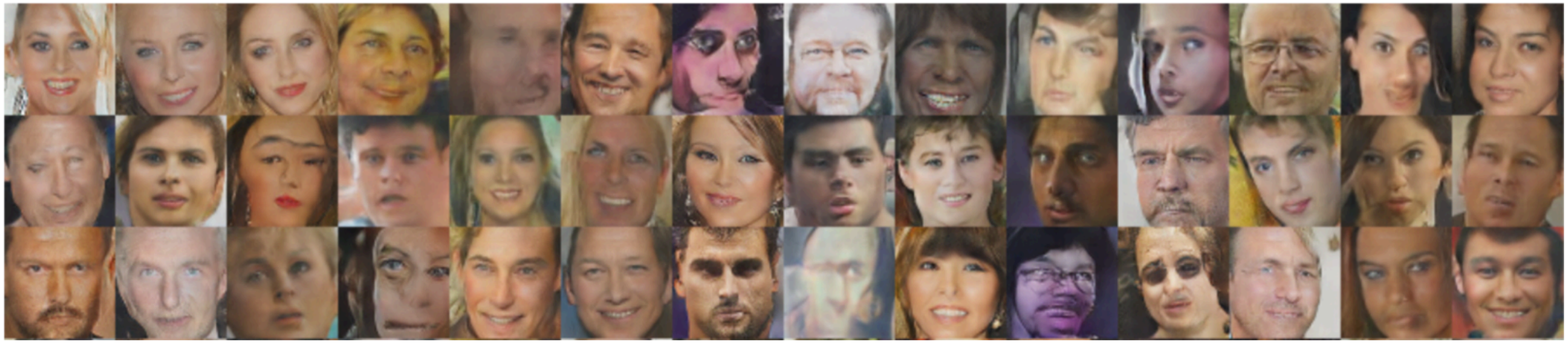
Style GAN

Source: Ian Goodfellow

Examples of image synthesis

- **DC-GAN** **[Radford'16]** (trained on ~350k images for face)

Face
(350k ex.)



Bedrooms
(3M ex.)

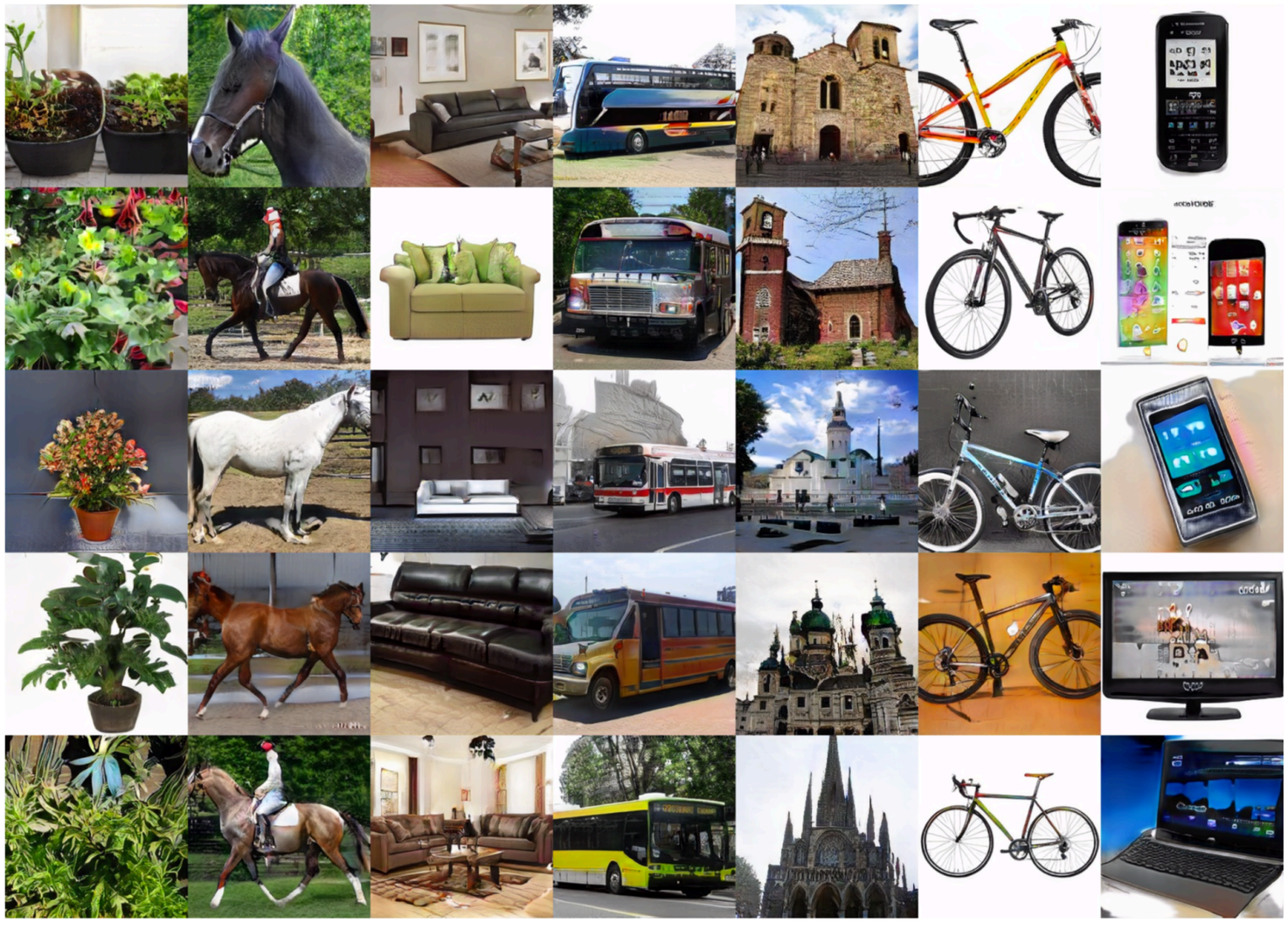
???

Examples of image synthesis

- PG-GAN **[Karras'18]** (trained on ~30k images)

Examples of image synthesis

- PG-GAN **[Karras'18]** (trained on ~100k images/categories)



Examples of image synthesis

- BIG-GAN **[Brock'19]** (trained on ~292M images with 8.5K categories)

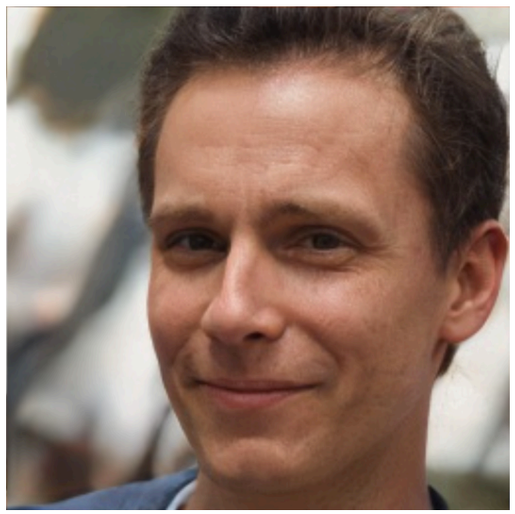



https://ganbreeder.app/

Examples of image synthesis

- Style-GAN **[Karras et al.'19]** (trained on ~80k images)





https://thispersondoesnotexist.com/

Motivation

- **Deep Learning Paradigm:** the **bigger** (= deeper/wider networks), the **better** (*e.g.* BIG-GAN **[Brock'19]**)

- … but requires a lot of (offline) intensive computations and (most of the time) **a large dataset**

- Very strong suspicion of **overfitting** and **memorization** of generator networks, but difficult to investigate

- … although supported by **some evidence** in the literature

Reported limitations

- **Classification Networks** with random labels/images **[Zang'17]**
  « ***The effective capacity of neural networks is sufficient for memorizing the entire data set*** »
  « Explicit regularization may improve generalization performance, but is [not] by itself sufficient »

- Successful **membership inference attack** on **Classification Networks** **[Shokri'17]**

- **Generative Networks** (AutoEncoder) for disk generation **[Newson'18]**
  - **Approximate** the training examples with a **template model**
  - **Failure in generalization** capacity for data interpolation and role of regularization

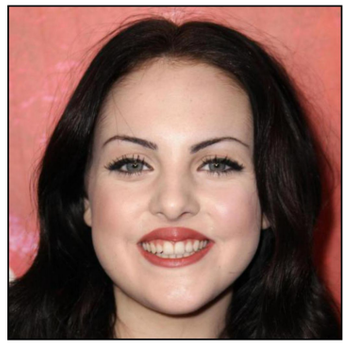
- Discriminator in GANs can **memorize** the training dataset **[Hayes'17]** **[Brock'19]**

Problem

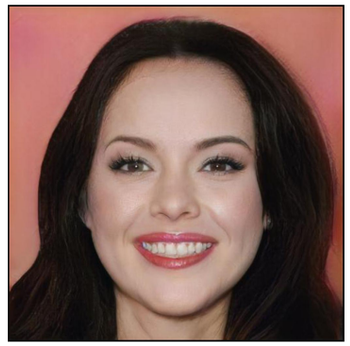
- **Question :** How to evaluate Generative Networks regarding **memorization** ?

training image (LSUN tower)

generated image (WGAN)



training image (CelebA HQ)



