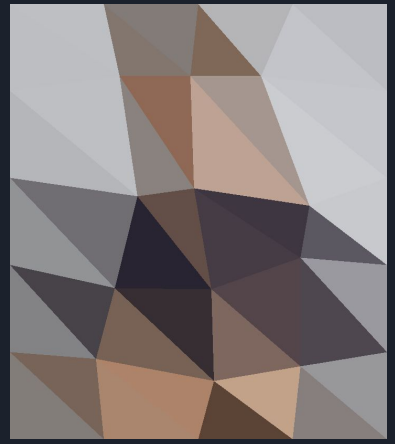
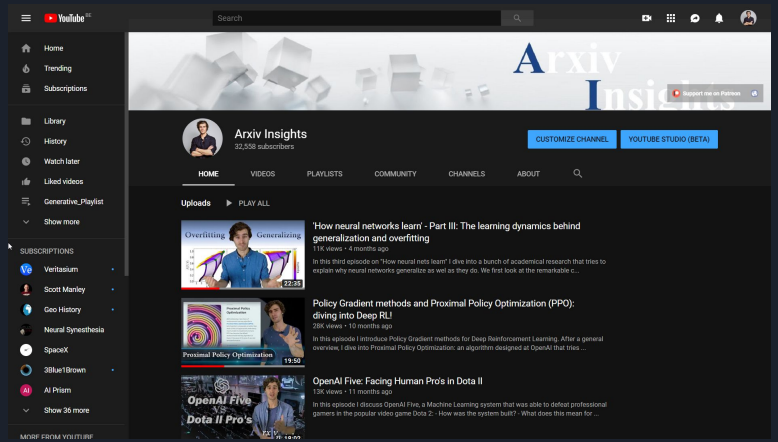
An overview of Generative Adversarial Networks

by Xander Steenbrugge

A quick word about myself..

- Discovered ML in 2014 through my Masters thesis on BCI
- Been an expert Deep Learning consultant for 5 years
- Currently head of applied ML Research @ML6, Belgium
- Run a YouTube channel on AI “Arxiv Insights”





ML6

Overview

- Quick introduction to GANs
- GAN optimization objective

  → 5min Technical Deepdive
- State-of-the-art GAN techniques
- Playing with the latent space of StyleGAN

  → You will be able to do this yourself!!
- Outlook on the generative landscape

deeplearning.ai presents

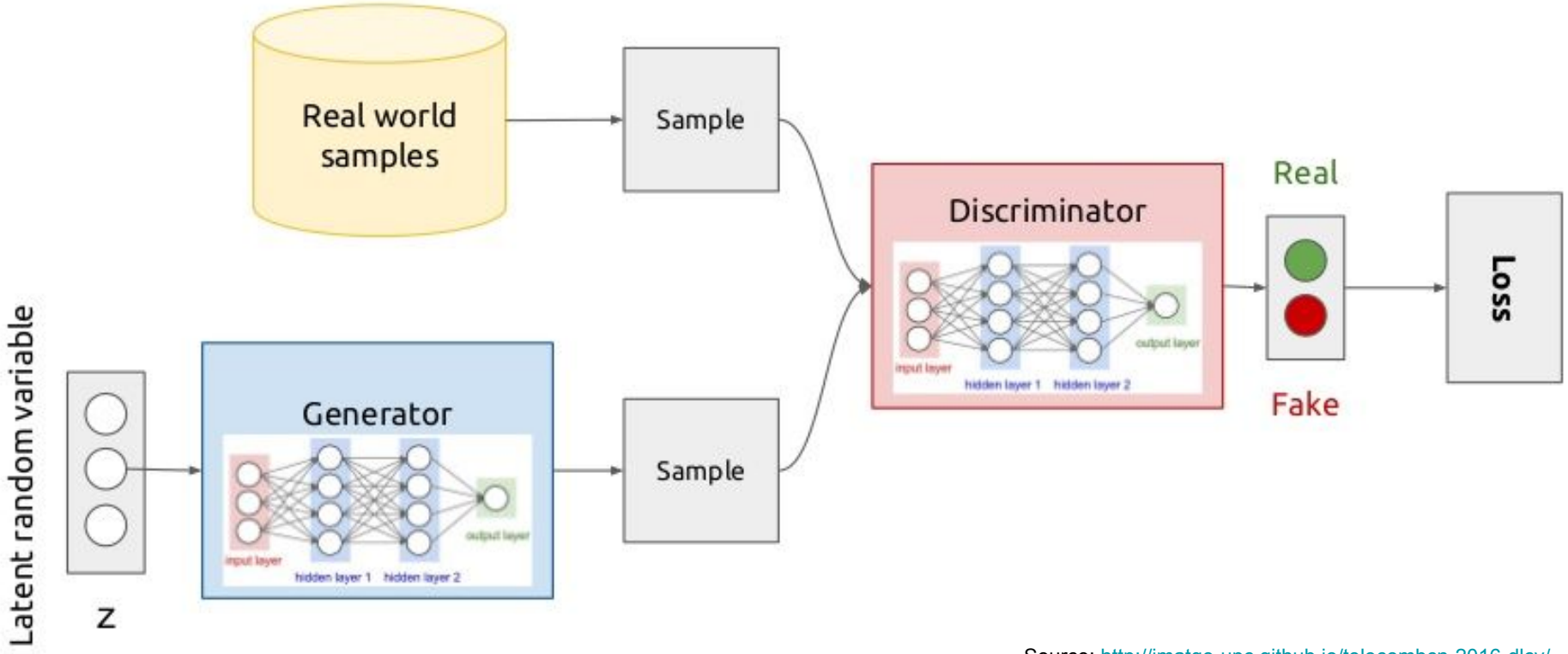
Heroes of Deep Learning

Ian Goodfellow

Research Scientist at Google Brain

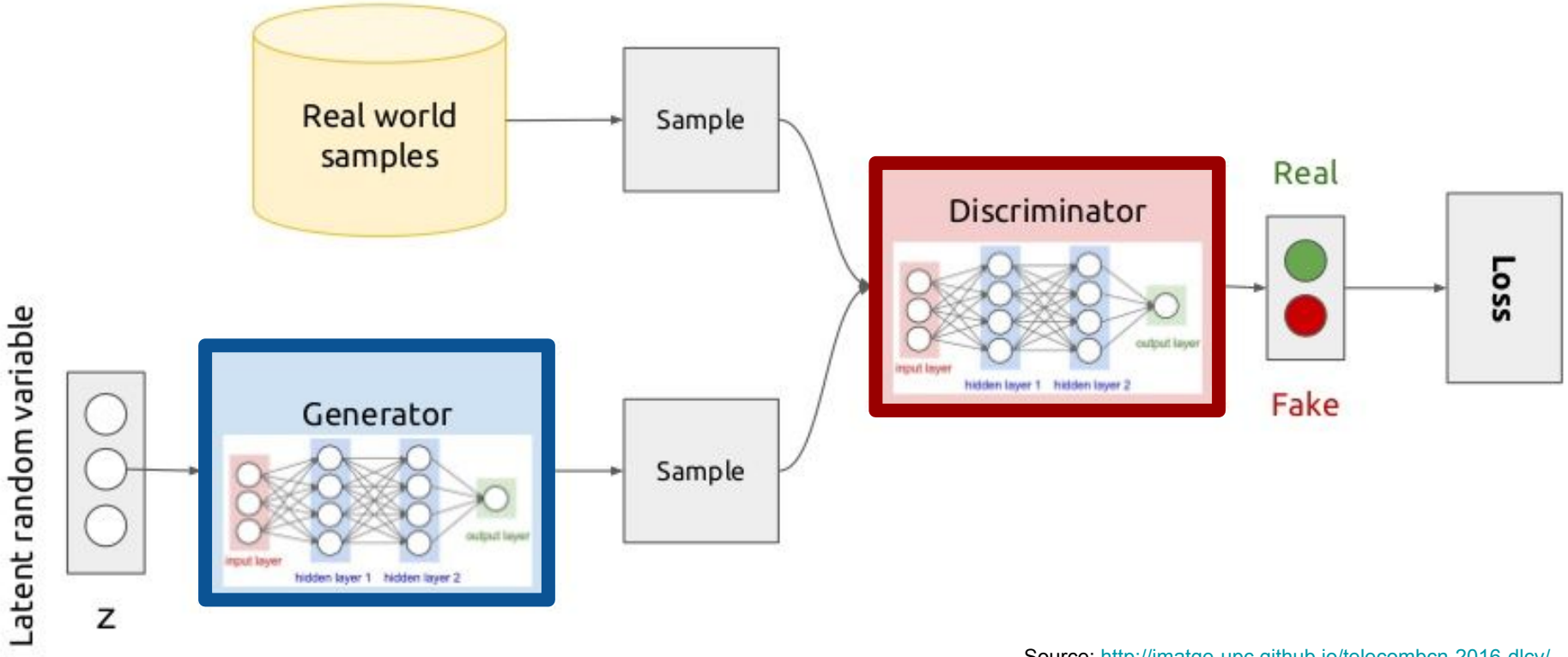
“Generative Adversarial Networks’
2014

Generative Adversarial Network (GAN)



Source: http://imatge-upc.github.io/telecombcn-2016-dlcv/

Generative Adversarial Network (GAN)



Source: http://imatge-upc.github.io/telecombcn-2016-dlcv/

Generative Adversarial nets (2014) https://arxiv.org/abs/1406.2661

64x64

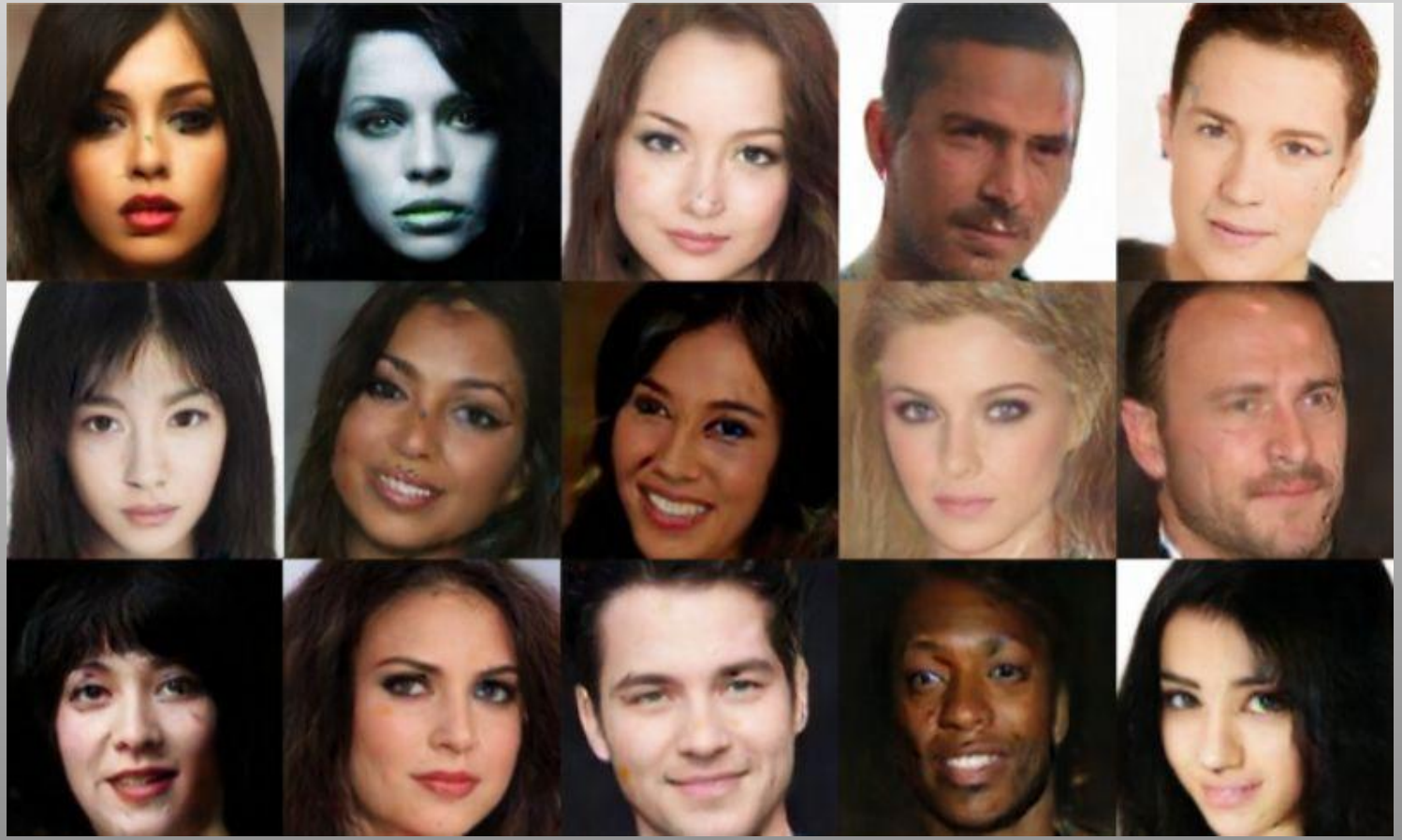
~4K values

Generative Adversarial nets (2015)

64x64

~12K values

Generative Adversarial nets (2016)



128x128

~50K values

Generative Adversarial nets (November 2017) https://arxiv.org/abs/1710.10196

1024x1024

>3000K values

5 mins Technical Deepdive

Generative Adversarial Nets

Ian J. Goodfellow, Jean Pouget-Abadie*, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, Yoshua Bengio‡
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

1 Introduction

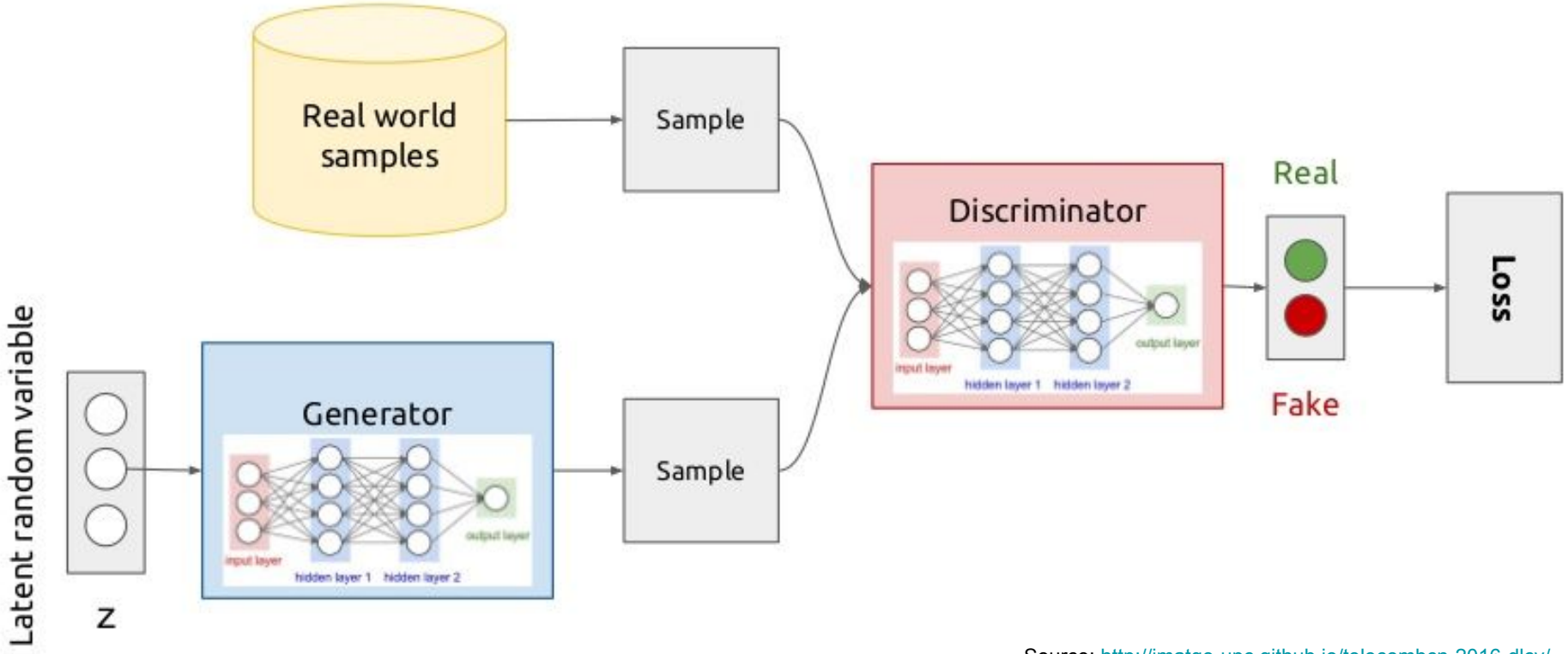
GAN's 101

A Generative Adversarial Network (GAN) involves Generator (**G**) and Discriminator (**D**) networks whose purpose, respectively, is to map random noise to samples and discriminate real and generated samples. Formally, the GAN objective, in its original form (Goodfellow et al., 2014) involves finding a Nash equilibrium to the following two player min-max problem:

$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))], \tag{1}$$

where $\boldsymbol{z} \in \mathbb{R}^{d_z}$ is a latent variable drawn from distribution $p(\boldsymbol{z})$ such as $\mathcal{N}(0, I)$ or $\mathcal{U}[-1, 1]$. When applied to images, **G** and **D** are usually convolutional neural networks (Radford et al., 2016). Without auxiliary stabilization techniques, this training procedure is notoriously brittle, requiring finely-tuned hyperparameters and architectural choices to work at all.