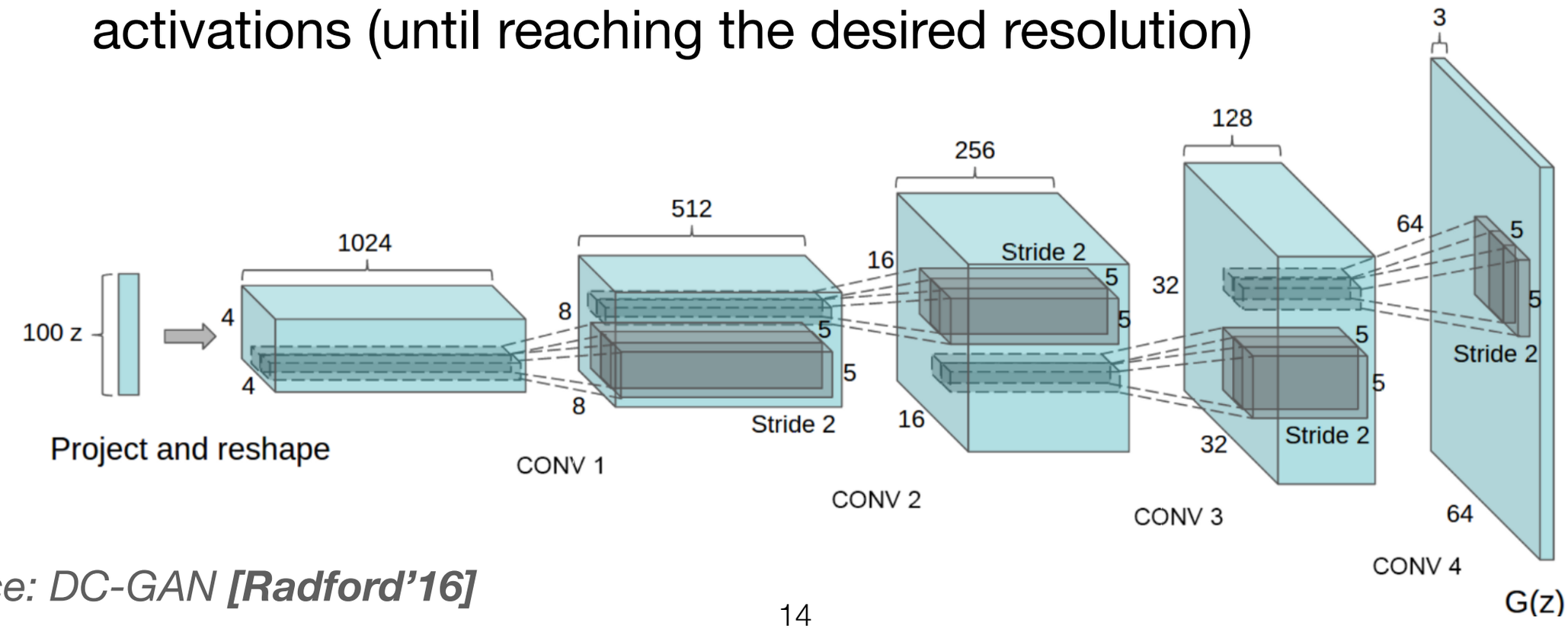
generated image (PGGAN)

- **Related problem:** How to ensure **privacy** of the training data ?

A short review of generative networks

Generative Architecture

- **Typical architecture:** fully convolutional neural network
  - uses a (batch of) *iid* random vectors z as the input
  - starts with a fully connected layer
  - followed by blocks of small upsampling convolutions and ReLU activations (until reaching the desired resolution)



Source: DC-GAN ***[Radford'16]***

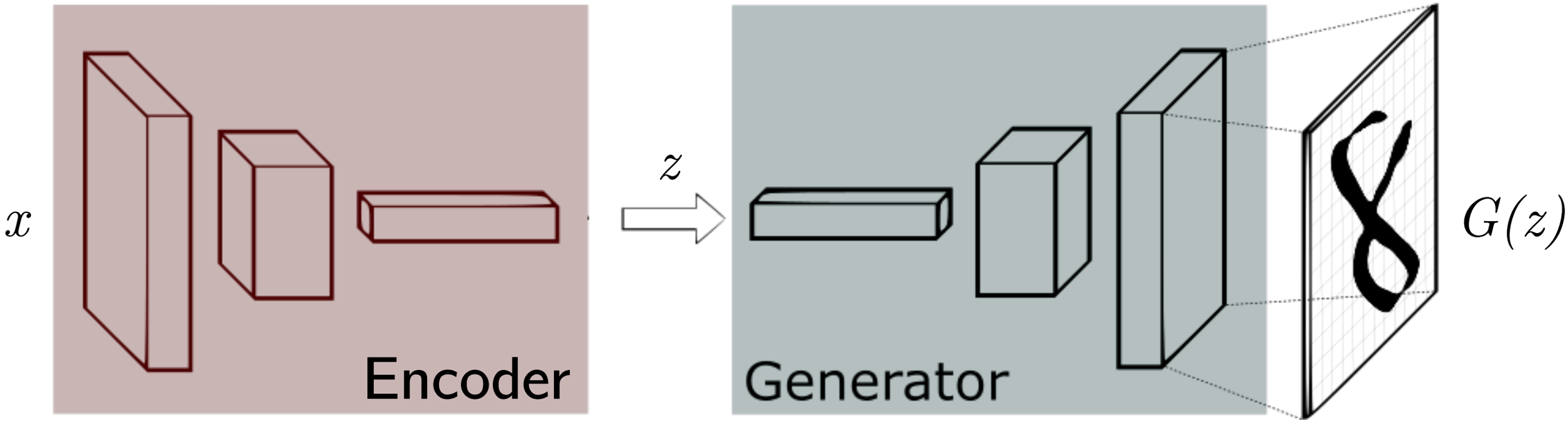
Generative Models

Many different approaches to train a generative network:

- As a decoder: **Auto-encoder** (AE) **[LeCun'87]**, and **Variational Auto-Encoder** (VAE) **[Kingma'14]** (for sampling)
- **Generative Adversarial Network (GAN)** **[Goodfellow'14]**
- **Generative Latent Optimization** network (GLO) **[Bojanowski'18]**
- **Flow-based** generative model (GLOW) **[Kingma and Dhariwal, 18]**
- **Generative Moment Matching networks** (GMMNs)
- **Hybrid methods** (Cycle-GAN **[Chen'18]**, AE-GAN **[Li'17]**, …)

Auto-Encoder

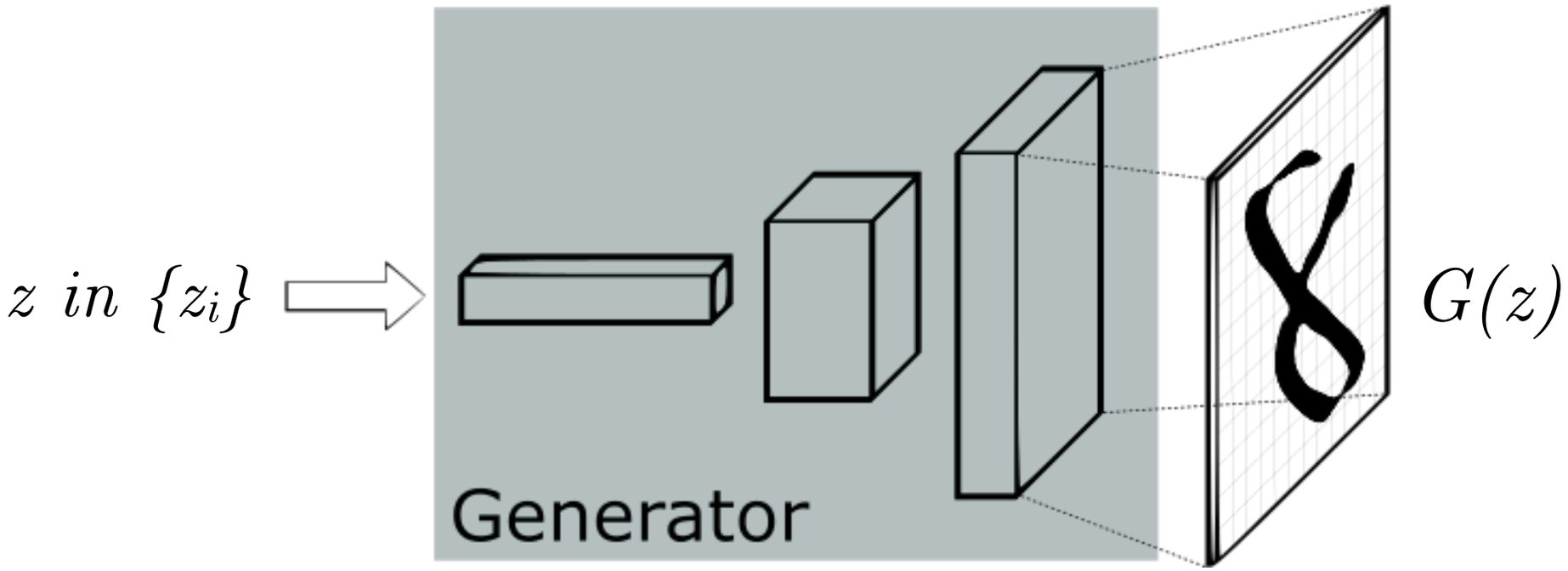
- **Principle:** Encode an input image x into a **latent code** $z=E(x)$ and decode with the generator $G(z)$



- **Optimization problem:** $$\min_{G,E} \sum_{x_i \in \mathcal{D}} \|G(E(x_i)) - x_i\|_2^2$$
- **Variant:** Variational Auto-Encoder (z is a sampled from a pdf parametrized by E)

Generative Latent Optimization

- **Principle:** Both optimize latent codes $\{z_i\}$ and the generator G

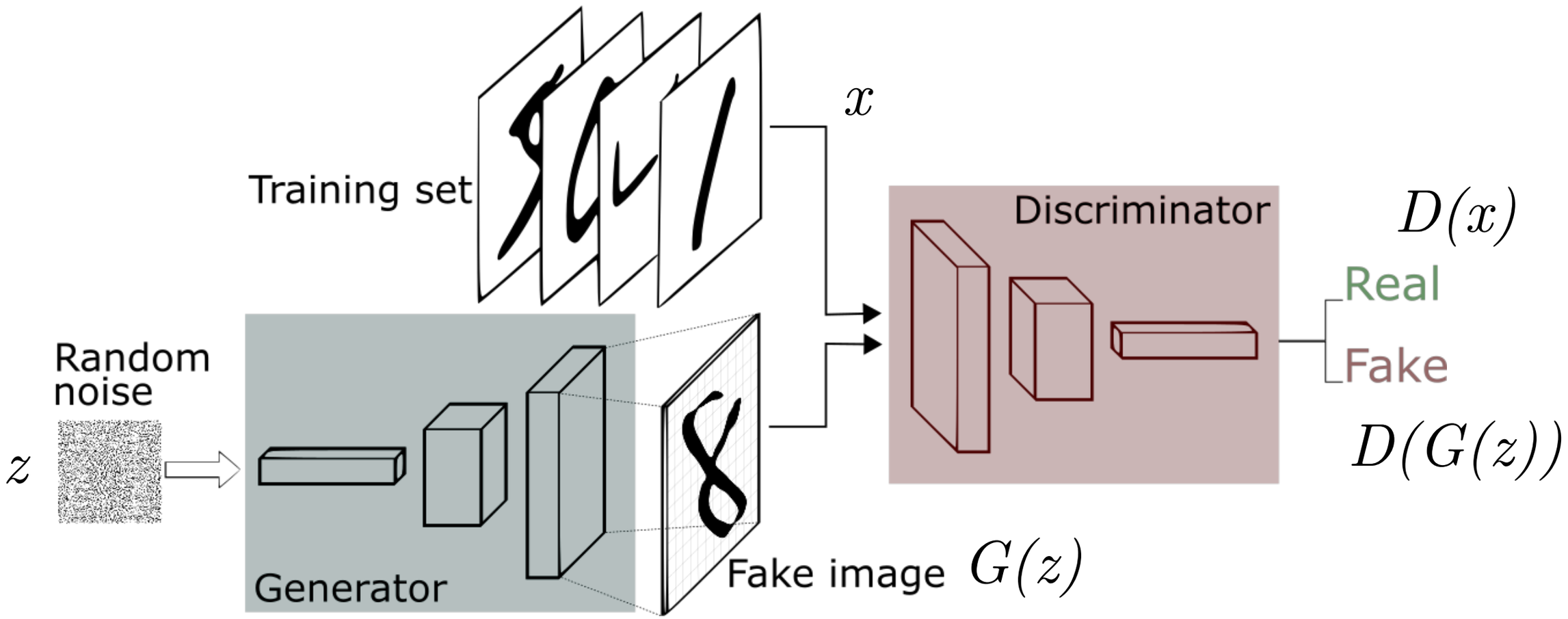


- **Optimization problem:** $$\min_{G,\{z_i\}} \sum_{(z_i,x_i)} \|G(z_i) - x_i\|_2^2$$

- **Synthesis:** sampling from a pdf fitted on latent codes

Generative Adversarial Networks

- **Principle [Goodfellow'14]:** Adversarial optimization strategy between a generator G and a **Discriminator** D detecting real from fake images



- **Optimization:** $\max_D \min_G \mathbb{E}_{z \sim p_Z, x \sim p_{\mathcal{D}}} \log(D(x)) + \log(1 - D(G(z)))$

- **Synthesis:** $G(z)$ computed from a sample z from the training pdf p_Z

Image credit: Thalles Silva

Generative Network Evaluation

Generative Network Evaluation

« *One of the most important research topics in generative modeling is therefore* ***not just how to improve generative models****, but in fact,* ***designing new techniques to measure our progress****.* » ***Deep Learning Book*** **[Goodfellow et al. 2016]**

- In theory, one would like to measure the discrepancy between the **data distribution** p_D and the **generated distribution** p_G

- In practice, a lot of restrictions prevent from doing so (images in high dimensional space ~10^6, empirical approximation of unknown distribution p_D, metric between images, *etc*)

- State of the art approach: **Fréchet Inception Distance** (FID) **[Heusel'17]** is the distance between **Inception Network features** from image from D and G, represented as **Gaussian Density**.

Memorization & Overfitting

- Mostly studied for classification networks:
  - **Generalization** is the capability of a *classification* network trained on an empirical training set to perform as well on unseen data (in practice, a **test set**);
  - **Overfitting** happens when the objective loss function is much **lower for the training set than the test set**

    Memorization is when the loss is *arbitrarily small*

Memorization & Overfitting

- For *generative* networks, it is much less clear in the literature; by analogy, the consensus is that
  - **Memorization** is verbatim copy of (some) training data;
  - **Overfitting** is when synthesized images are « too close » from training examples

- **How to evaluate overfitting and detect memorization in practice** ?

Overfitting Evaluation

- Mostly by **visual inspection** (**[Radford'16],** Birthday paradox **[Arora'17]**)

- **Nearest Neighbor search in the training dataset** of generated images (PGGAN **[Karras'18]**, BigGAN **[Brock'19]** …)

- Comparison of the **estimated distributions** of generated images and training images (*e.g.* with kernel density estimation) for low dimensional data **[Wu'17]**

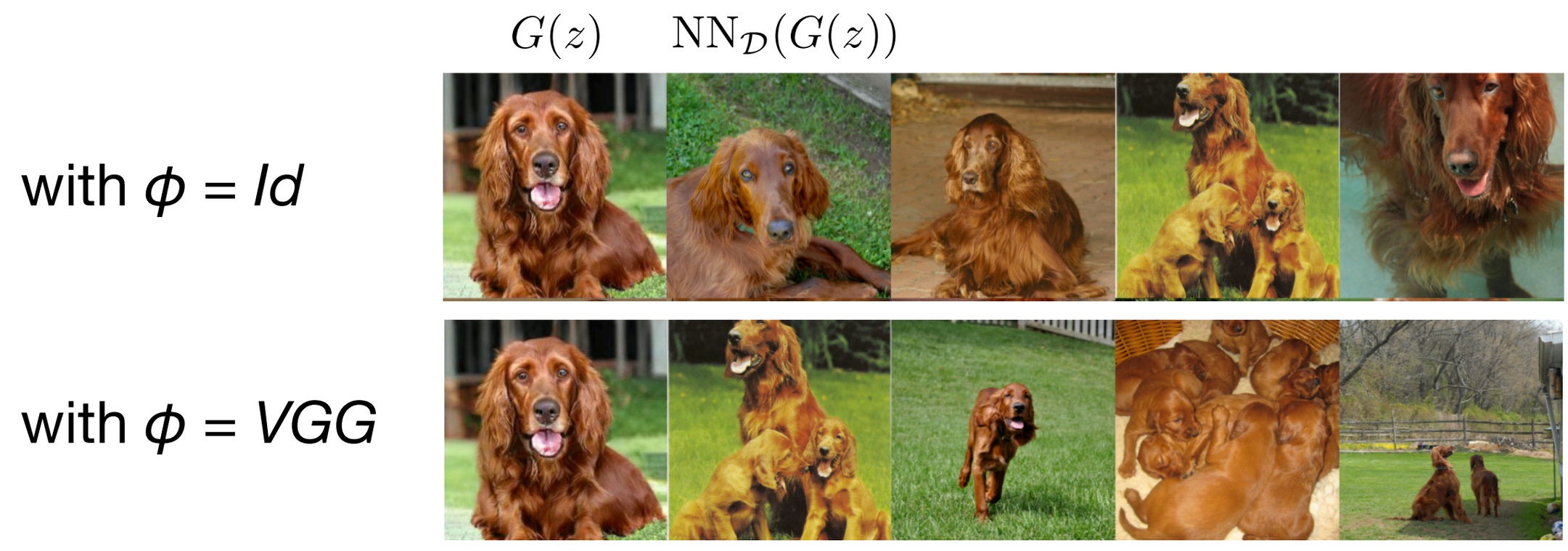
Nearest Neighbor search

- Nearest Neighbor search in dataset

$$\mathrm{NN}_{\mathcal{D}}(G(z)) = \arg\min_{y \in \mathcal{D}} \|\phi(G(z)) - \phi(y)\|_2^2$$

where ϕ extracts features (pixels, patches, CNN …)

- Example from BigGAN **[Brock'19]**



Nearest Neighbor search

- Nearest Neighbor search in dataset

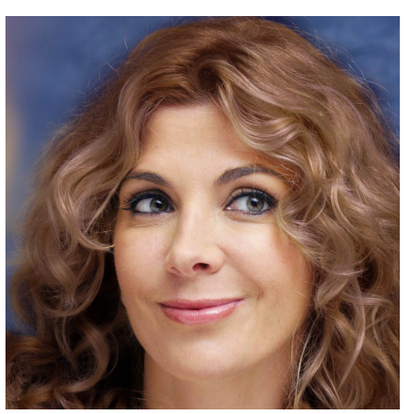
$$\mathrm{NN}_{\mathcal{D}}(G(z)) = \arg\min_{y \in \mathcal{D}} \|\phi(G(z)) - \phi(y)\|_2^2$$

where ϕ extracts features (pixels, patches, CNN …)

- … is **not reliable** !