Generative Adversarial Network (GAN)



Source: http://imatge-upc.github.io/telecombcn-2016-dlcv/

GAN objective function

$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}\ (x)}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

GAN objective function

$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}\ (\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

- Minimize this loss wrt the Generator parameters
- Maximize this loss wrt the Discriminator parameters

GAN objective function

$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$$

- The Discriminator network outputs a single scalar value D(x) per image, which indicates how likely it is that x is a real image
- It does the same for generated images G(z), outputting a score D(G(z)) for fake images
- Our data is 'labelled' in the sense that: real images have label 1, fake images have label 0

GAN objective function

$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

We want the **Discriminator** to:

GAN objective function

$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$$

We want the **Discriminator** to:

- Recognize real images x as 'real' → output a high value (close to 1)
- Recognize fake images G(z) as 'fake' → output a low value (close to 0)

GAN objective function

$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$$

We want the **Discriminator** to:

- Recognize real images x as 'real' → output a high value (close to 1)
- Recognize fake images G(z) as 'fake' → output a low value (close to 0)

GAN objective function

$$\min_{G} \max_{D} \mathbb{E}_{x \sim q_{\text{data}}\ (\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

We want the Generator to:

- Generate fake images G(z) that look real to the Discriminator

GAN objective function

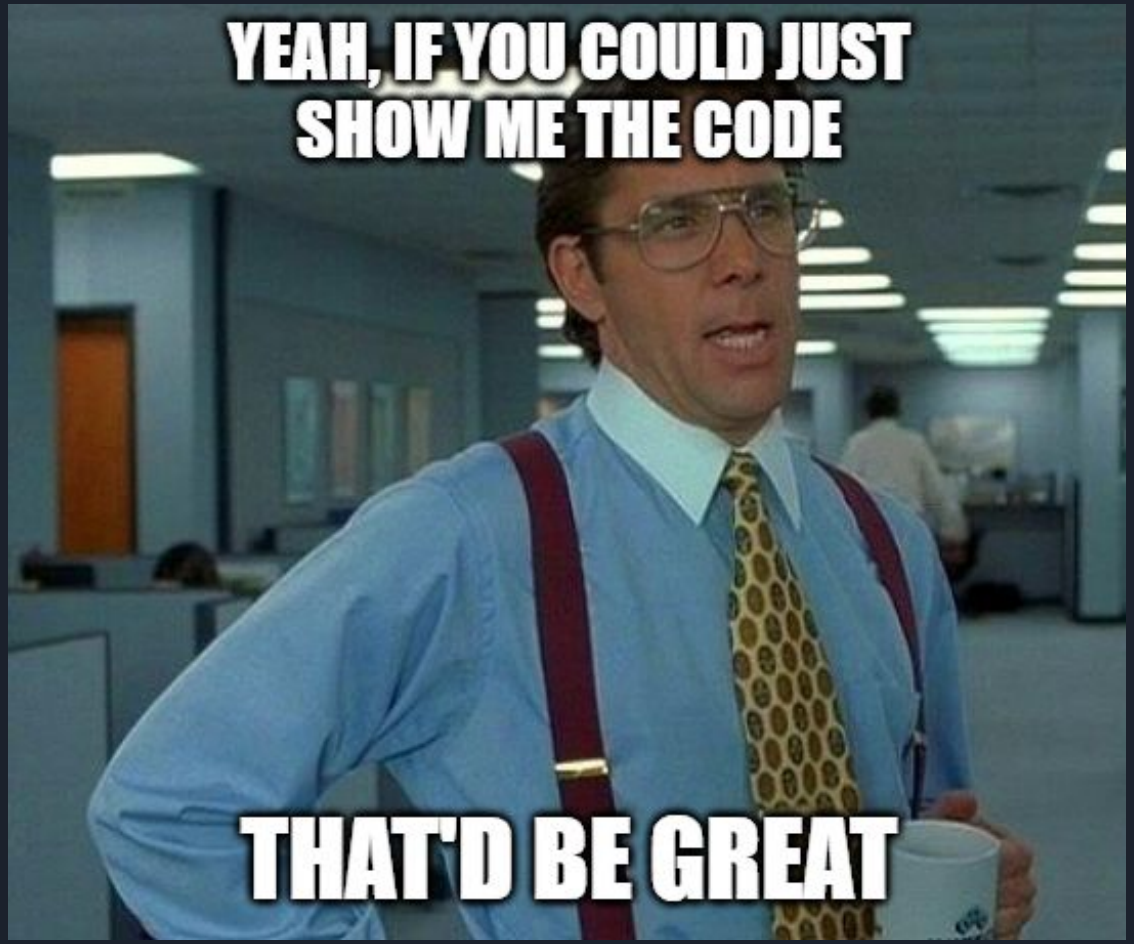
$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - D(G(z)))]$$

We want the Generator to:

- Generate fake images G(z) that look real to the Discriminator

GAN objective function

$$\min_G \max_D \mathbb{E}_{x \sim q_{\text{data}}\ (x)}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$


YEAH, IF YOU COULD JUST
SHOW ME THE CODE
THAT'D BE GREAT

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Source: https://arxiv.org/pdf/1406.2661.pdf

2.2 Wasserstein GANs

[2] argues that the divergences which GANs typically minimize are potentially not continuous with respect to the generator's parameters, leading to training difficulty. They propose instead using the *Earth-Mover* (also called Wasserstein-1) distance $W(q, p)$, which is informally defined as the minimum cost of transporting mass in order to transform the distribution q into the distribution p (where the cost is mass times transport distance). Under mild assumptions, $W(q, p)$ is continuous everywhere and differentiable almost everywhere.

The WGAN value function is constructed using the Kantorovich-Rubinstein duality [25] to obtain

$$\min_G \max_{D \in \mathcal{D}} \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r} \left[D(\boldsymbol{x}) \right] - \mathbb{E}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} \left[D(\tilde{\boldsymbol{x}}) \right] \tag{2}$$

where $\mathcal{D}$ is the set of 1-Lipschitz functions and $\mathbb{P}_g$ is once again the model distribution implicitly defined by $\tilde{\boldsymbol{x}} = G(\boldsymbol{z}), \quad \boldsymbol{z} \sim p(\boldsymbol{z})$. In that case, under an optimal discriminator (called a *critic* in the paper, since it's not trained to classify), minimizing the value function with respect to the generator parameters minimizes $W(\mathbb{P}_r, \mathbb{P}_g)$.

The WGAN value function results in a critic function whose gradient with respect to its input is better behaved than its GAN counterpart, making optimization of the generator easier. Empirically, it was also observed that the WGAN value function appears to correlate with sample quality, which is not the case for GANs [2].


Nowadays:
Many different flavors...

GAN Type	Key Take-Away
GAN	The original (JSD divergence)
WGAN	EM distance objective
Improved WGAN	No weight clipping on WGAN
LSGAN	L2 loss objective
RWGAN	Relaxed WGAN framework
McGAN	Mean/covariance minimization objective
GMMN	Maximum mean discrepancy objective
MMD GAN	Adversarial kernel to GMMN
Cramer GAN	Cramer distance
Fisher GAN	Chi-square objective
EBGAN	Autoencoder instead of discriminator
BEGAN	WGAN and EBGAN merged objectives
MAGAN	Dynamic margin on hinge loss from EBGAN

GAN Objective Functions: GANs and Their Variations

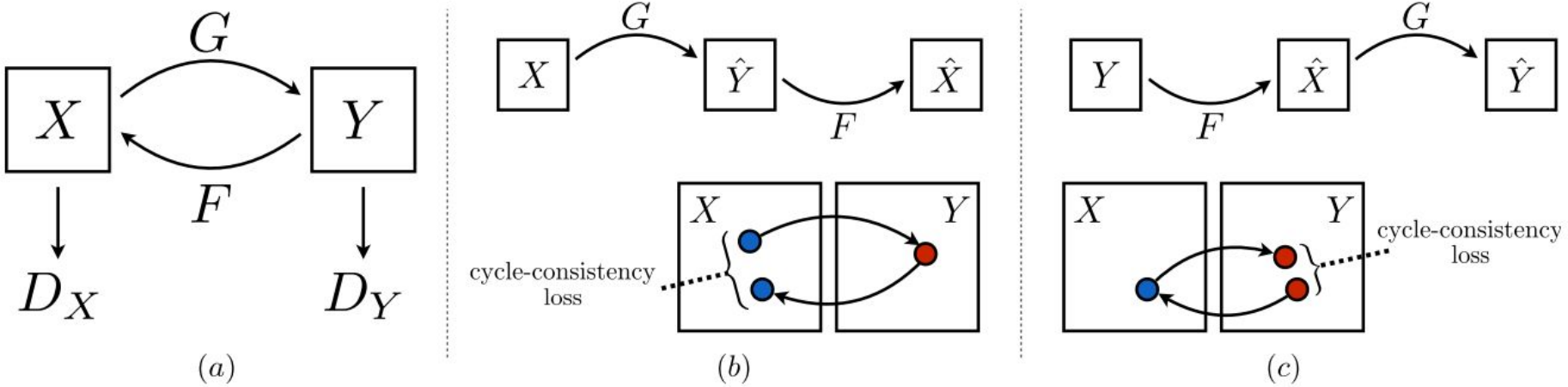
There are hundreds of types of GANs. How does an objective function play into what a GAN looks like?

Hunter Heidenreich Follow

Aug 23, 2018 · 12 min read

Cycle-GAN

Cycle-consistency loss:



Cycle-GAN

BigGan (Google)

BigGan specifics

- “We demonstrate that GANs benefit dramatically from scaling, and train models with two to four times as many parameters and eight times the batch size compared to prior art.”

- Each model is trained on 128 to 512 cores of a Google TPUv3 Pod (Google, 2018)

BigGan “Deep”

Table 9: BigGAN-deep architecture for 512×512 images.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(128 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock $4ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock up $2ch \rightarrow ch$
ResBlock $ch \rightarrow ch$
ResBlock up $ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{512 \times 512 \times 3}$
3×3 Conv $3 \rightarrow ch$
ResBlock down $ch \rightarrow ch$
ResBlock $ch \rightarrow ch$
ResBlock down $ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock down $2ch \rightarrow 4ch$
ResBlock $4ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock down $4ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed$(y) \cdot \boldsymbol{h}$ + (linear $\rightarrow 1$)

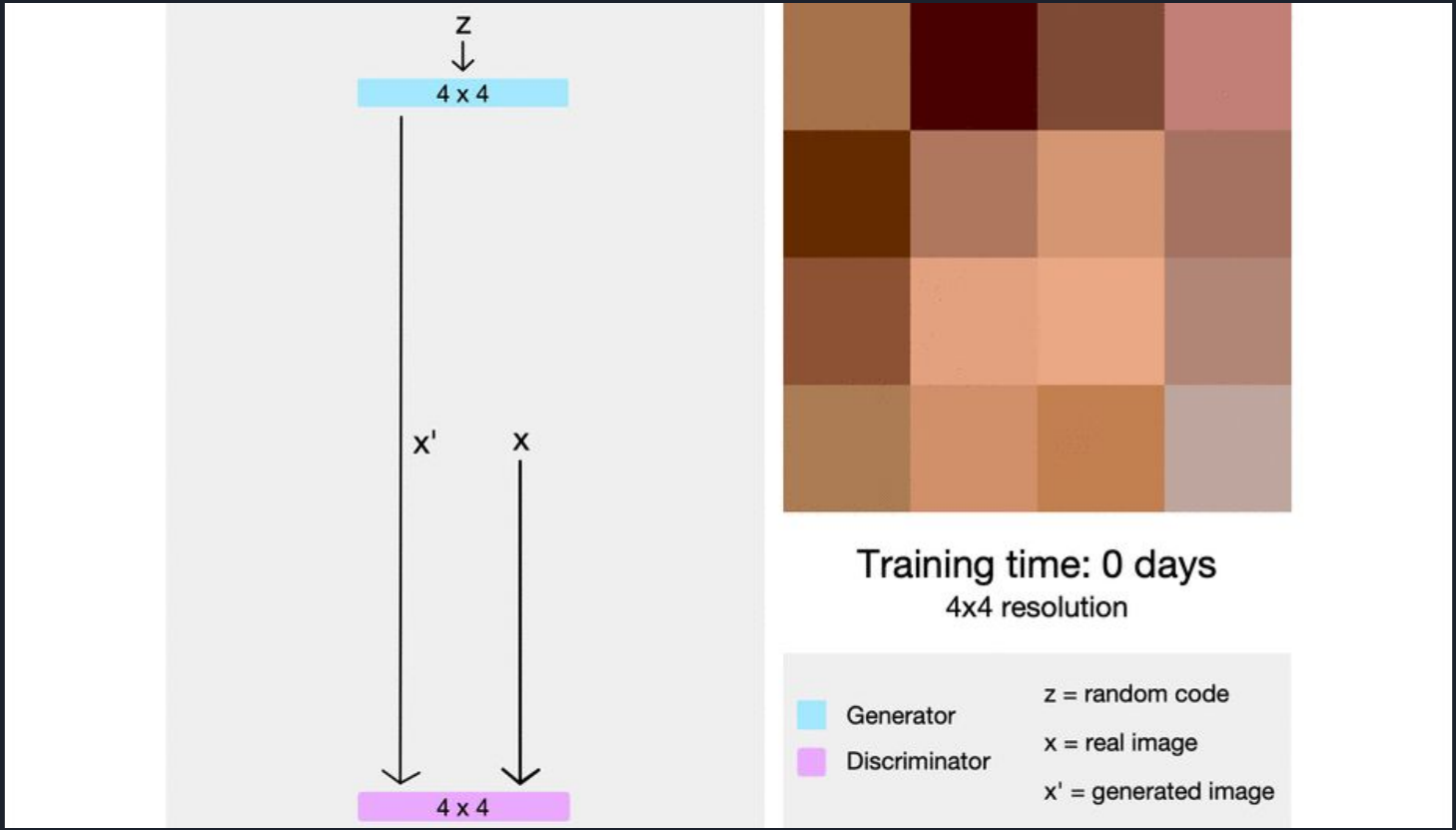
(b) Discriminator

BigGAN interpolations




NVIDIA
GAME OF
Gans

Progressive growing

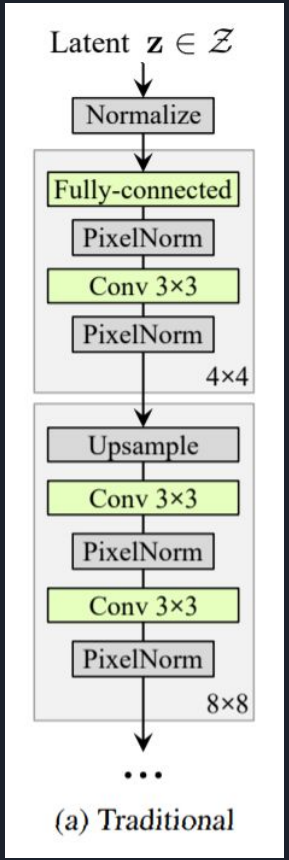


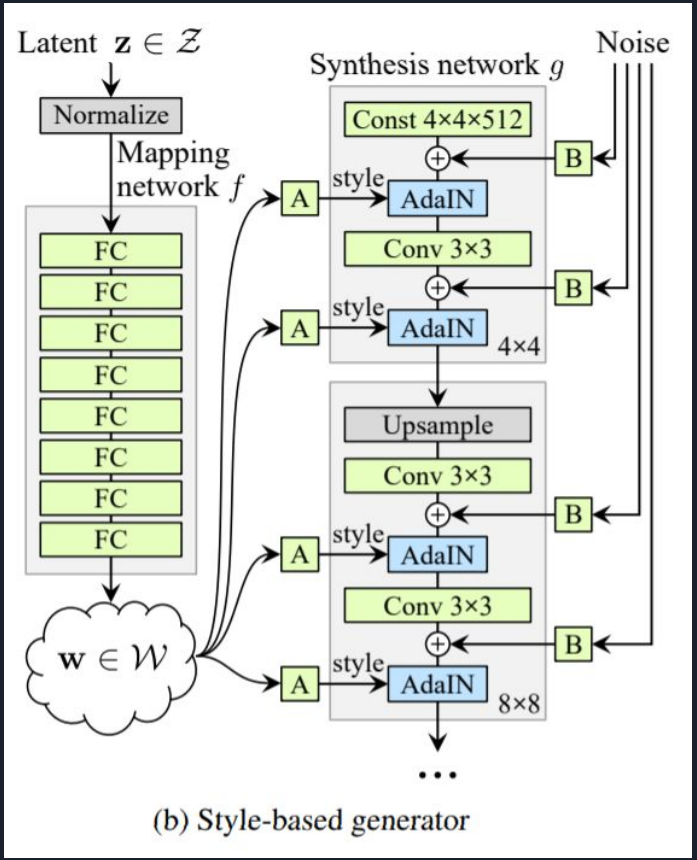

4x4

Style-GAN by Nvidia (January 2019)

Coarse styles
$(4^2 – 8^2)$

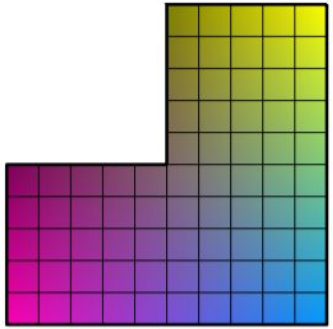
StyleGAN architecture:



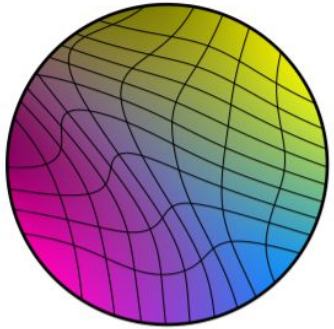


Why would you... ?