$y \in \mathcal{D}$

x



$\mathrm{NN}_{\mathcal{D}}(x)$

Nearest Neighbor search

- Nearest Neighbor search in dataset

$$\mathrm{NN}_{\mathcal{D}}(G(z)) = \arg\min_{y \in \mathcal{D}} \|\phi(G(z)) - \phi(y)\|_2^2$$

where ϕ extracts features (pixels, patches, CNN …)

- … is **not reliable** !

$y \in \mathcal{D}$

x

$\mathrm{NN}_{\mathcal{D}}(x)$

with $\phi = Id$

with $\phi = VGG$

Latent recovery

Latent recovery

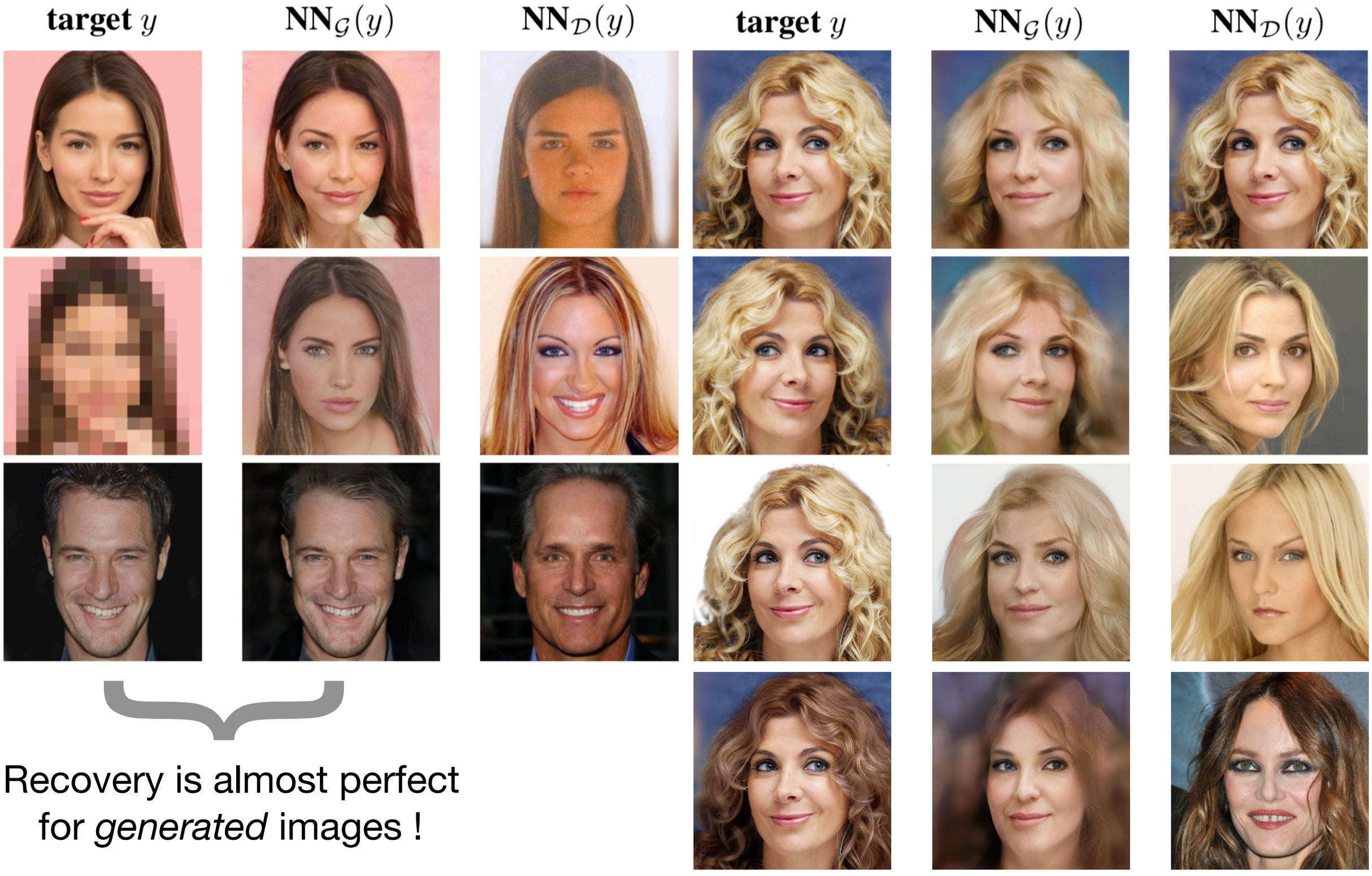
- **Idea:** for images y in the training dataset, search the **nearest neighbor in the manifold of generated images**

$$\mathrm{NN}_{\mathcal{G}}(y) = G(z^\star(y))$$

$$z^\star(y) \in \arg\min_z \|\phi(G(z)) - \phi(y)\|_2^2$$

- … is surprisingly **much more reliable** !
  - for various choice of ϕ : downsampling, masking, color and spatial distorsion, …
  - for other metrics
  - for various optimization algorithms

Latent recovery with PG-GAN

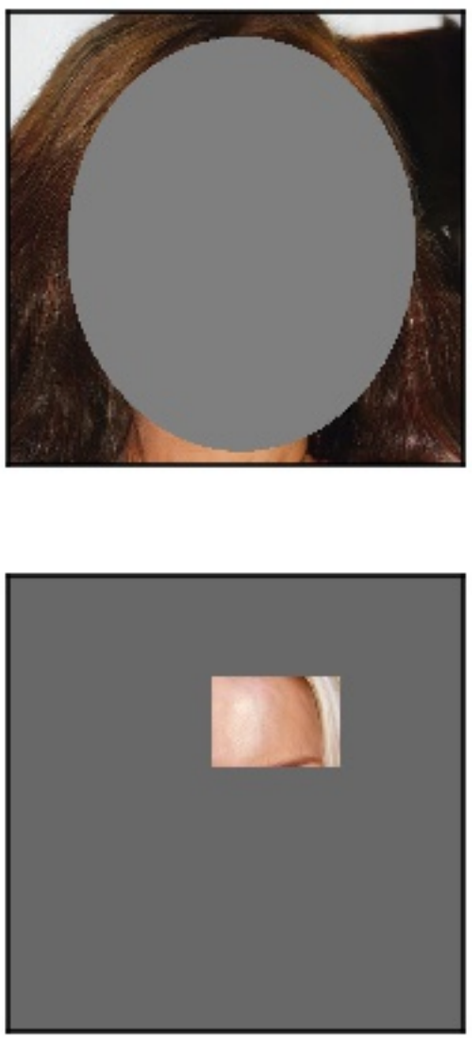


Recovery is almost perfect for *generated* images !

Recovery of generated images

- For generated images, optimization is surprisingly robust !

$G(z_0)$
Ground Truth

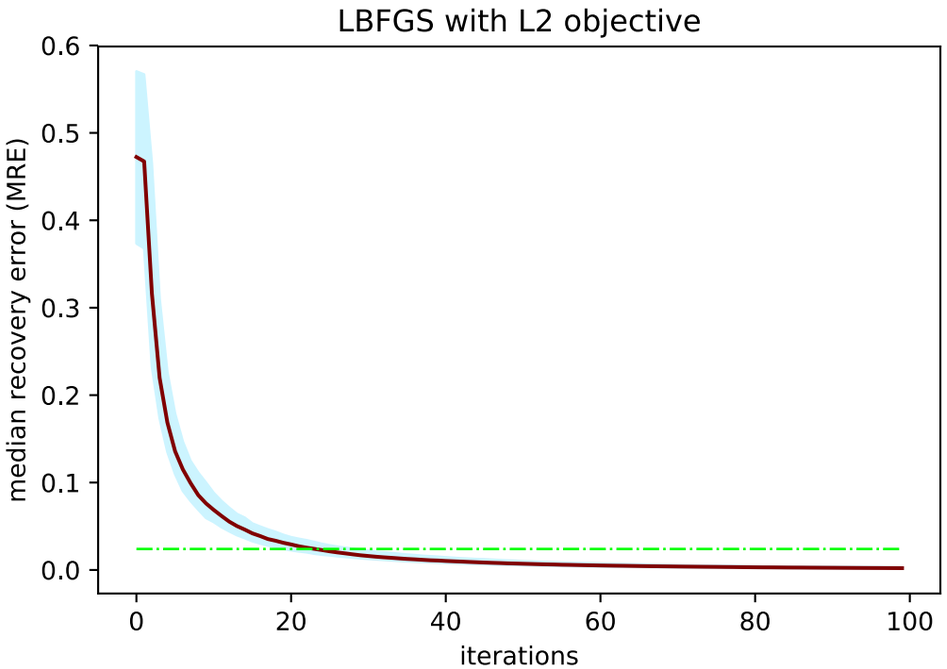


$y=\varphi(G(z))$
Query (after masking)

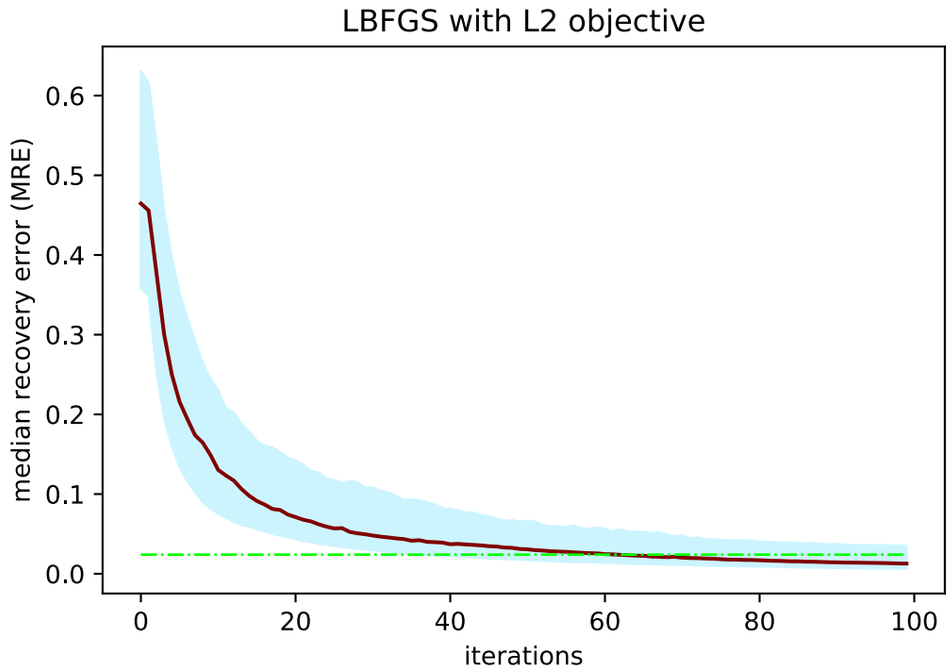
$G(z^*(y))$
Recovered image

Latent Optimization

- Non-convex optimization problem
- Only a few iterations required with L-BFGS algorithm and L2 loss function (even for images outside the training and the generated sets)



DC-GAN (128 pixels)



PG-GAN (1024 pixels)

More details in: arxiv.org/abs/1901.03396

Assessing overfitting

Latent recovery for overfitting

- **Proposed definition for overfitting:** discrepancy measure of latent recovery error **between test and train datasets**

- **Protocol:** (for instance on CelebA-HQ with 28k images)

  1) Dataset is divided into a test set T (2k) and train set D (26k)

  2) Generative model is trained on D

  3) Compute latent recovery errors for test and train set (2 x 2k)

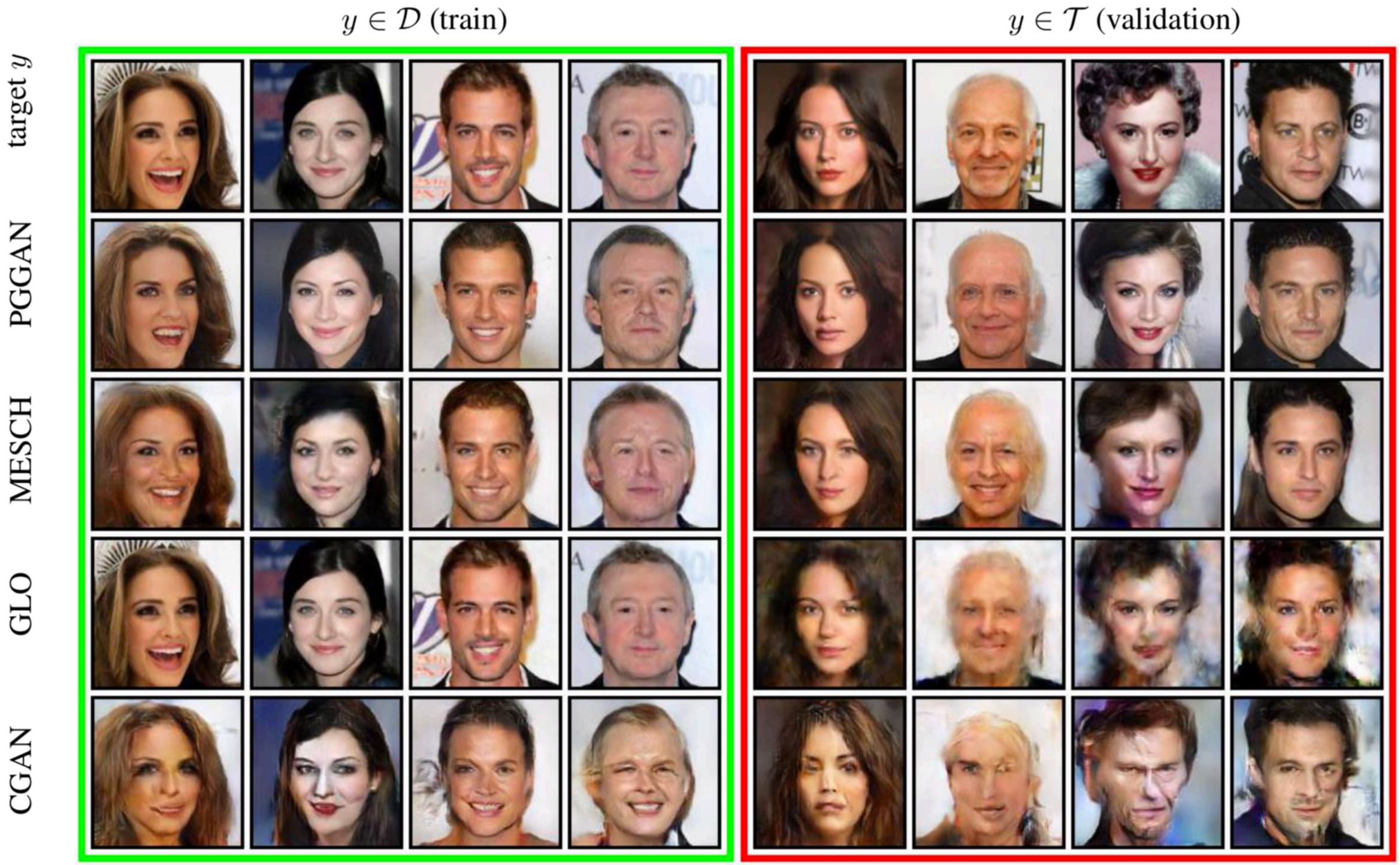
$$\forall y \in \mathcal{D} \cup \mathcal{T},\ \|G(z^\star(y)) - y\|_2^2 \ \text{ where } \ z^\star(y) \in \arg\min_z \|\phi(G(z)) - \phi(y)\|_2^2$$

Experimental Setup

- **Tested Generative models:**
  - **AE** and **GLO**: used for baseline, models trained to memorize
  - **GANs:** DC-GAN, PG-GAN and MESCH **[Mescheder'17]** (based on ResNet **[He'18]** and WGAN-GP **[Gulrajani'18]**)
  - **Hybrid GANs:** Auto-Encoder GAN **[Choi'18] [Li'17],** Cycle-GAN **[Zhu'18]**
- **Datasets:** CelebA-HQ, LSUN (tower and bedrooms), MNIST, CIFAR, Yosemite
- **Various datasize** (for all but pure GANs, that fail with small datasets)

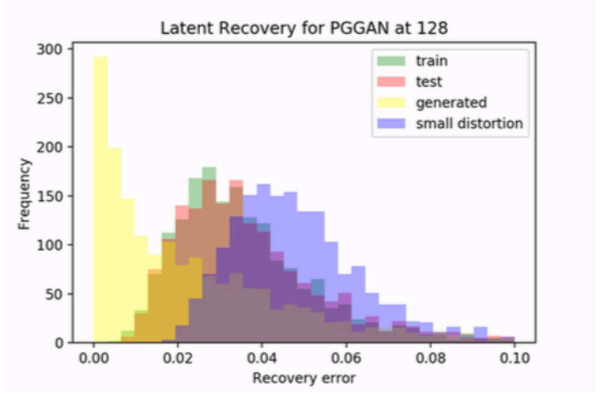
Latent recovery on CelebA-HQ

- Various generative models on CelebA-HQ

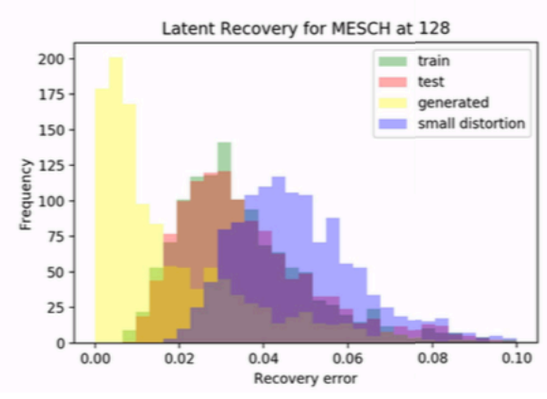


Latent recovery on CelebA-HQ

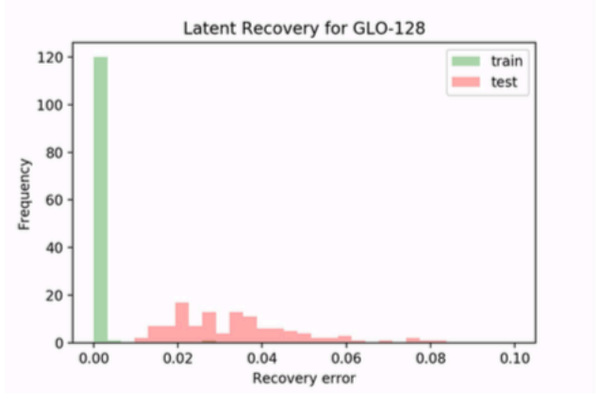
- Histograms of quadratic recovery error (MSE)
- Legend: **Train**, **Test**, **Generated**, and **Distorded**
- **GANs are not memorizing**
- For other models, overfitting depends on the data size (here 128, 1024 and 8192)