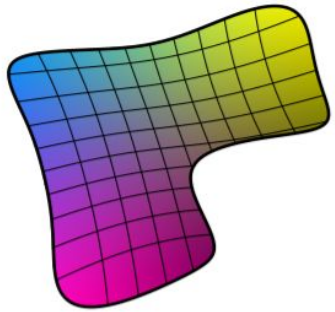
Why would you... ?



(a) Distribution of features in training set



(b) Mapping from $\mathcal{Z}$ to features



(c) Mapping from $\mathcal{W}$ to features

Figure 6. Illustrative example with two factors of variation (image features, e.g., masculinity and hair length). (a) An example training set where some combination (e.g., long haired males) is missing. (b) This forces the mapping from $\mathcal{Z}$ to image features to become curved so that the forbidden combination disappears in $\mathcal{Z}$ to prevent the sampling of invalid combinations. (c) The learned mapping from $\mathcal{Z}$ to $\mathcal{W}$ is able to "undo" much of the warping.

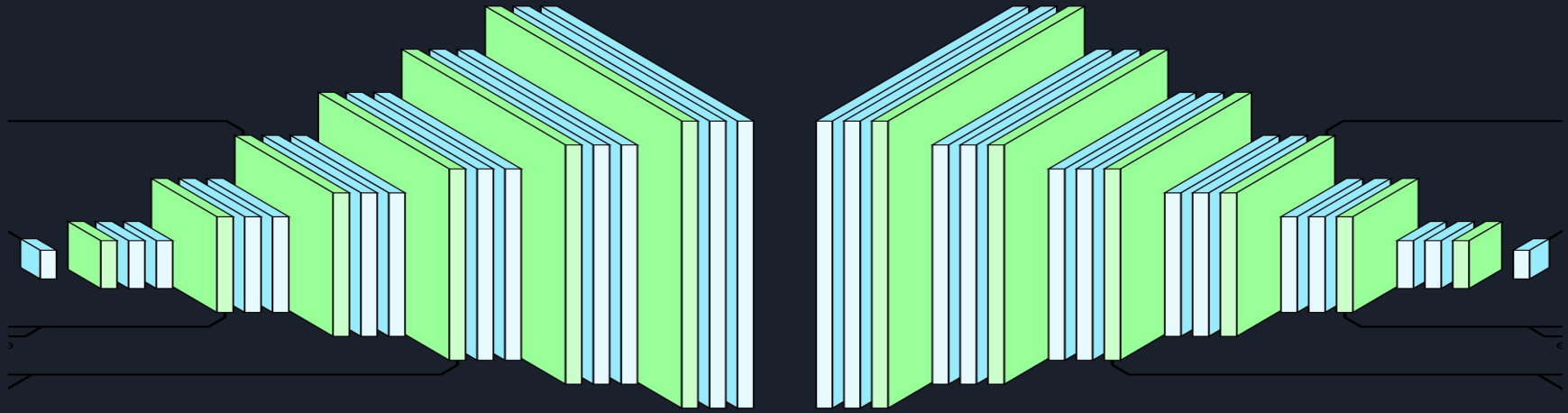
Part II:

Hands on fun with StyleGAN

→ Download the Colab Notebook and do it yourself!

Playing with StyleGAN

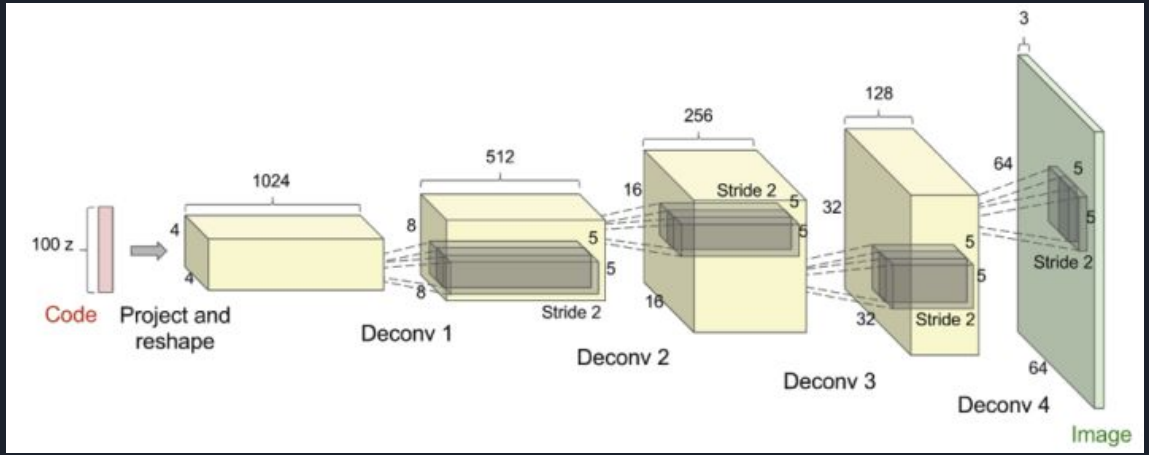
- StyleGAN's latent space has STRUCTURE
- This structure is learned fully unsupervised during the adversarial training process
- How can we leverage this structure?



Playing with StyleGAN

1. Instead of manipulating images in the pixel domain, let's manipulate them in the latent space!
2. To do this, we first need to find a query image inside StyleGAN's latent space

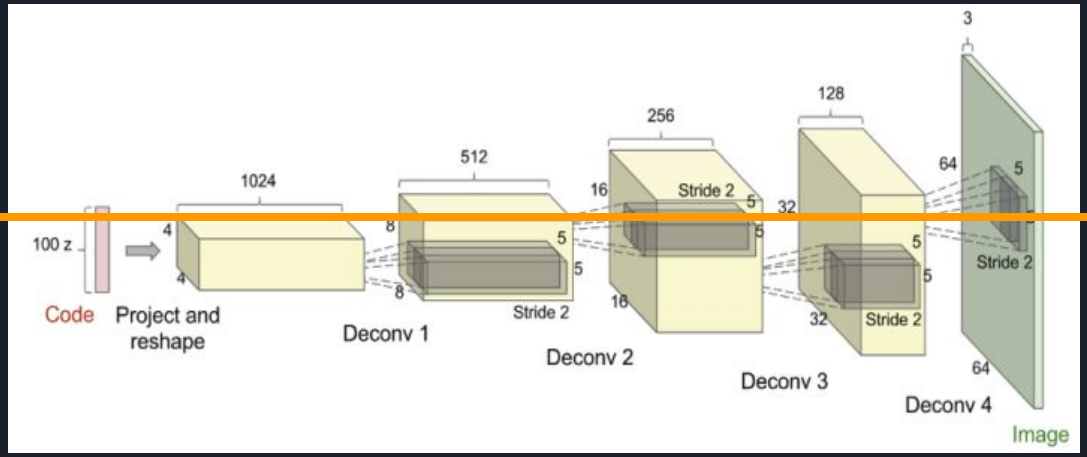
Structured latent space



Big mess of pixels

https://bit.ly/2ZWFtQw

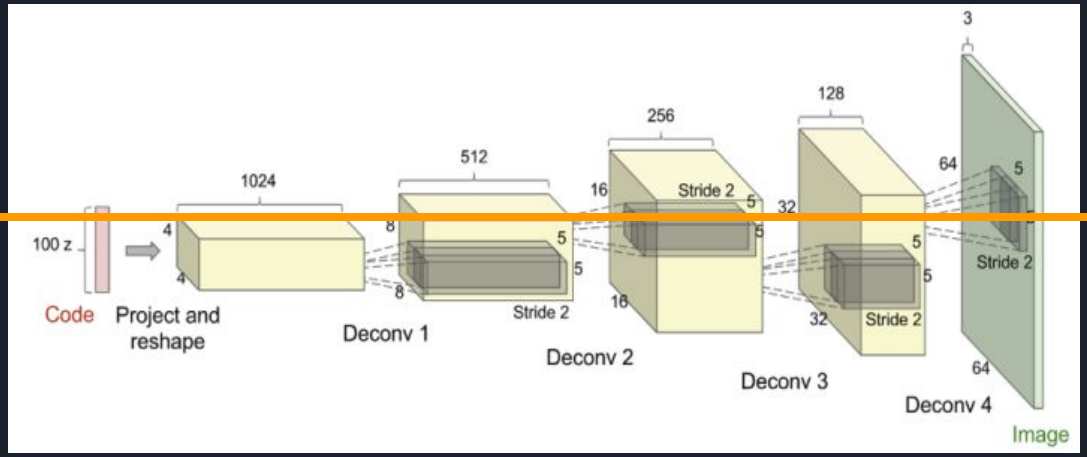
Given a query image,
how to find the latent code?



Unknown latent code

Query Image

Given a query image, how to find the latent code?

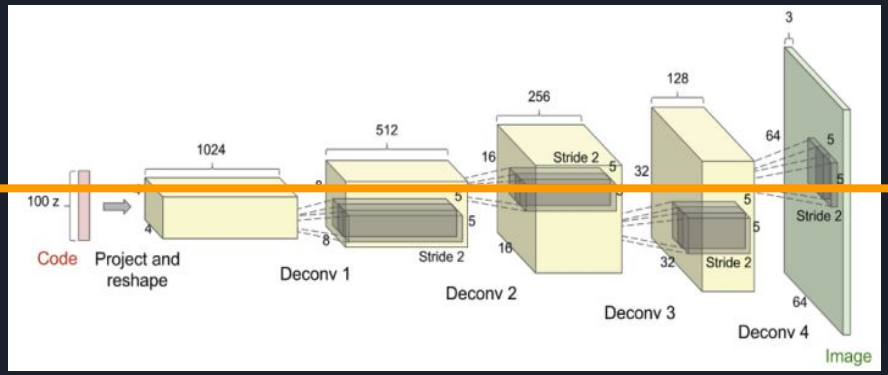


Random latent code

Random Image

How to find the latent code?

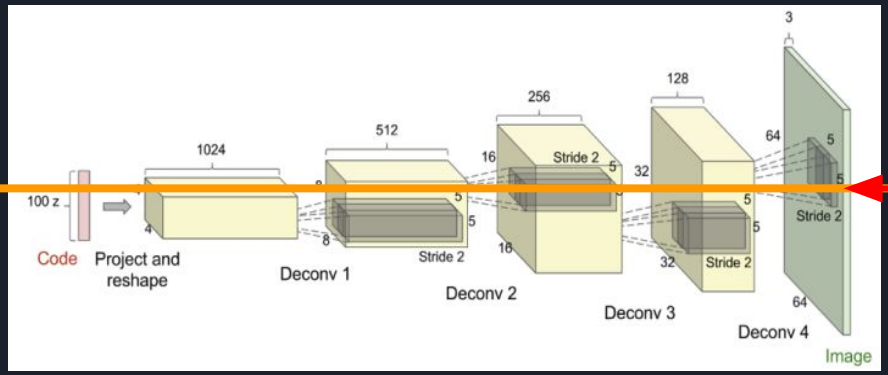
Current latent code



Current Image

Gradient Descent bitches!

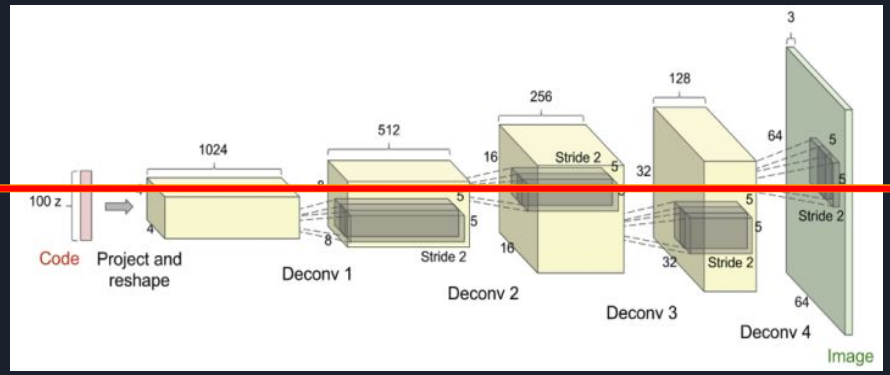
Current latent code



Current Image

L2-Loss in pixels

Gradient Descent bitches!



Current latent code

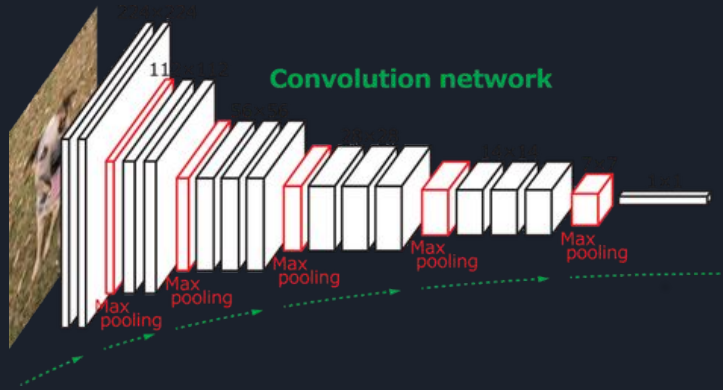
Current Image

L2-Loss in pixels

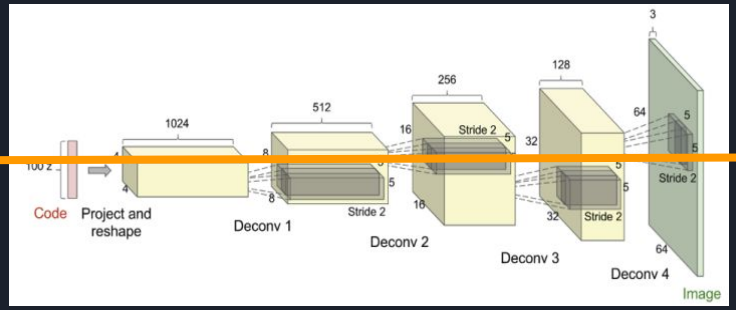
Well... Not really

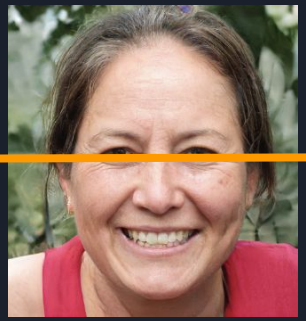
- L2-Pixel loss alone gives bad results
- The optimization gets stuck in very bad local optima

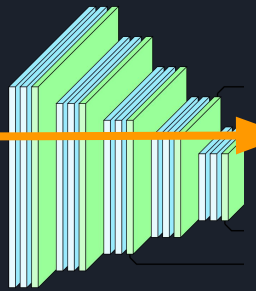
- We need some bigger guns...
- VGG!