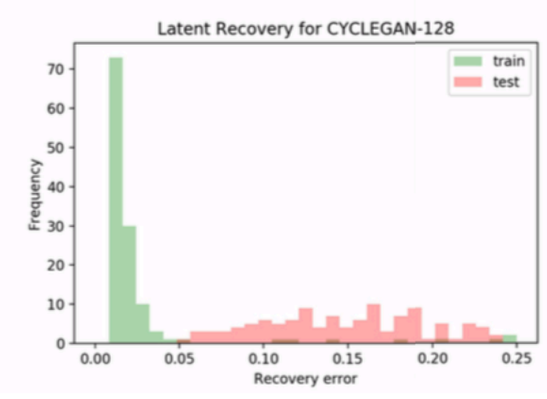
(a) PGGAN



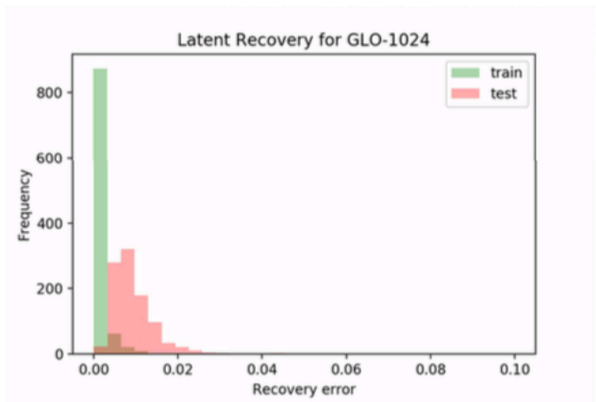
(b) MESH



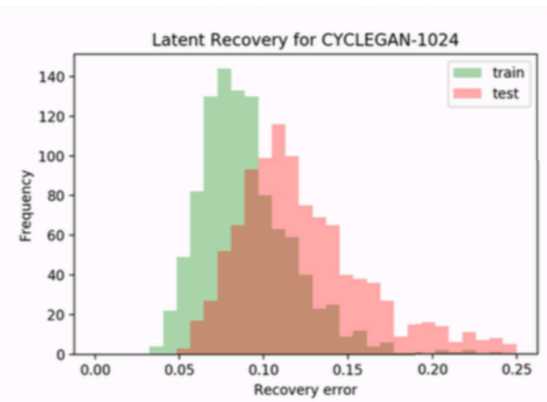
(c) GLO-128



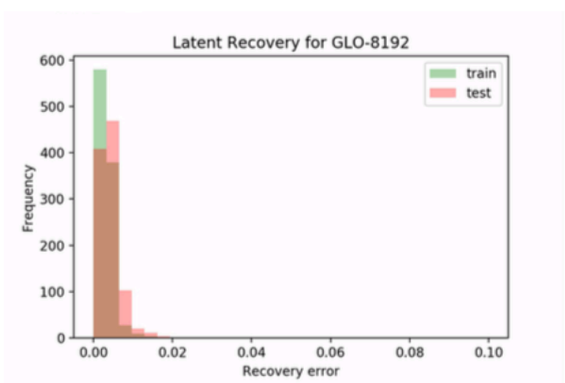
(d) CGAN-128



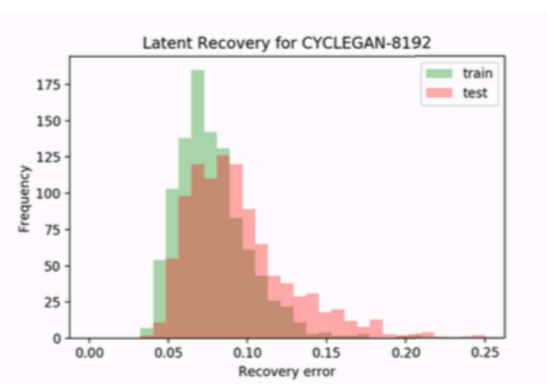
(e) GLO-1024



(f) CGAN-1024



(g) GLO-8192



(h) CGAN-8192

Statistical analysis

Measure of overfitting as the difference of test and train errors:

- **Median Recovery Error** $\mathrm{MRE}_G(\mathcal{Y}) = \mathrm{median}\left\{\min_z \|y_i - G(z)\|^2\right\}_{y_i \in \mathcal{Y}}$

  Normalization $\text{MRE-gap}_G = \left(\mathrm{MRE}_G(\mathcal{T}) - \mathrm{MRE}_G(\mathcal{D})\right) / \mathrm{MRE}_G(\mathcal{T})$

- **Hypothesis testing:** *p-value* of the Kolmogorov-Smirnov (KS) test (probability of larger difference)

Detection of overfitting using either

- above **10% threshold** on MRE Gap
- below **1% threshold** on p-value of KS test

Statistical results

		KS p-value	MRE-gap	MRE			
		train vs val		train	val	generated	small distort
CelebA-HQ	DCGAN	9.43e-01	1.79e-02	4.95e-02	5.04e-02	3.68e-03	5.69e-02
	MESCH	4.55e-01	6.96e-03	3.40e-02	3.43e-02	1.77e-02	4.63e-02
	PGGAN	2.22e-01	2.22e-02	3.31e-02	3.39e-02	1.78e-02	4.65e-02
	GLO-128	**0.00e+00**	9.70e-01	9.94e-04	3.30e-02	5.10e-05	9.32e-03
	GLO-1024	**0.00e+00**	7.59e-01	1.95e-03	8.08e-03	1.29e-03	4.46e-03
	GLO-8192	**2.25e-18**	1.75e-01	3.00e-03	3.64e-03	1.04e-03	3.20e-03
	GLO-26000	2.12e-01	3.69e-02	4.27e-03	4.44e-03	4.08e-04	4.43e-03
	AEGAN-128	**0.00e+00**	9.02e-01	1.54e-02	1.57e-01	N/A	2.82e-02
	AEGAN-1024	**0.00e+00**	2.68e-01	8.52e-02	1.16e-01	N/A	8.69e-02
	AEGAN-8192	**3.17e-27**	1.61e-01	7.42e-02	8.84e-02	N/A	7.55e-02
	AEGAN-26000	1.25e-01	1.85e-02	9.96e-02	1.01e-01	N/A	1.00e-01
	CYCLEGAN-256 M2F	**0.00e+00**	4.75e-01	9.03e-03	1.72e-02	N/A	-
	CYCLEGAN-4096 M2F	**0.00e+00**	2.62e-01	6.44e-03	8.73e-03	N/A	-
LSUN	DCGAN (tower)	7.02e-02	1.36e-02	7.96e-02	8.07e-02	1.49e-02	7.31e-02
	DCGAN (bedroom)	3.65e-01	5.34e-03	7.06e-02	7.10e-02	7.03e-02	7.09e-02
	GLO-8192 (bedroom)	**6.70e-06**	1.70e-01	5.45e-03	6.56e-03	5.37e-04	5.01e-03
	GLO-32768 (bedroom)	2.62e-01	5.40e-02	6.58e-03	6.25e-03	8.40e-04	5.44e-03
Yosemite	CYCLEGAN-256 s2w	**1.60e-16**	3.68e-01	1.67e-02	2.64e-02	N/A	-
	CYCELGAN-512 s2w	**6.10e-33**	3.78e-01	1.39e-02	2.23e-02	N/A	-
MNIST	DCGAN	2.41e-01	8.85e-02	3.00e-02	2.75e-02	6.89e-03	-
	GLO-1024	**0.00e+00**	6.78e-01	2.86e-04	8.88e-04	1.49e-03	-
	GLO-16384	3.48e-01	6.45e-03	8.72e-04	8.77e-04	1.41e-03	-
	AEGAN-16384	**7.43e-02**	2.29e-02	4.56e-02	4.67e-02	N/A	-
CIFAR10	DCGAN	5.40e-01	3.65e-03	2.29e-01	2.28e-01	1.30e-03	-
	GLO-1024	**0.00e+00**	5.84e-01	2.77e-03	6.67e-03	8.53e-04	-
	GLO-16384	3.48e-01	6.45e-03	8.72e-04	8.77e-04	1.41e-03	-

Local overfitting of GAN

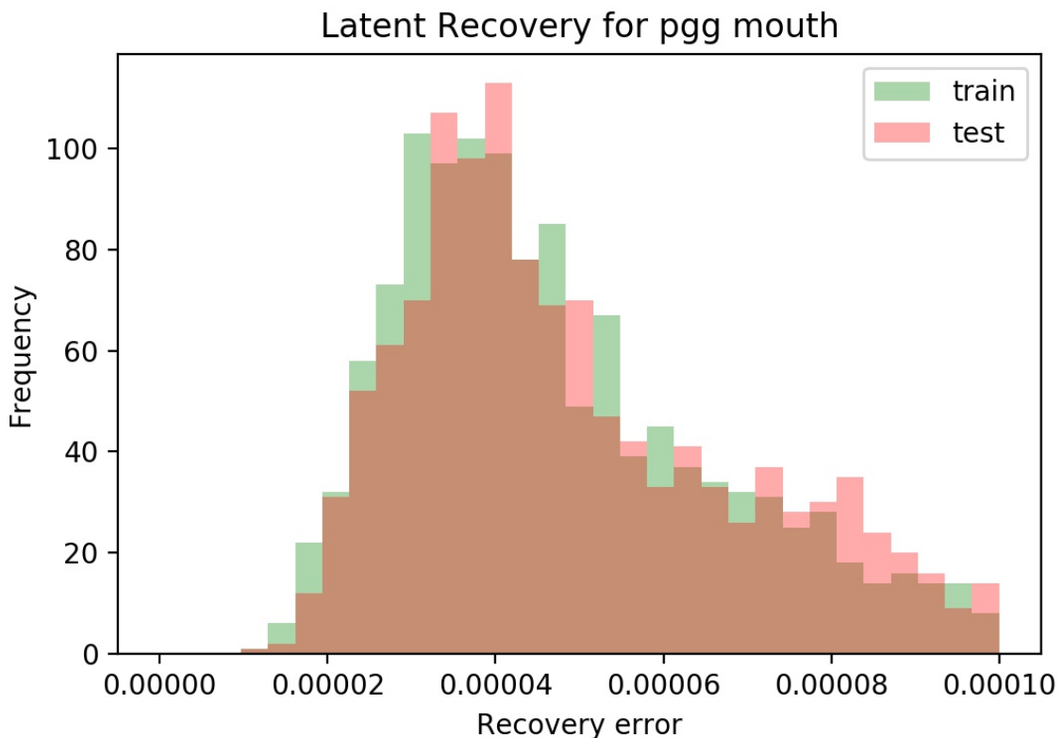
- One could wonder if overfitting occurs *only in small parts* of the image
- Tests on mouth and eye regions: still **no overfitting detected** for PG-GAN !

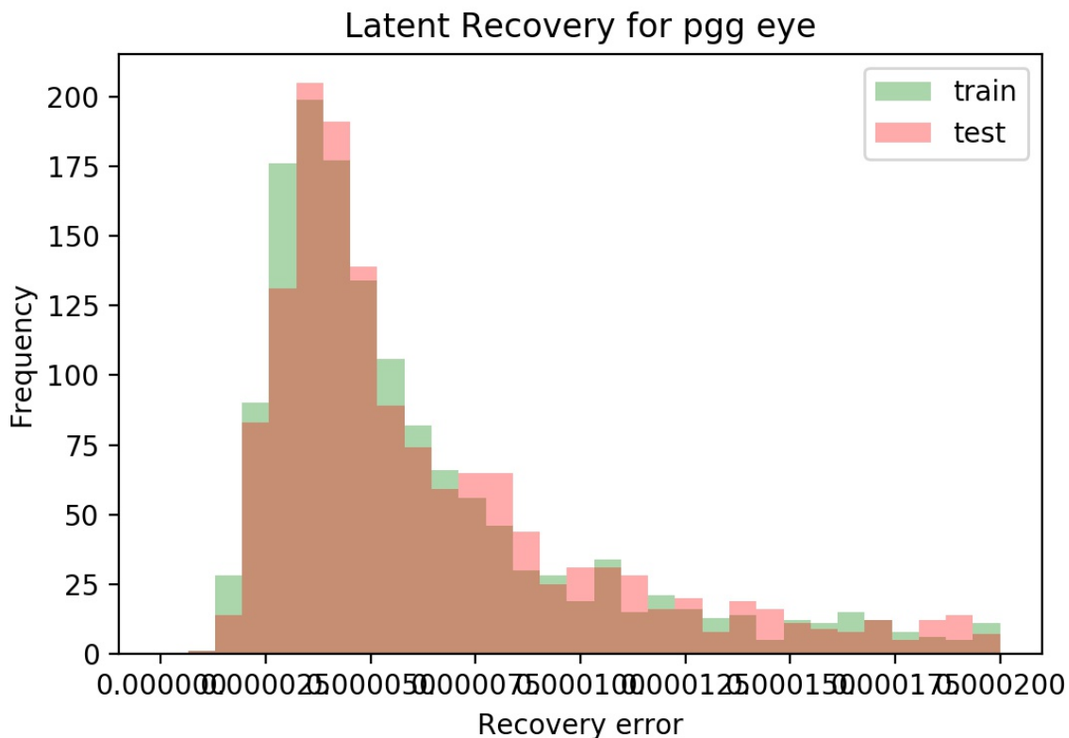
target patch

recoveries (3 rand. init.)

target patch

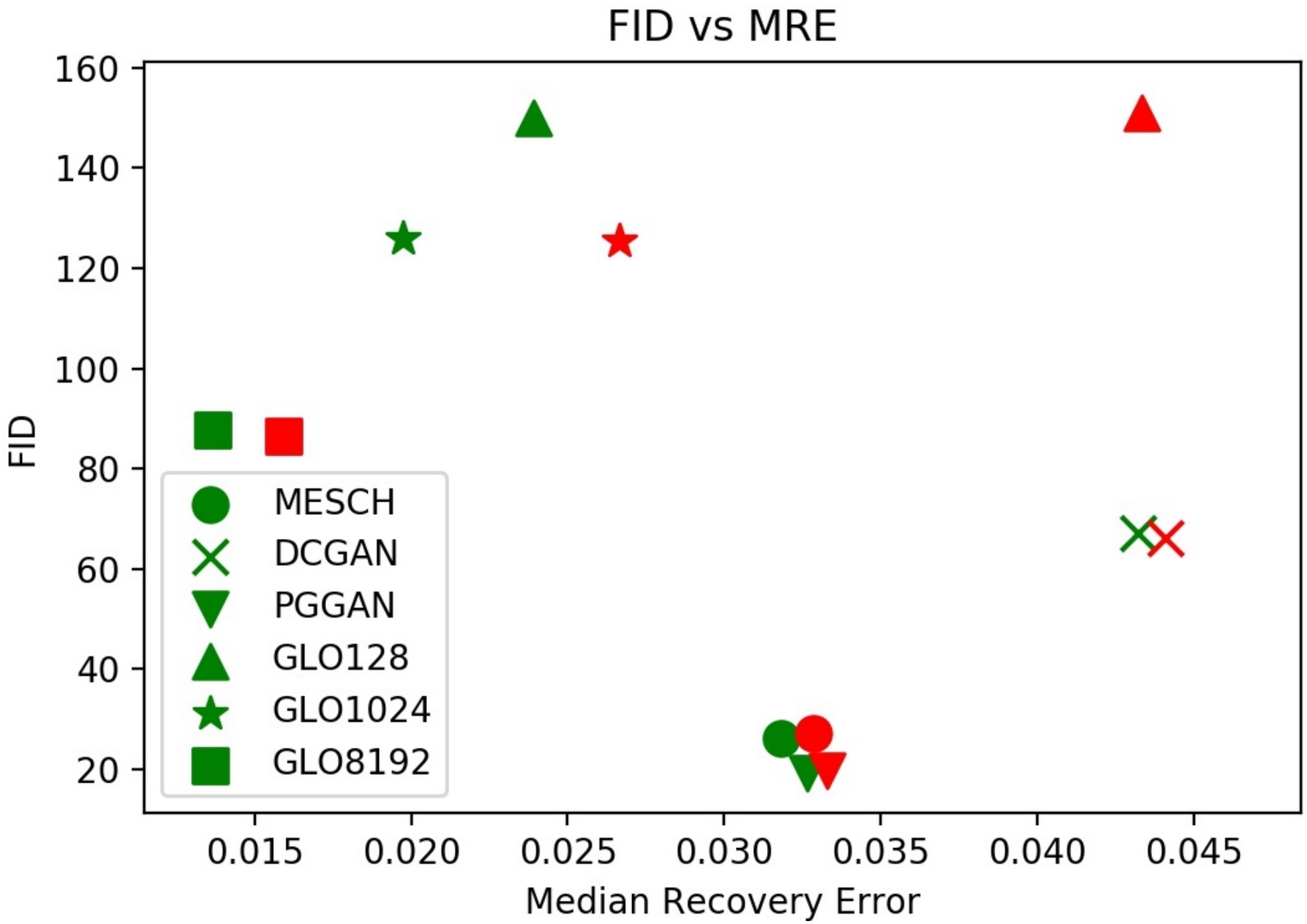
recoveries (3 rand. init.)





Comparison with FID

- FID doesn't detect overfitting (for **Train** and **Test** images)



Applications

> skip ?