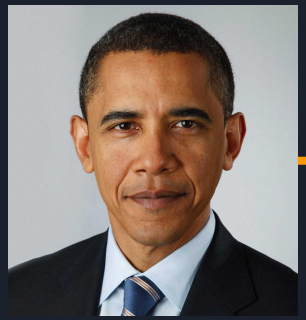
Idea:
use a pretrained VGG as a ‘lens’

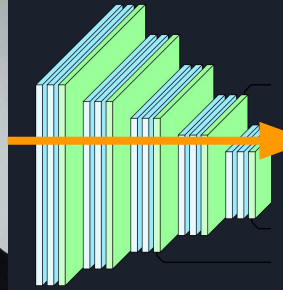






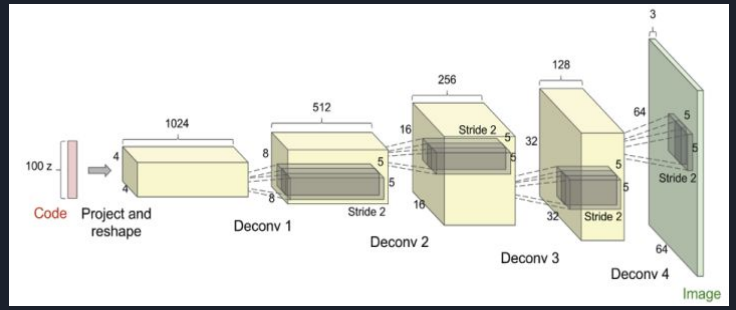
Semantic Feature vector

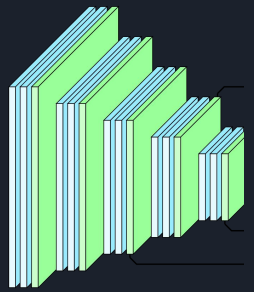




(Fixed) semantic Feature vector

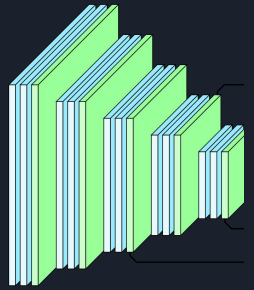
Idea:
use a pretrained VGG as a ‘lens’





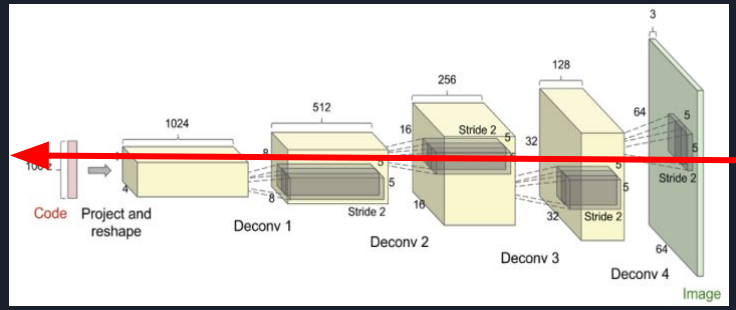
Semantic Feature vector

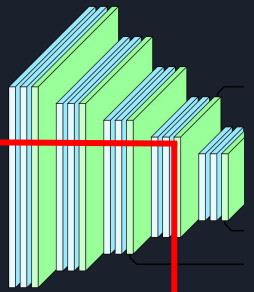
L2-Loss in semantic ‘VGG’ space



(Fixed) semantic Feature vector

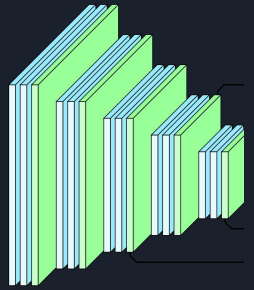
Idea:
use a pretrained VGG as a ‘lens’





Semantic Feature vector

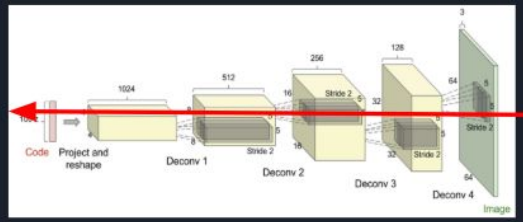
L2-Loss in semantic ‘VGG’ space

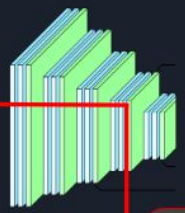


(Fixed) semantic Feature vector

Find any face inside StyleGAN’s latent space:

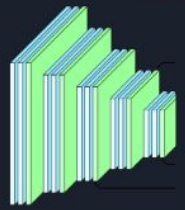
That shit is ****ing slow dude...





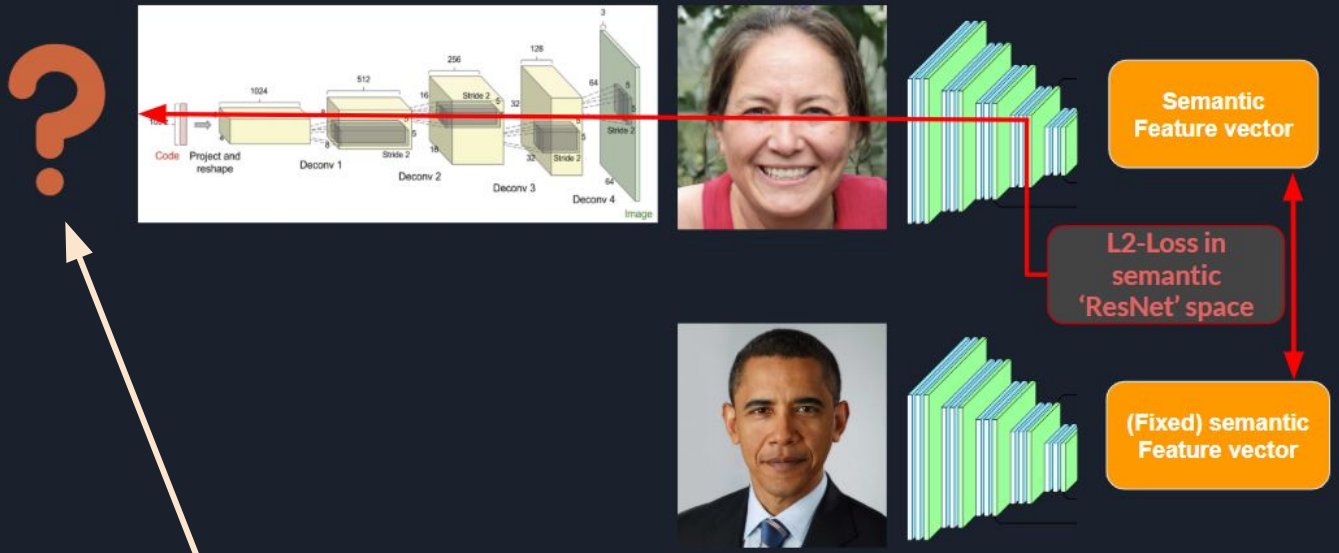
Semantic Feature vector

L2-Loss in semantic 'ResNet' space



(Fixed) semantic Feature vector

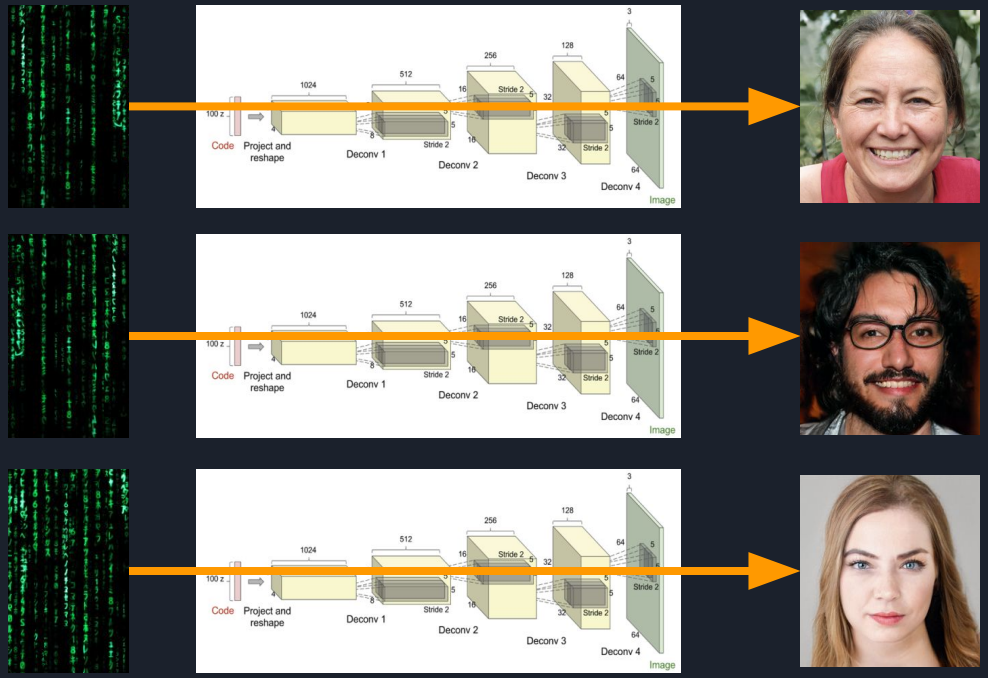
That shit is ****ing slow dude...



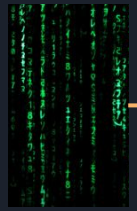
Why don't we try and make a good guess for this latent code?

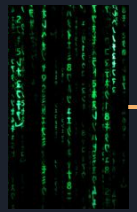
1. Random Sample

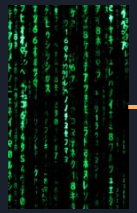
2. Generate Images

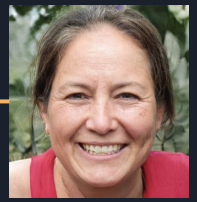


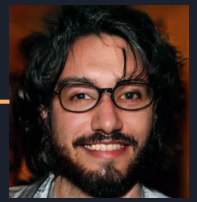
3. Collect Dataset

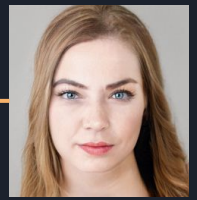






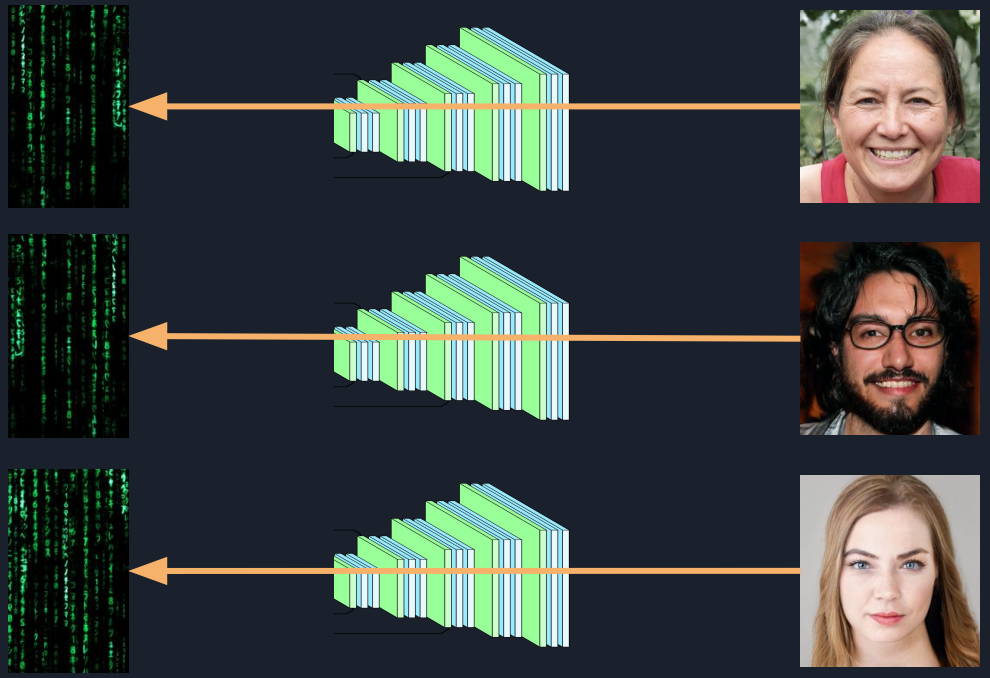






3. Collect Dataset

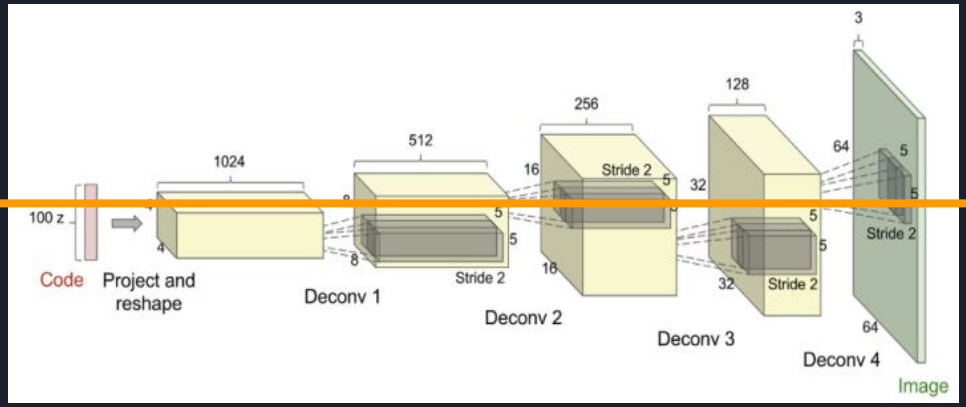
ResNets duh...

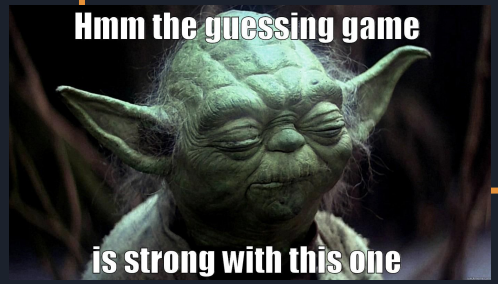


4. Train a model!!!

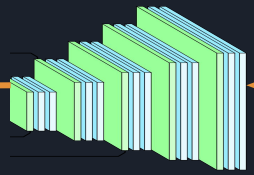
Given a query image, how to find the laten code?

Query Image

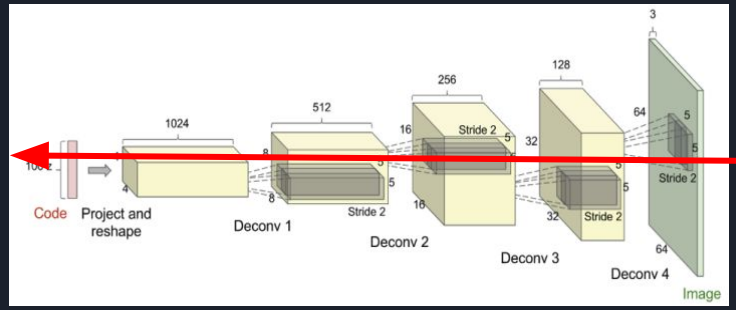


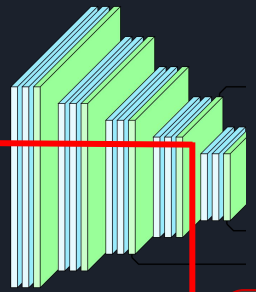


1. Guess the code



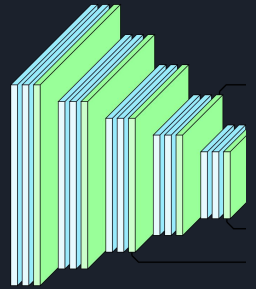
After guessing, optimize further with SGD





Semantic Feature vector

L2-Loss in semantic 'VGG' space



(Fixed) semantic Feature vector

train_resnet.py Restore all ResNet options, add smaller option 3 months ago

README.md

StyleGAN — Encoder for Official TensorFlow Implementation

This is my StyleGAN Encoder; there are many like it, but this one is mine. Thanks to @Puzer for the original, of which this is a fork, and to @SimJeg for the initial code that formed the basis of the ResNet model used here, and to @Pender for his fork as well!

From left to right: original image, predicted image from a ResNet trained on generated StyleGAN faces, and the final encoded image.

What I've added:

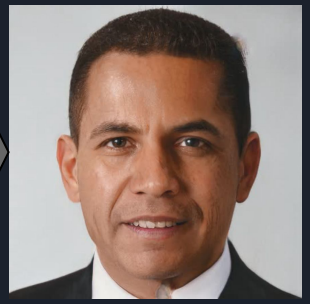
Additional tweaks:

- Add a L1-penalty to the latent code to make sure it’s close to StyleGANs concept of a “face”
- Combination of the VGG semantic loss + Normal L2-pixel loss
- Tunable VGG feature layer: 8,9,10, …
- MS-SIM perceptual loss
- Learning rate decay during optimization
- Possibility to apply a face-mask before computing losses
- ...

Latent space optimization

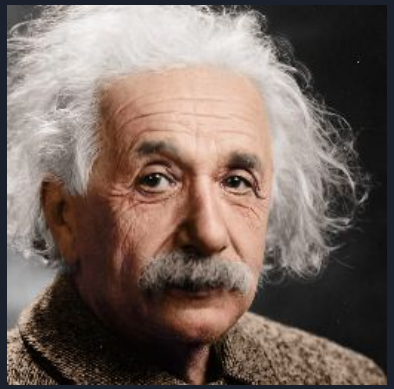
Query Image

Initial guess

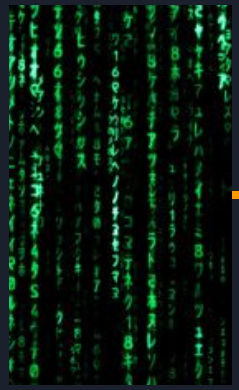


Optimized Image

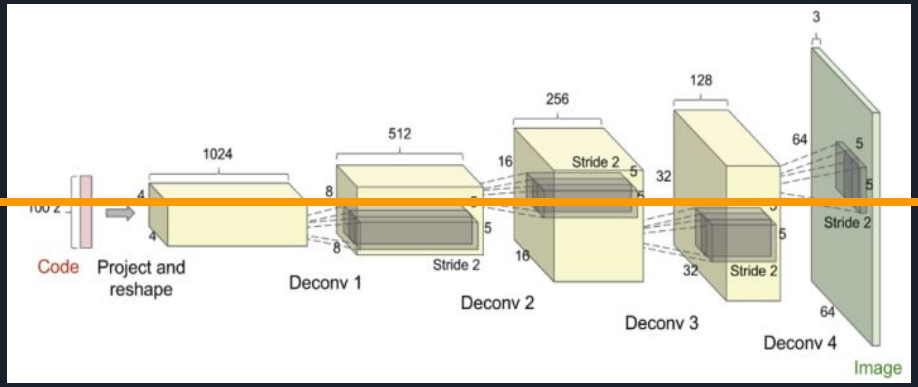


Ok, so we have the code, now what?



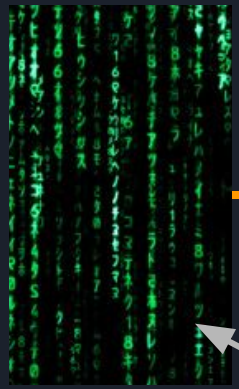
Optimized latent code

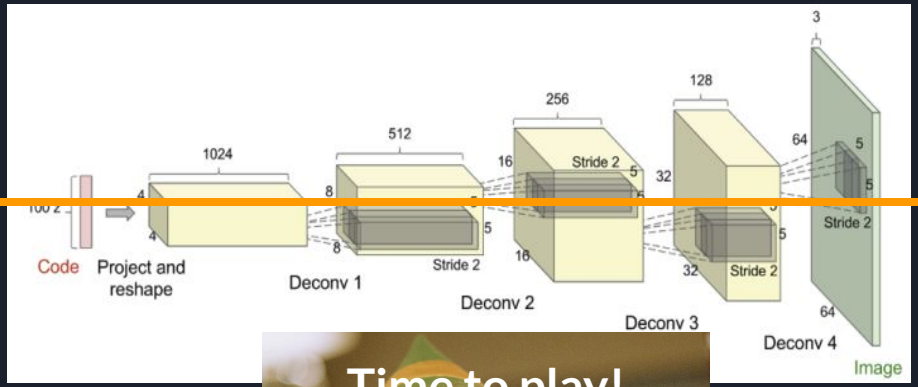


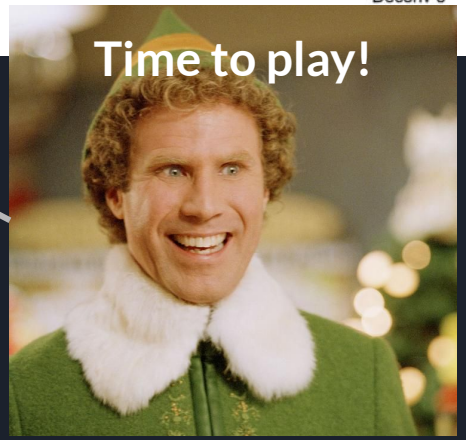
Optimized Image

Query Image

Ok, so we have the code, now what?



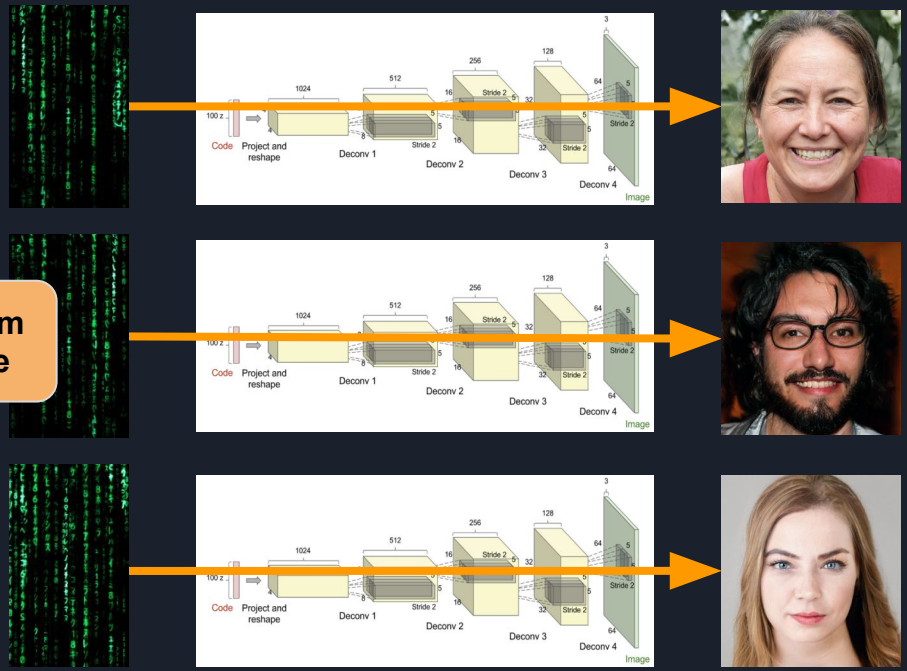




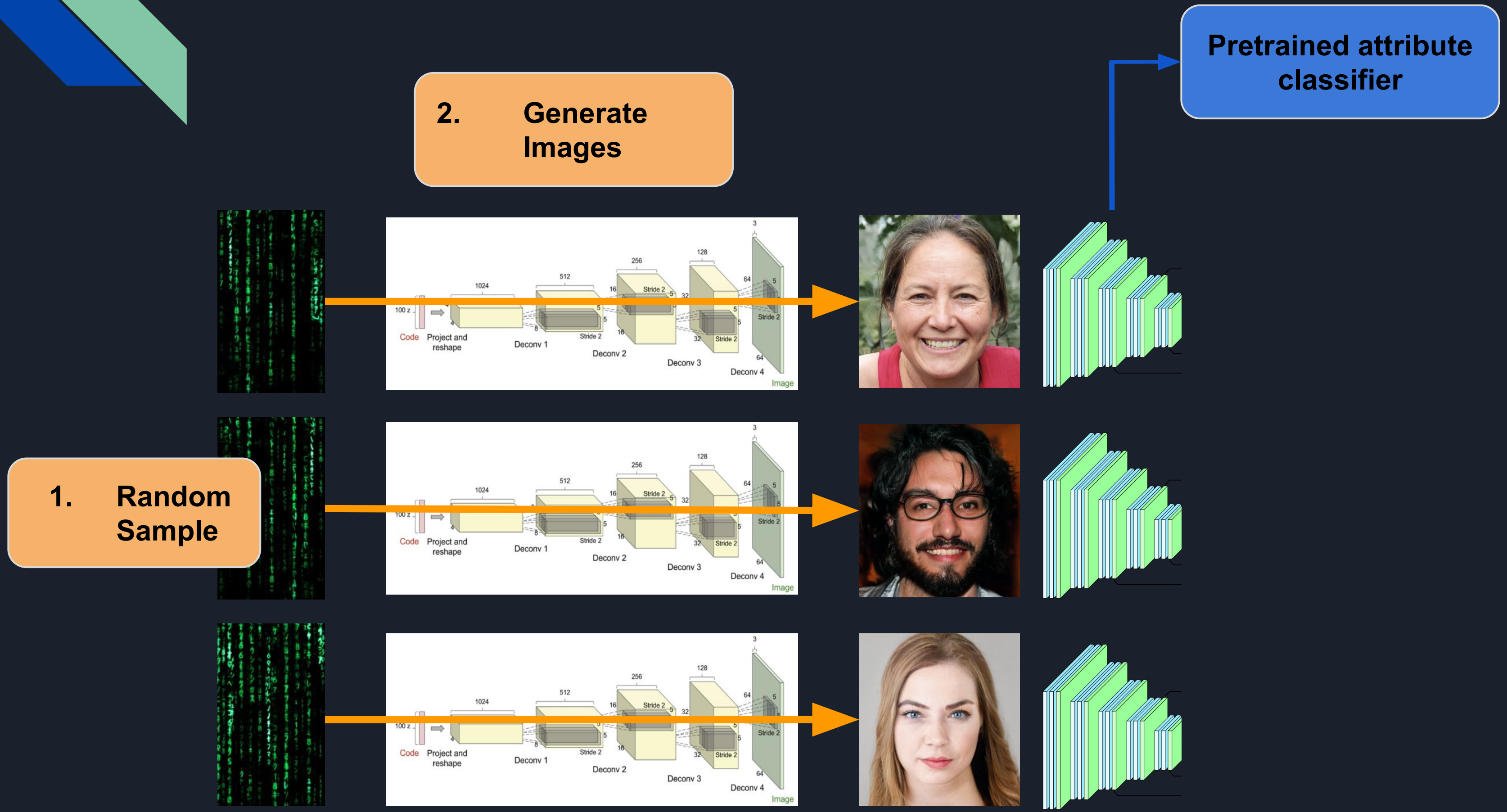
Let’s create another dataset!

2. Generate Images

1. Random Sample



Let’s create another dataset!

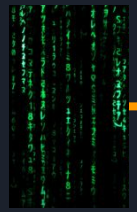


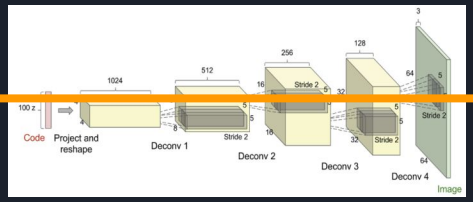
Let’s create another dataset!

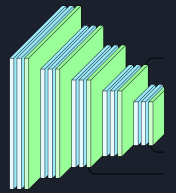
1. Random Sample

2. Generate Images

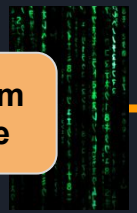
3. Create a dataset

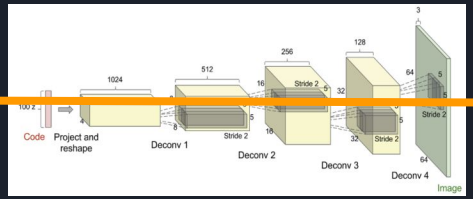


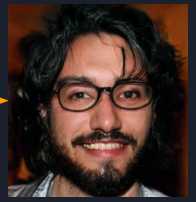


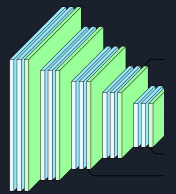


- ‘Woman’
- 61 years old
- ‘Smiling’
- ...

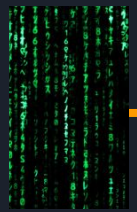


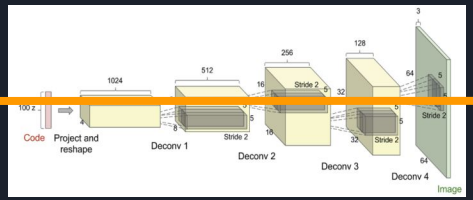


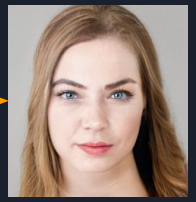


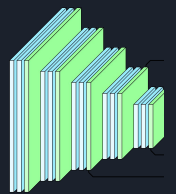


- ‘Man’
- 35 years old
- ‘Smiling’
- ‘Glasses’





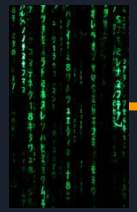




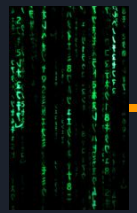
- ‘Woman’
- 32 years old
- ‘Neutral’
- ...

Let’s create another dataset!

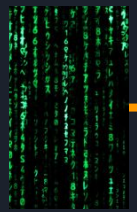
Create a dataset!



- ‘Woman’
- 61 years old
- ‘Smiling’
- ...

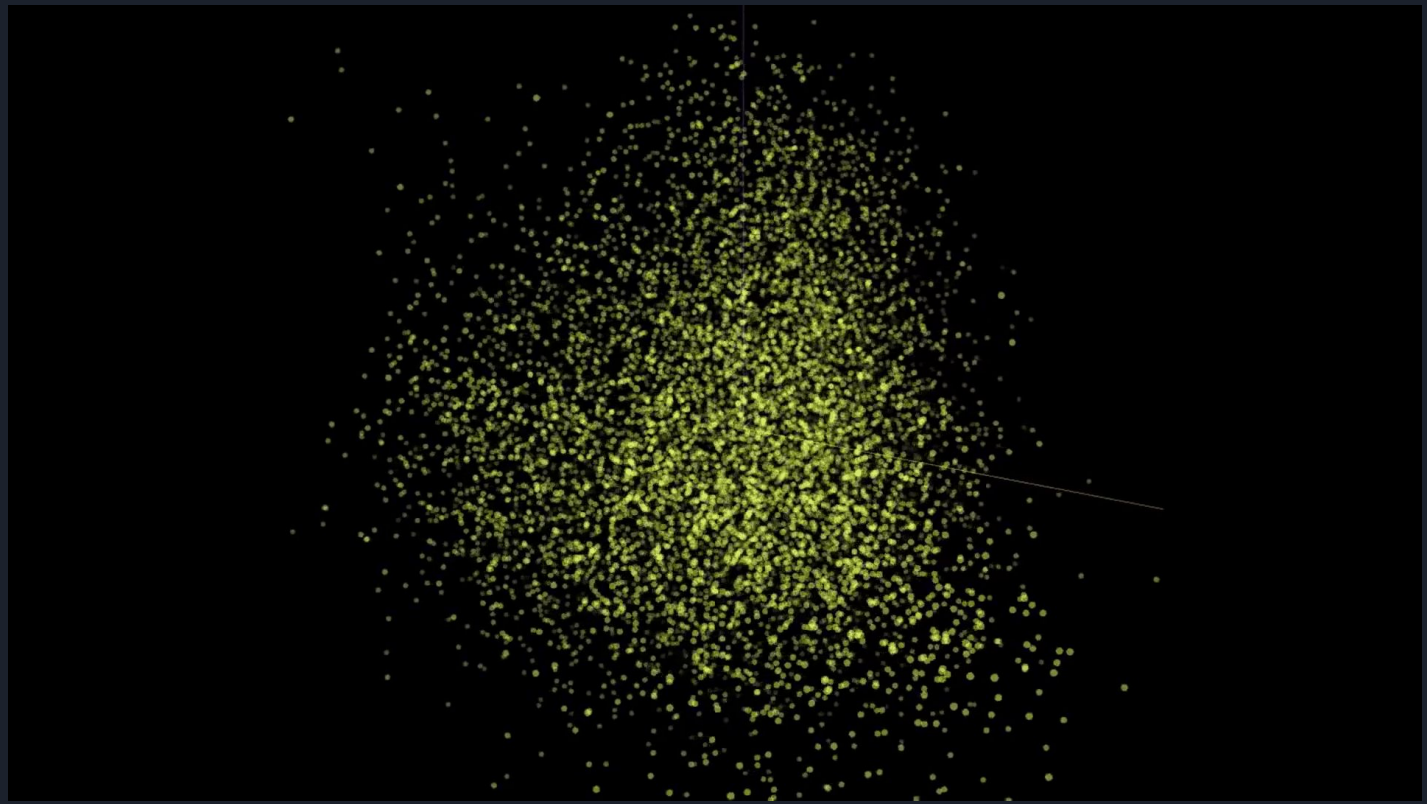


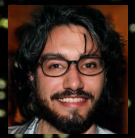
- ‘Man’
- 35 years old
- ‘Smiling’
- ‘Glasses’