Application of latent recovery

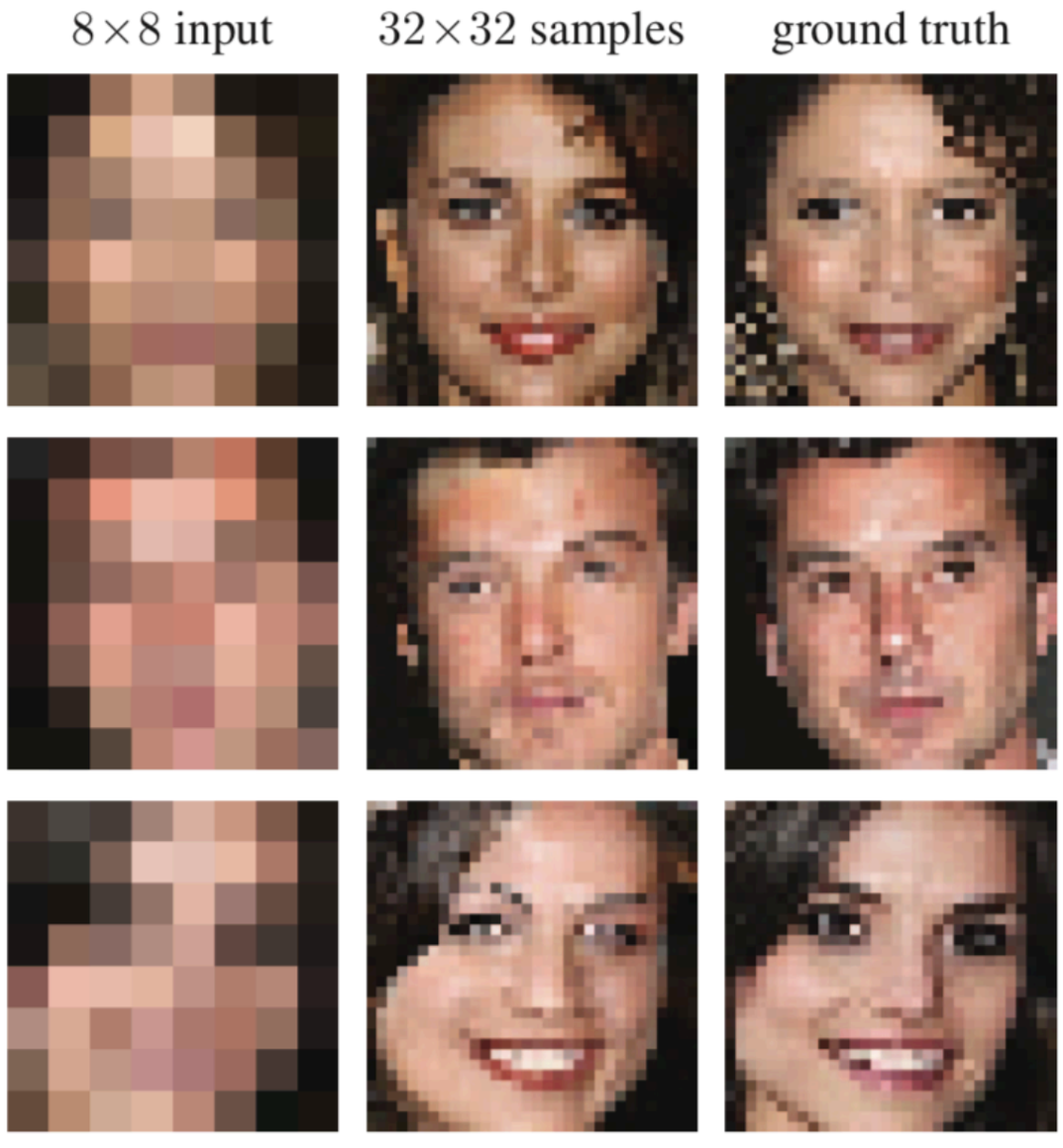
- Most approaches in the literature use **hybrid approches** mixing *Adversarial* and *Auto-Encoder* training strategies that have been shown to be **prone to overfitting**

- **Idea:** combine pure GAN method and latent recovery to prevent overfitting and identity preserving **without requiring training**

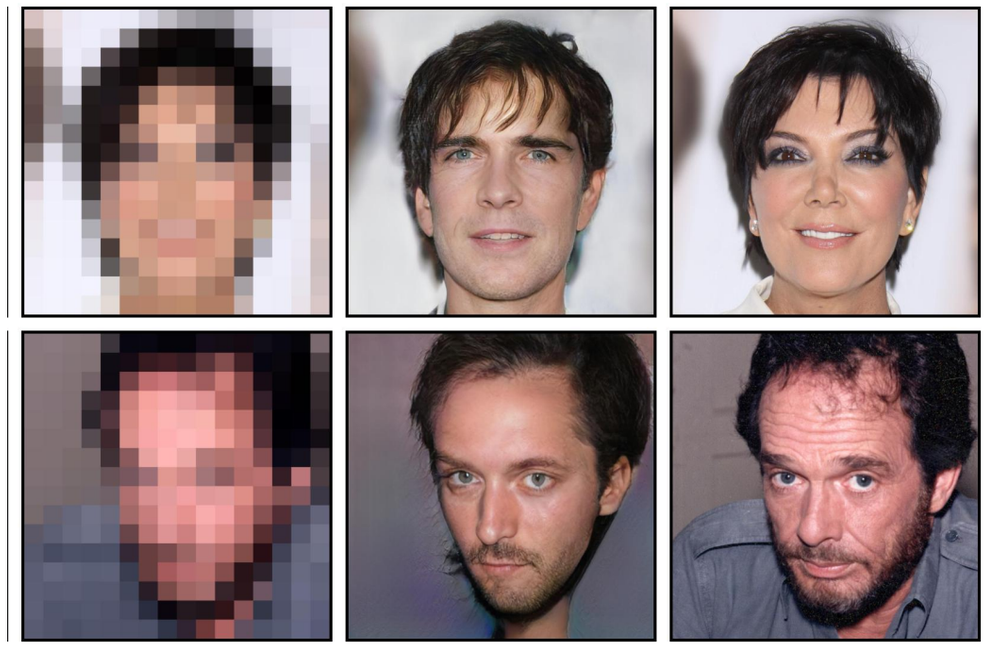
- We already demonstrate its practical interest for **face de-identification**

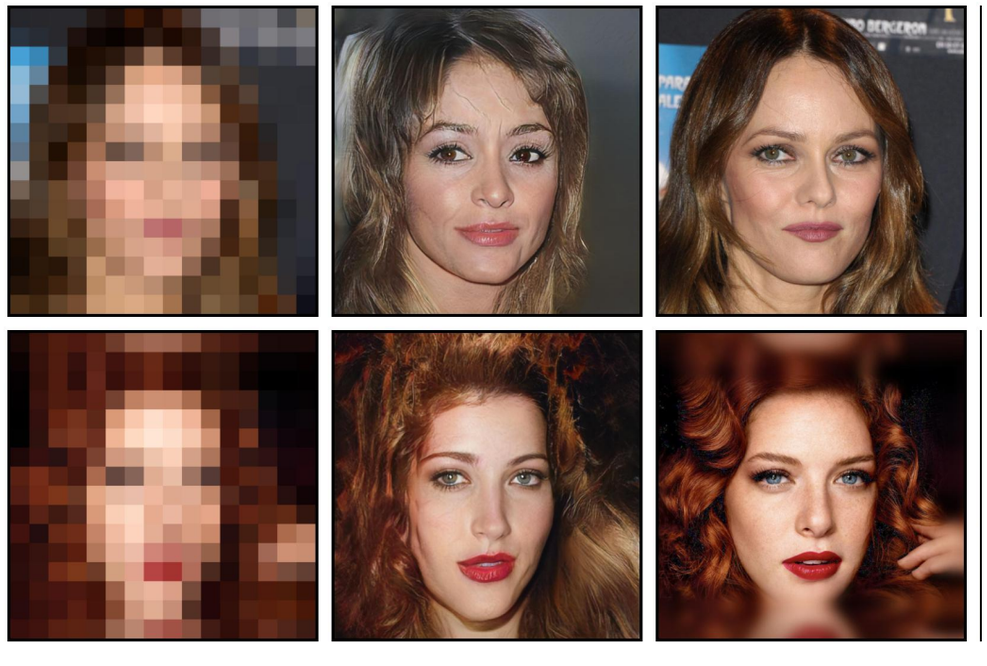
Super-Resolution

- Generative Network **[Dahl'17]**



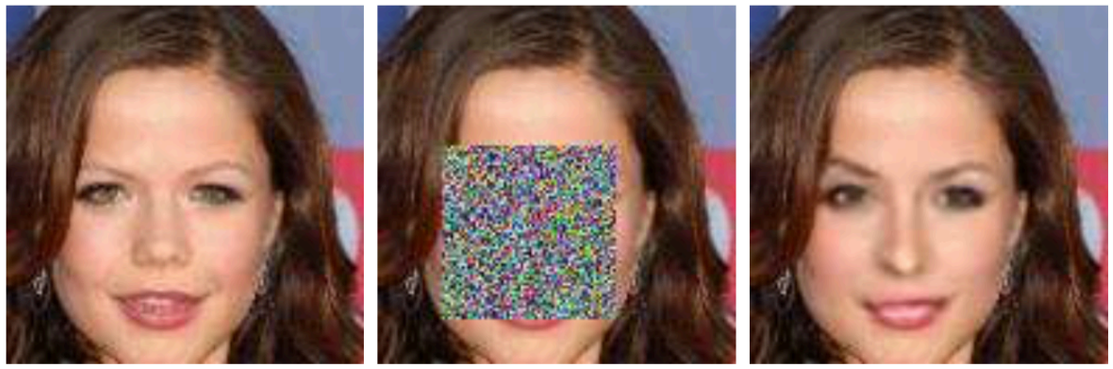
- Using PG-GAN and pooling for latent recovery





Face Completion

- Hybrid AE-GAN **[Li'17]**



- Hybrid PG-GAN **[Chen'18]**

- Latent recovery (without post-processing for blending)

Conclusion

Conclusion

- A simple **definition** for overfitting & memorization for generative networks
- Overfitting is evaluated using **latent recovery**
- Proof that overfitting can happen in Generative Networks, especially for popular hybrid GAN approaches
- FID does not detect overfitting
- Interesting applications to face processing (de-identification, completion, super-resolution) preserving the privacy of the training set
- Future work: faster latent recovery, improving optimization for non registered dataset

Thank you !

Miscellaneous

- Preprint: arxiv.org/abs/1901.03396

- Code: GitHub.com/ryanwebster90