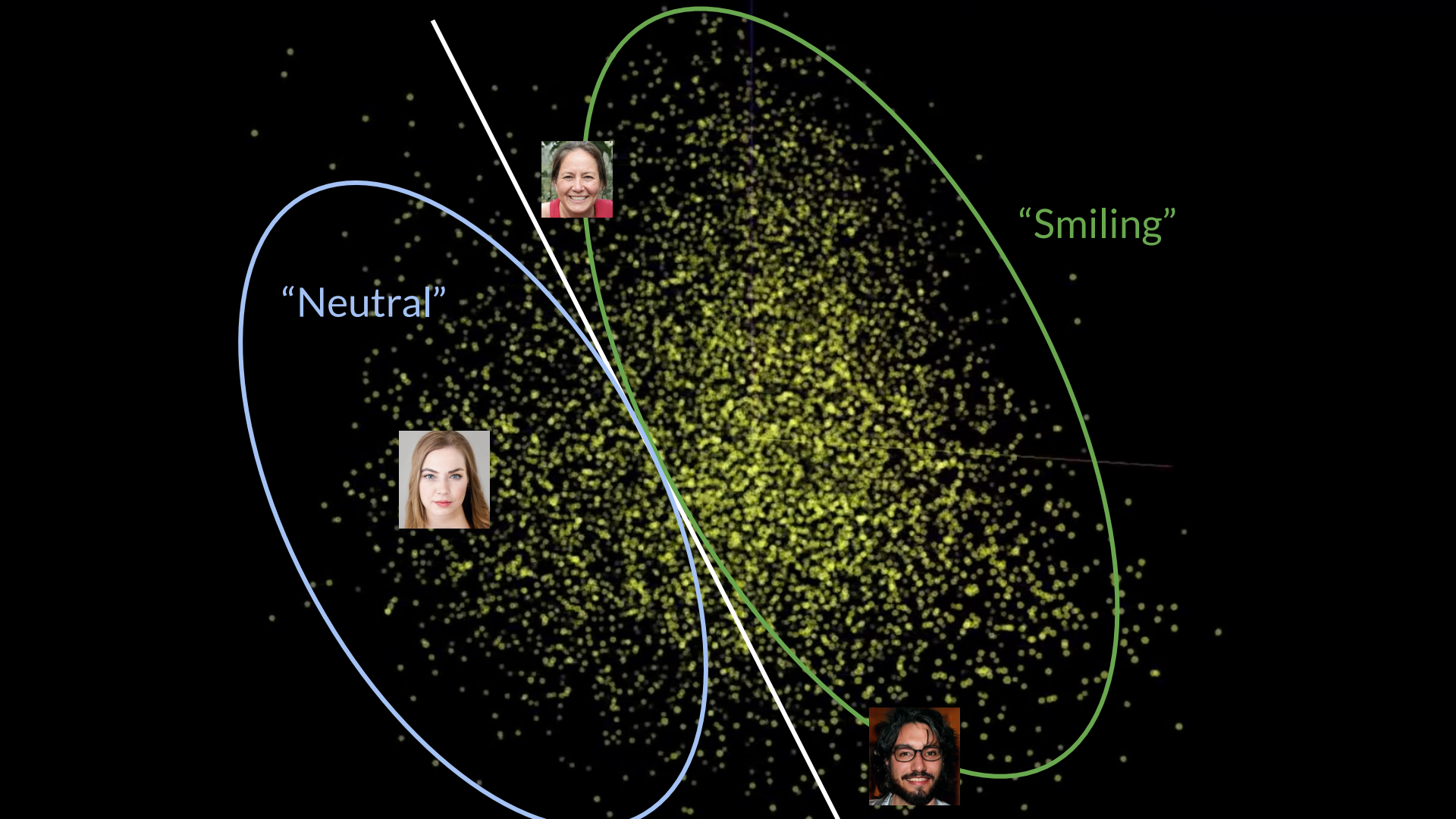
- ‘Woman’
- 32 years old
- ‘Neutral’
- ...

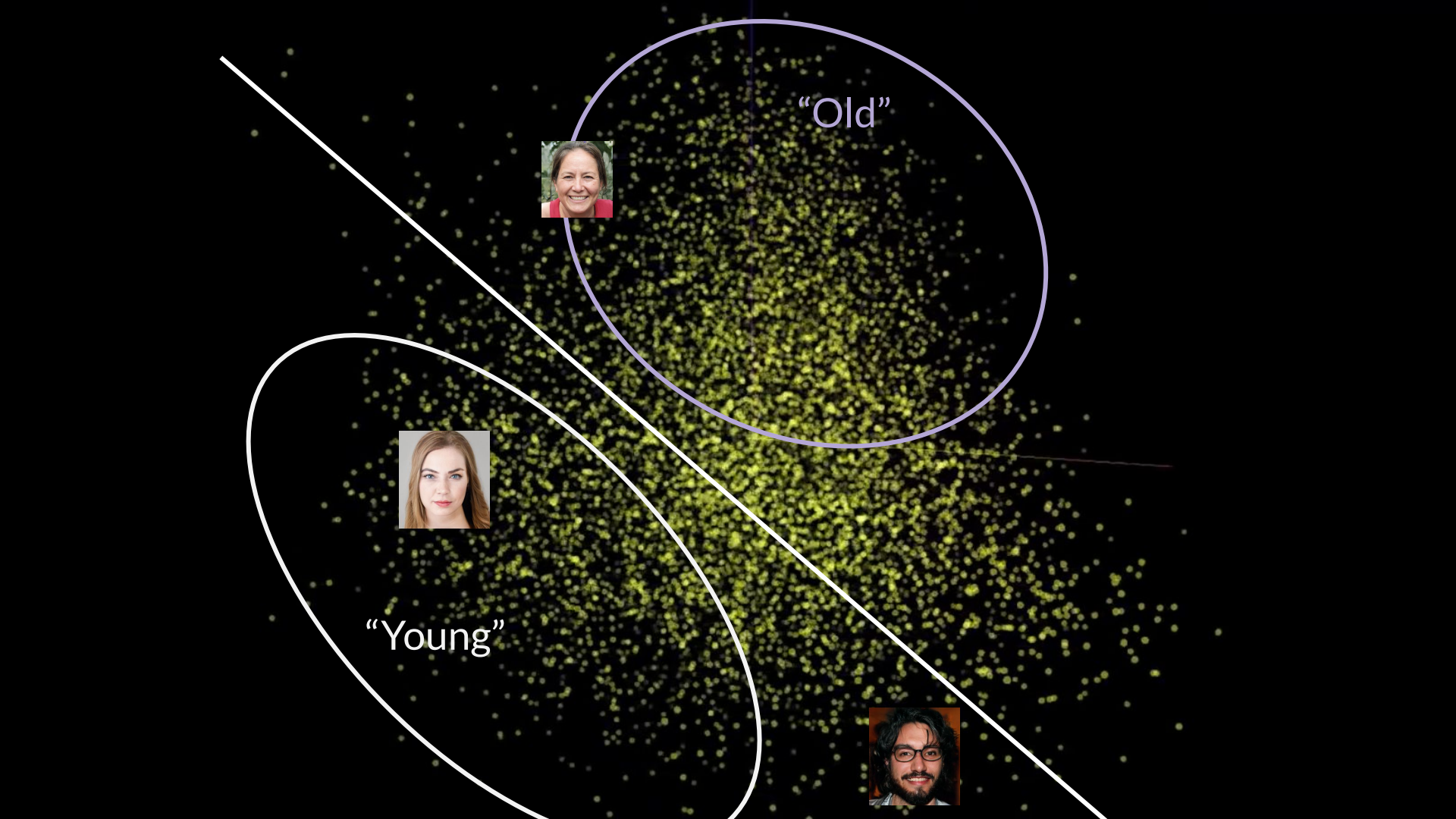
StyleGAN’s latent universe:



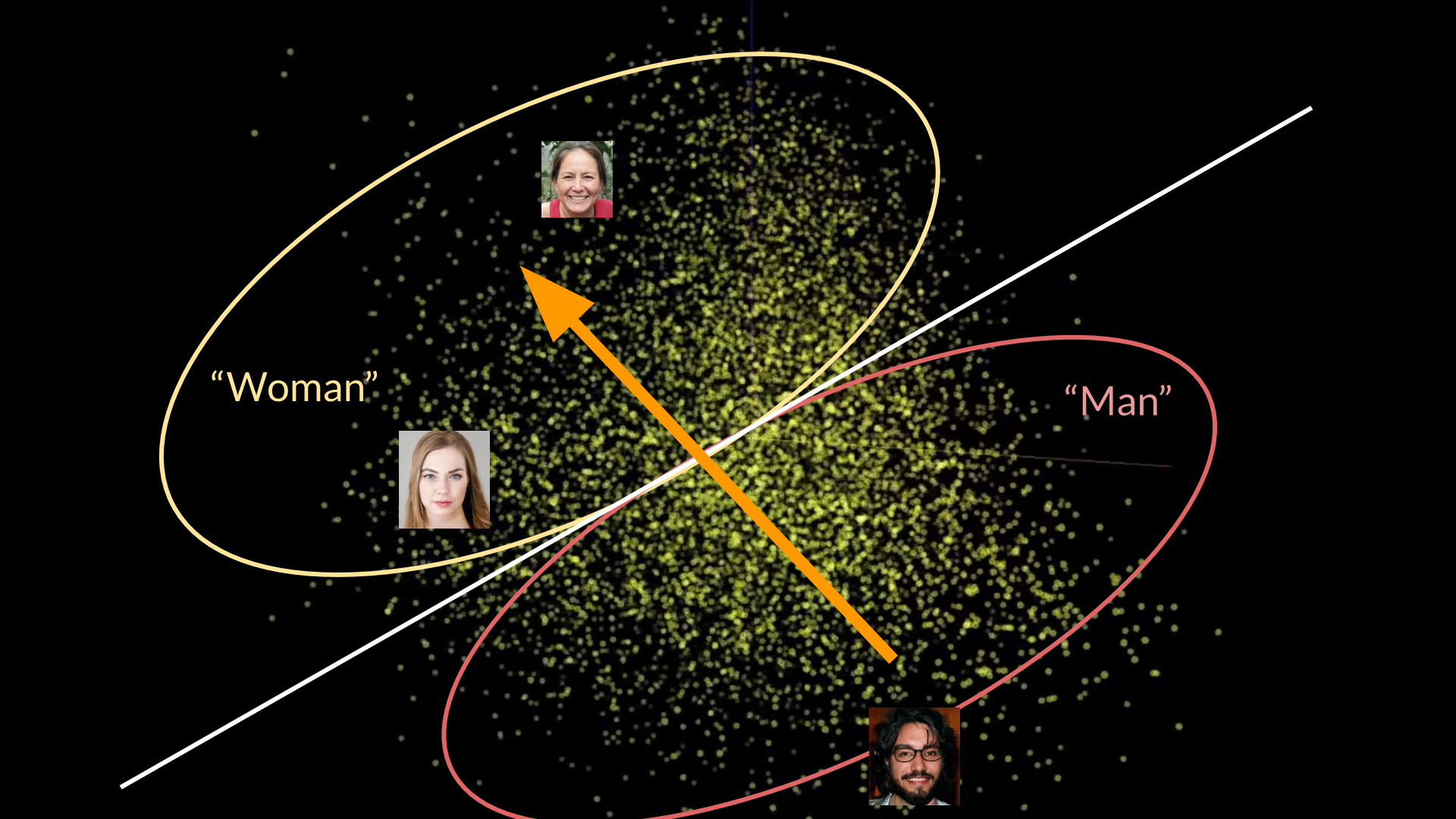


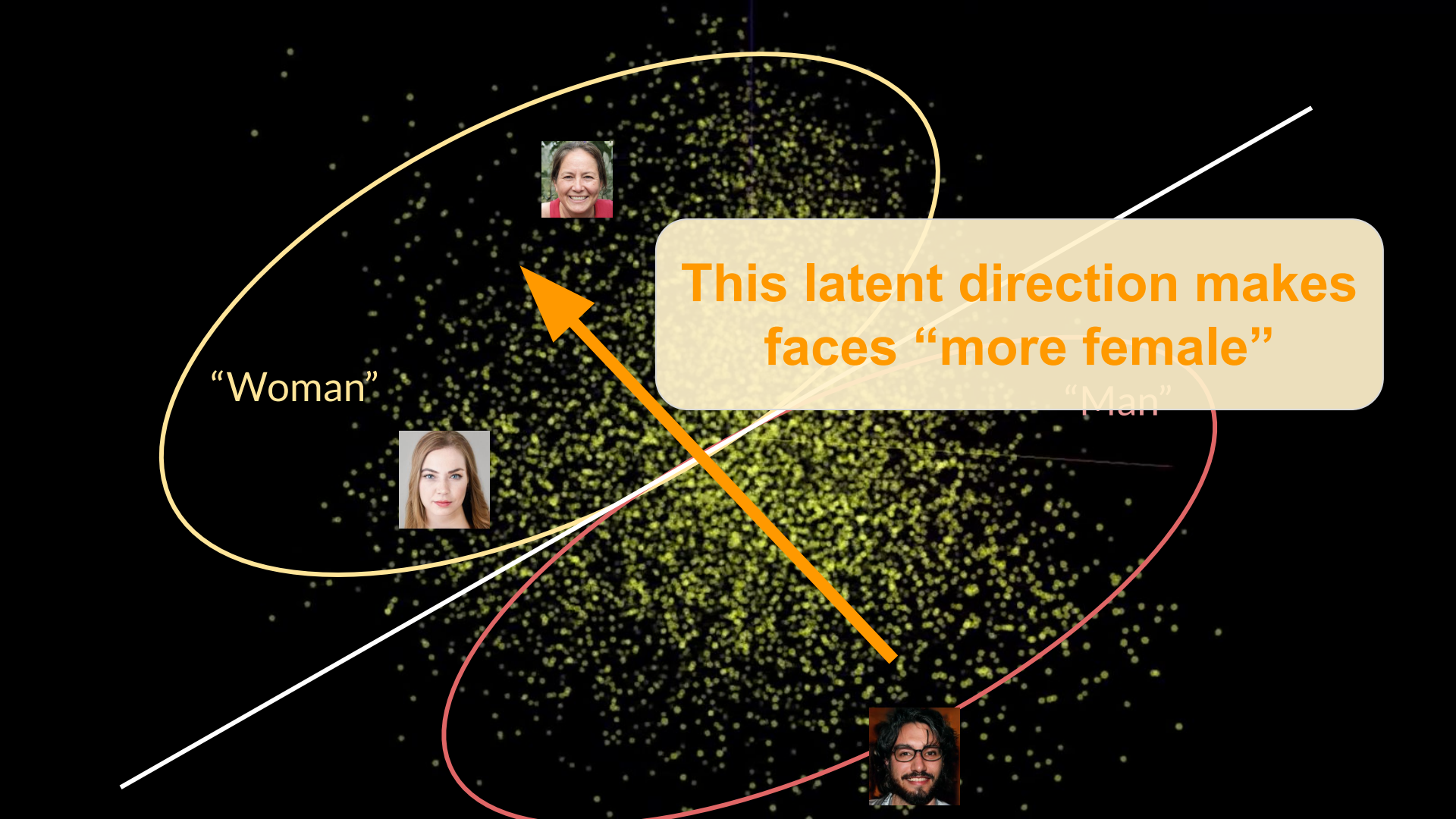

“Woman”
“Man”


“Neutral”
“Smiling”

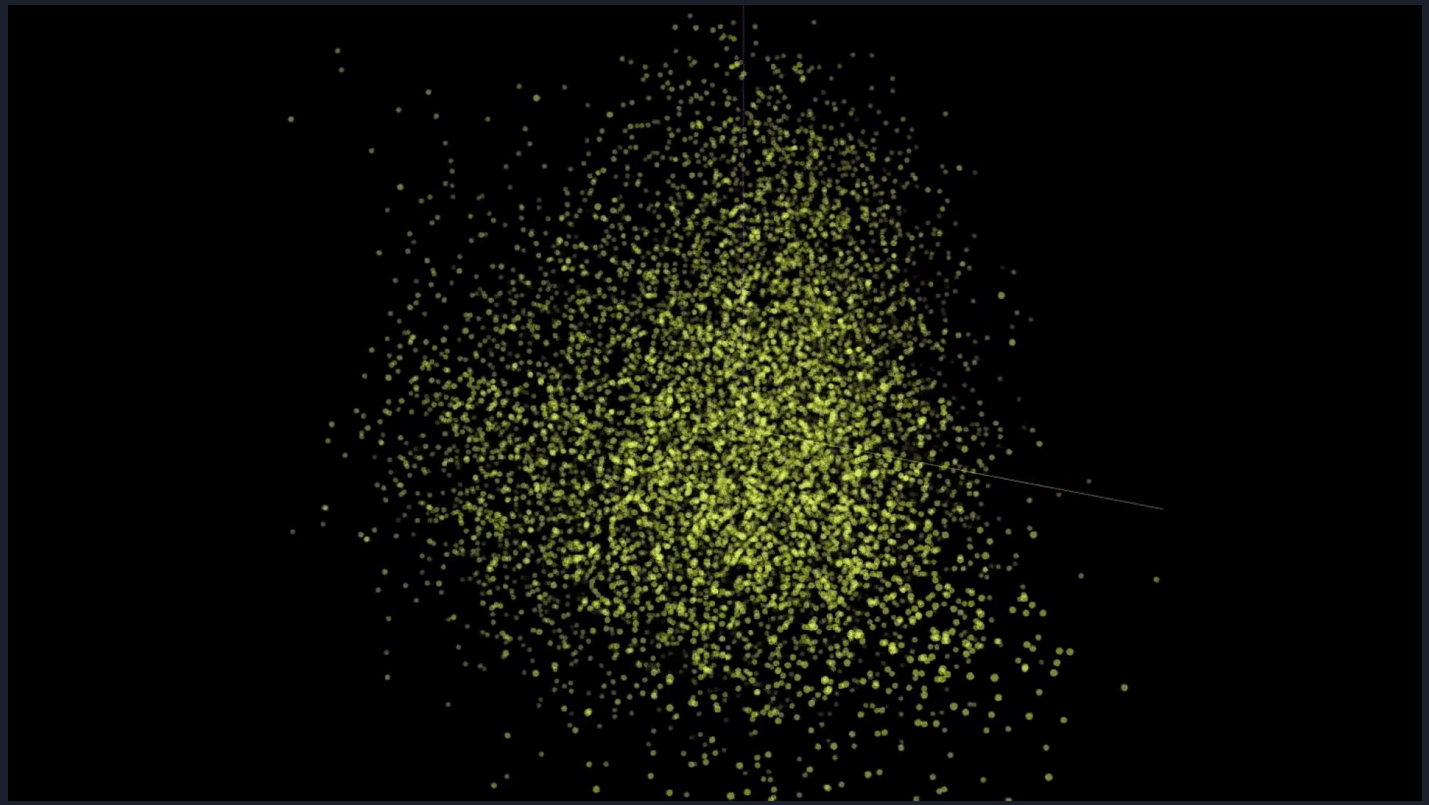

“Old”
“Young”


“Woman”
“Man”


“Woman”
“Man”


This latent direction makes faces “more female”
“Woman”
“Man”

We just found ‘meaning’ in the latent space!



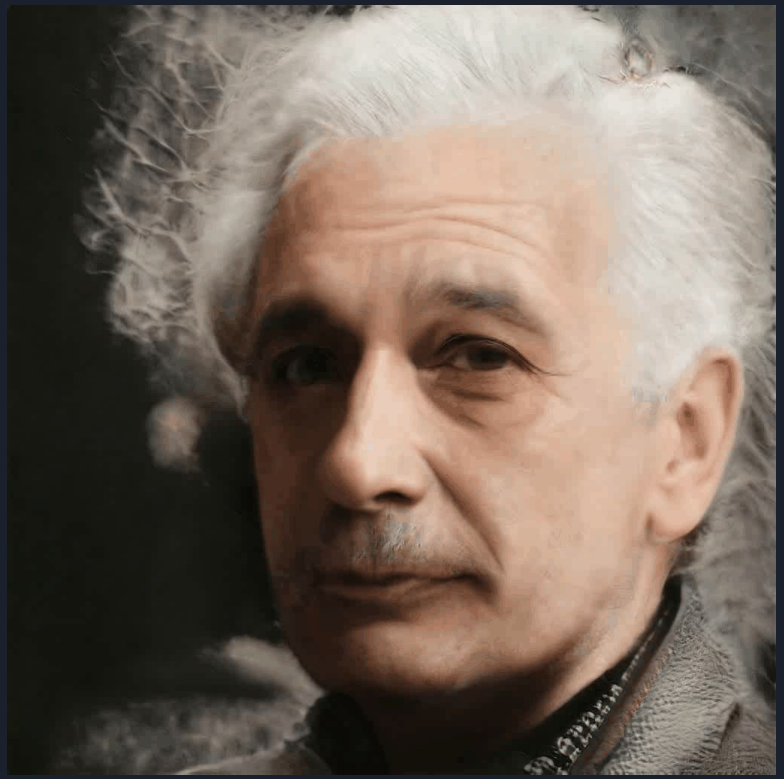
We just found ‘meaning’ in the latent space!

For any attribute we can create a dataset for,
we can now find a corresponding latent direction!

- Age_latent_direction.npy
- Gender_latent_direction.npy
- Smile_latent_direction.npy
- Pose_latent_direction.npy
- ...

“Smile”

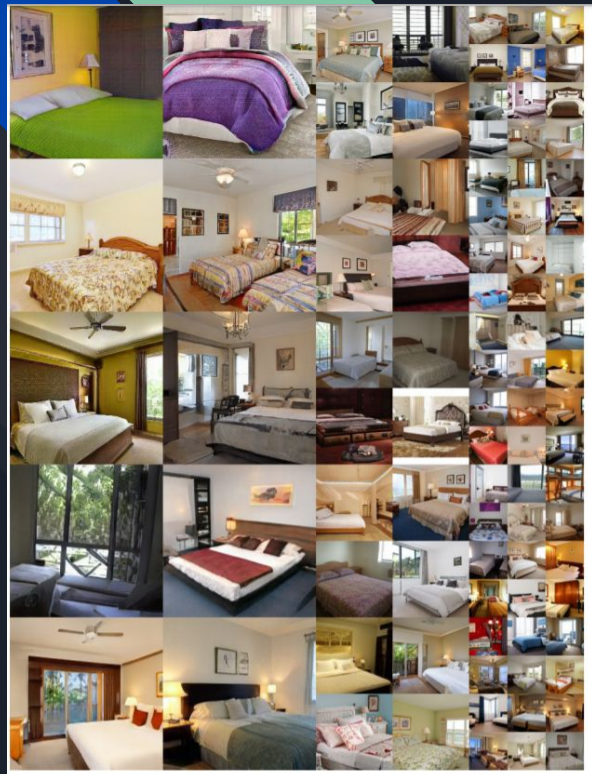
“Pose”

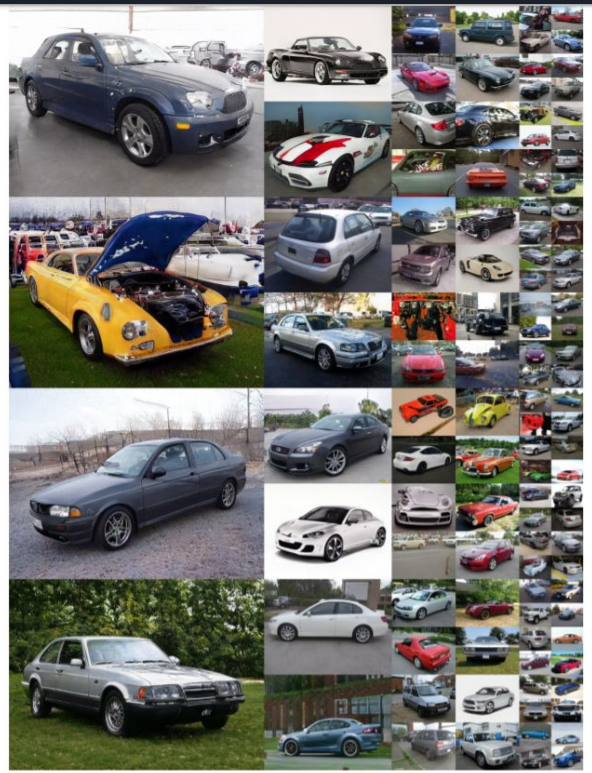


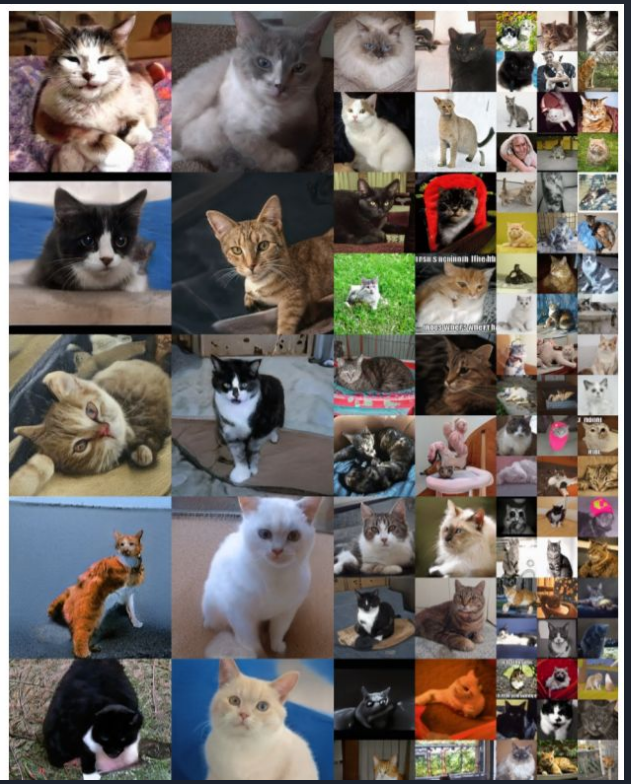
“Age”

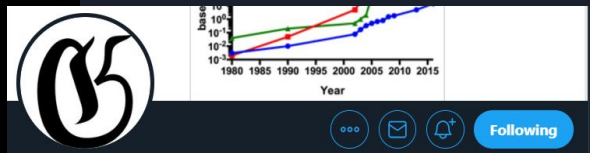
“Gender”

What’s next?









Following

Gwern
@gwern

Writer, independent researcher, Internet besserwisser. *Watashi kininarimasu*! Links: reddit.com/r/gwern/

Present day. Present time. (Ahahaha!) gwern.net Joined November 2008

Following

Alexander Reben

@artBoffin

I am an artist and roboticist. My interest is in human-machine relationships including how technology can be a lens with which to study humanity.

areben.com Joined May 2008

Edit profile

Xander Steenbrugge

@xsteenbrugge

Machine Learning researcher @ml6team & YouTuber @ 'Arxiv Insights'. Passionate about AI, Virtual Reality, renewable energy, space exploration & philosophy.

België youtube.com/arxivinsights Joined January 2015

Generating High-Resolution Fashion Model Images Wearing Custom Outfits

Gökhan Yildirim Nikolay Jetchev Roland Vollgraf Urs Bergmann
Zalando Research
{gokhan.yildirim,nikolay.jetchev,roland.vollgraf,urs.bergmann}@zalando.de

Abstract

Visualizing an outfit is an essential part of shopping for clothes. Due to the combinatorial aspect of combining fashion articles, the available images are limited to a predetermined set of outfits. In this paper, we broaden these visualizations by generating high-resolution images of fashion models wearing a custom outfit under an input body pose. We show that our approach can not only transfer the style and the pose of one generated outfit to another, but also create realistic images of human bodies and garments.

1. Introduction

Fashion e-commerce platforms simplify apparel shopping through search and personalization. A feature that can further enhance user experience is to visualize an outfit on a human body. Previous studies focus on replacing a garment on an already existing image of a fashion model [5, 2] or on generating low-resolution images from scratch by using pose and garment color as input conditions [8]. In this paper, we concentrate on generating high-resolution images wearing an outfit with a certain body pose. An outfit is composed of a set of maximum 6 articles. In order to obtain the body pose, we extract 16 keypoints using a deep pose estimator [10]. In Figure 1, we visualize a few samples from our dataset. The red markers on the fashion models represent the extracted keypoints. Both model and articles images have a resolution of 1024×768 pixels.

Figure 1: Samples from our dataset (red markers represent the extracted keypoints).

3. Experiments

The flowchart for the unconditional version of Style

Figure 3: Model images that are generated by the unconditional Style GAN.

Figure 4: Transferring the colors of an outfit or a body pose to a different generated model.

(a) Outfit #1

(b) Outfit #2

(c) Generated model images with outfit #1

(d) Generated model images with outfit #2

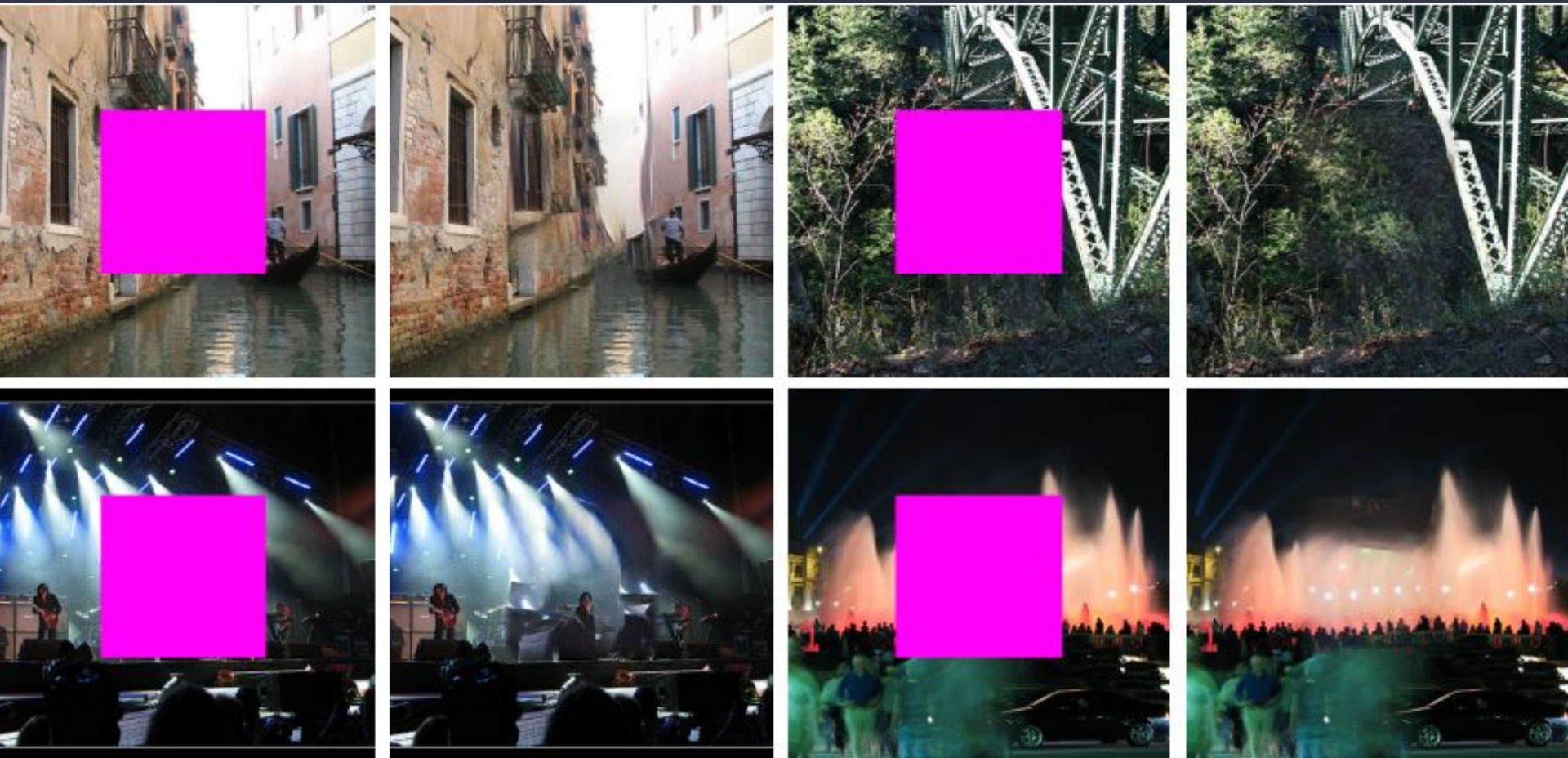
Generative 4x super resolution

original

bicubic
(21.59dB/0.6423)

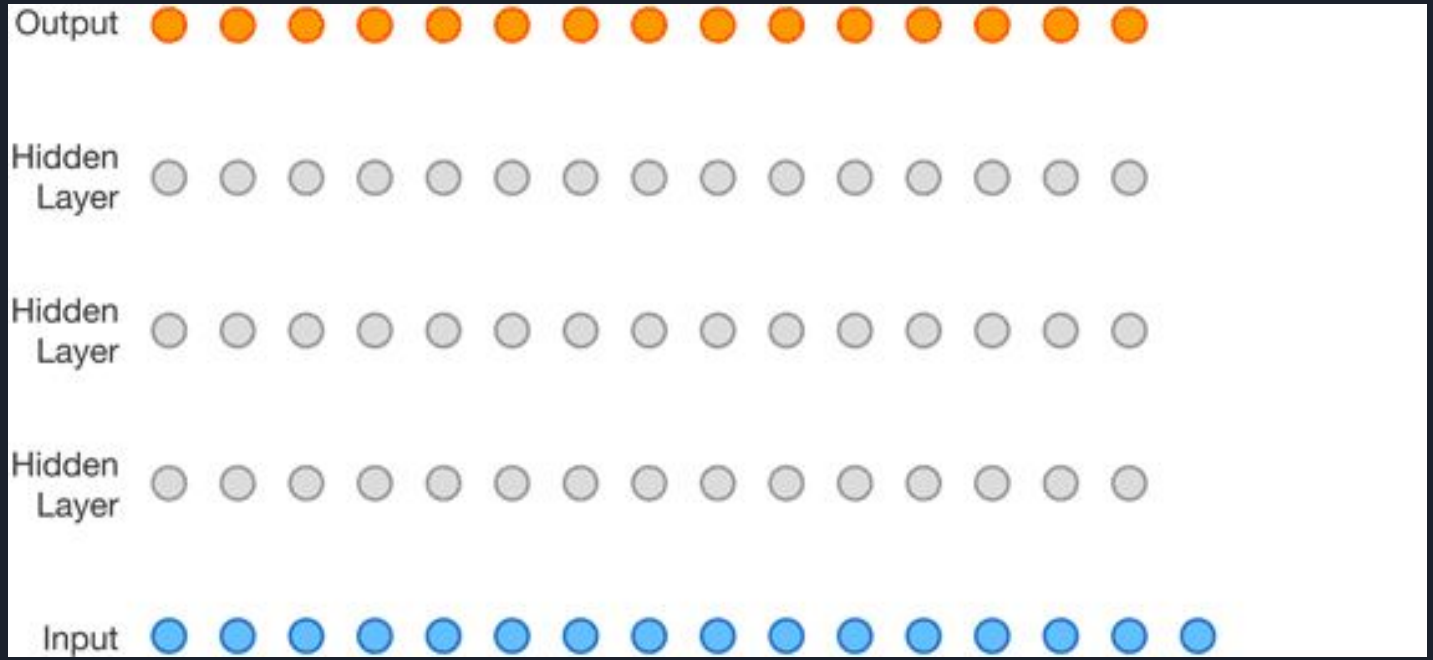
SRGAN
(20.34dB/0.6562)

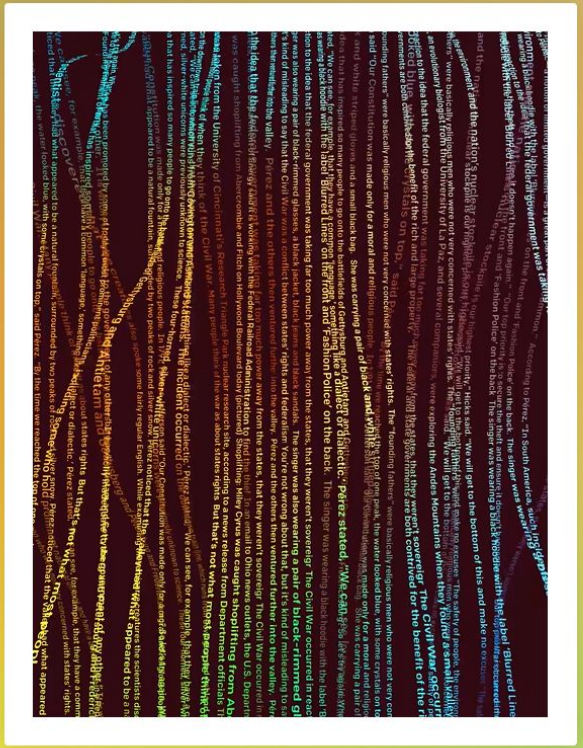
Neural inpainting → Photoshop 2.0



Sim2Real Transfer

WaveNet





Better Language Models and Their Implications

We've trained a large-scale unsupervised language model which generates coherent paragraphs of text, achieves state-of-the-art performance on many language modeling benchmarks, and performs rudimentary reading comprehension, machine translation, question answering, and summarization—all without task-specific training.

FEBRUARY 14, 2019
21 MINUTE READ

Practical Applications of generative models

- Image denoising
- Image generation (sampling)
- Image inpainting
- Attribute manipulation (VAE, VQ-VAE2)
- Learned compression (eg streaming) (GAN-based television anchor)
- Data efficiency: unsupervised pre-training
- Antibody design
- Hyperspectral sensing
- ...

- Generative models are here to stay
- Will have huge ramifications for digital media
- It’s still very early..

Questions?

Follow me: @xsteenbrugge

Subscribe: “Arxiv Insights”

YouTube

Reach out:
xander.steenbrugge@ml6.eu

Notebooks: tiny.cc/61ltbz

GIF credits: @jonathanfly

Thanks for watching!

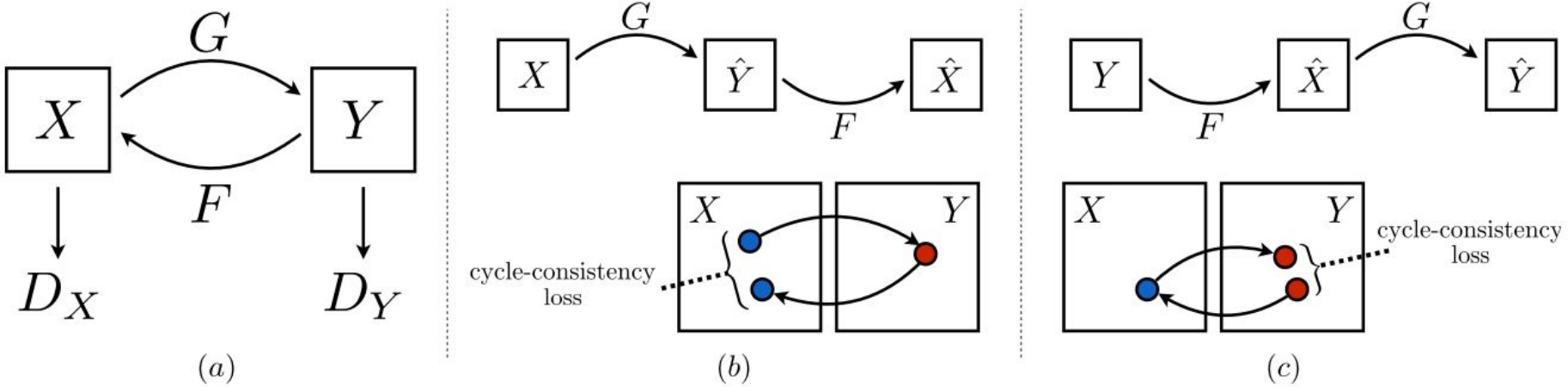
Follow me  @xsteenbrugge

Subscribe YouTube

Support me Become my patron on patreon

Cycle-GAN

Cycle-consistency loss:



Cycle-GAN

BigGan (Google)

BigGan specifics

- “We demonstrate that GANs benefit dramatically from scaling, and train models with two to four times as many parameters and eight times the batch size compared to prior art.”

- Each model is trained on 128 to 512 cores of a Google TPUv3 Pod (Google, 2018)

BigGan “Deep”

Table 9: BigGAN-deep architecture for 512×512 images.

$z \in \mathbb{R}^{128} \sim \mathcal{N}(0, I)$ Embed$(y) \in \mathbb{R}^{128}$
Linear $(128 + 128) \rightarrow 4 \times 4 \times 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock up $16ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock up $8ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock $4ch \rightarrow 4ch$
ResBlock up $4ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock up $2ch \rightarrow ch$
ResBlock $ch \rightarrow ch$
ResBlock up $ch \rightarrow ch$
BN, ReLU, 3×3 Conv $ch \rightarrow 3$
Tanh

(a) Generator

RGB image $x \in \mathbb{R}^{512 \times 512 \times 3}$
3×3 Conv $3 \rightarrow ch$
ResBlock down $ch \rightarrow ch$
ResBlock $ch \rightarrow ch$
ResBlock down $ch \rightarrow 2ch$
ResBlock $2ch \rightarrow 2ch$
ResBlock down $2ch \rightarrow 4ch$
ResBlock $4ch \rightarrow 4ch$
Non-Local Block (64×64)
ResBlock down $4ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 8ch$
ResBlock $8ch \rightarrow 8ch$
ResBlock down $8ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ResBlock down $16ch \rightarrow 16ch$
ResBlock $16ch \rightarrow 16ch$
ReLU, Global sum pooling
Embed$(y) \cdot \boldsymbol{h}$ + (linear $\rightarrow 1$)

(b) Discriminator

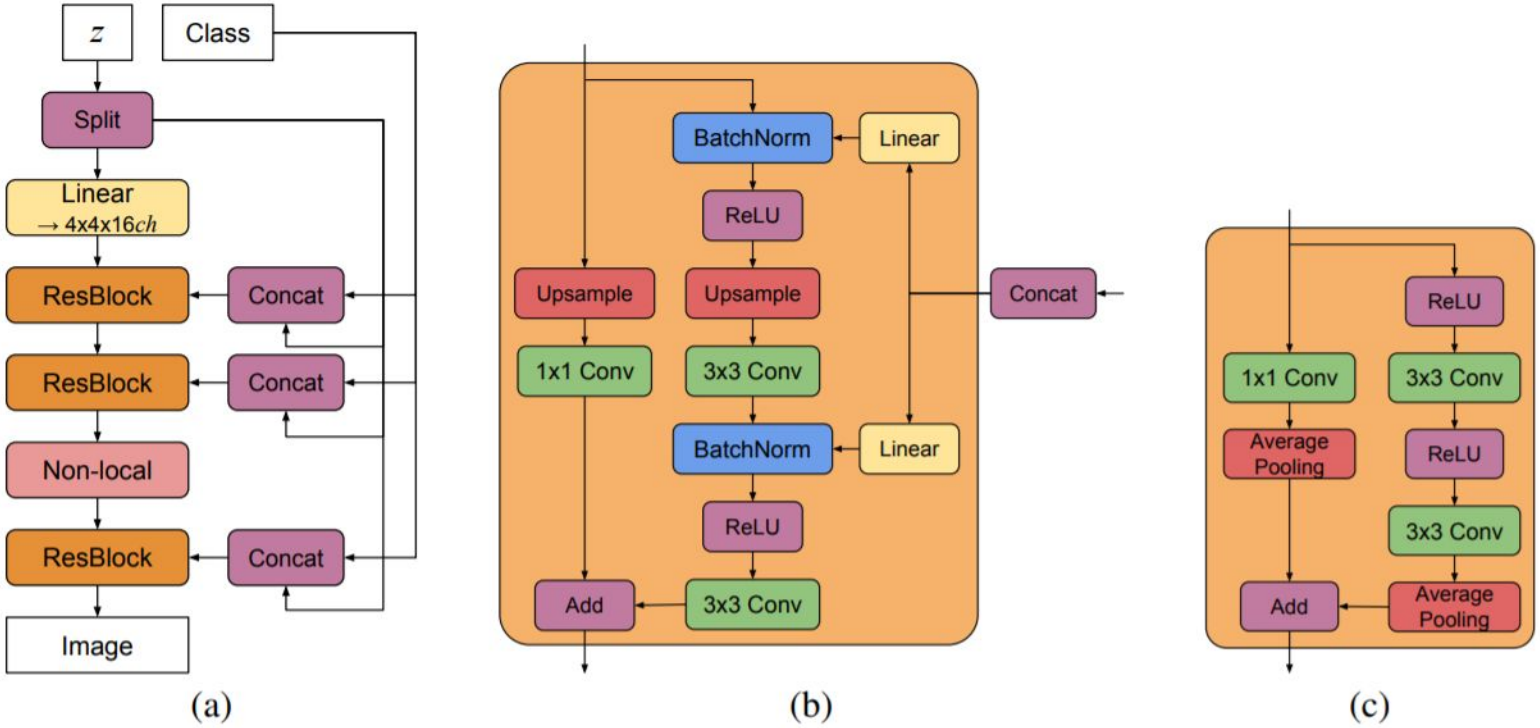


Figure 15: (a) A typical architectural layout for BigGAN's **G**; details are in the following tables. (b) A Residual Block (*ResBlock up*) in BigGAN's **G**. (c) A Residual Block (*ResBlock down*) in BigGAN's **D**.

Aren’t these models simply memorizing images?

Figure 14: Nearest neighbors in ResNet-50-avgpool (He et al., 2016) feature space. The generated image is in the top left.

Aren’t these models simply memorizing images?

Figure 12: Nearest neighbors in pixel space. The generated image is in the top left.

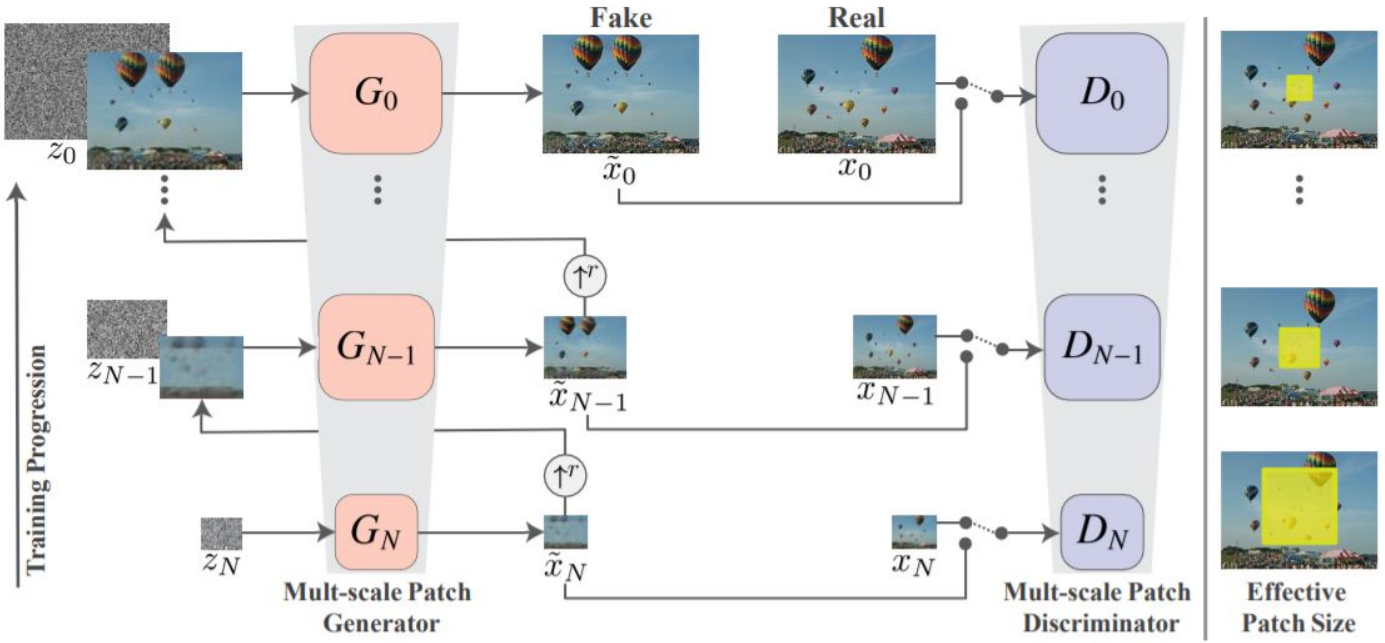


Figure 4: **SinGAN's multi-scale pipeline.** Our model consists of a pyramid of GANs, where both training and inference are done in a coarse-to-fine fashion. At each scale, G_n learns to generate image samples in which all the overlapping patches cannot be distinguished from the patches in the down-sampled training image, x_n, by the discriminator D_n; the effective patch size decreases as we go up the pyramid (marked in yellow on the original image for illustration). The input to G_n is a random noise image z_n, and the generated image from the previous scale $\tilde{x}_n$, upsampled to the current resolution (except for the coarsest level which is purely generative). The generation process at level n involves all generators $\{G_N \dots G_n\}$ and all noise maps $\{z_N, \dots, z_n\}$ up to this level. See more details at Sec. 2.

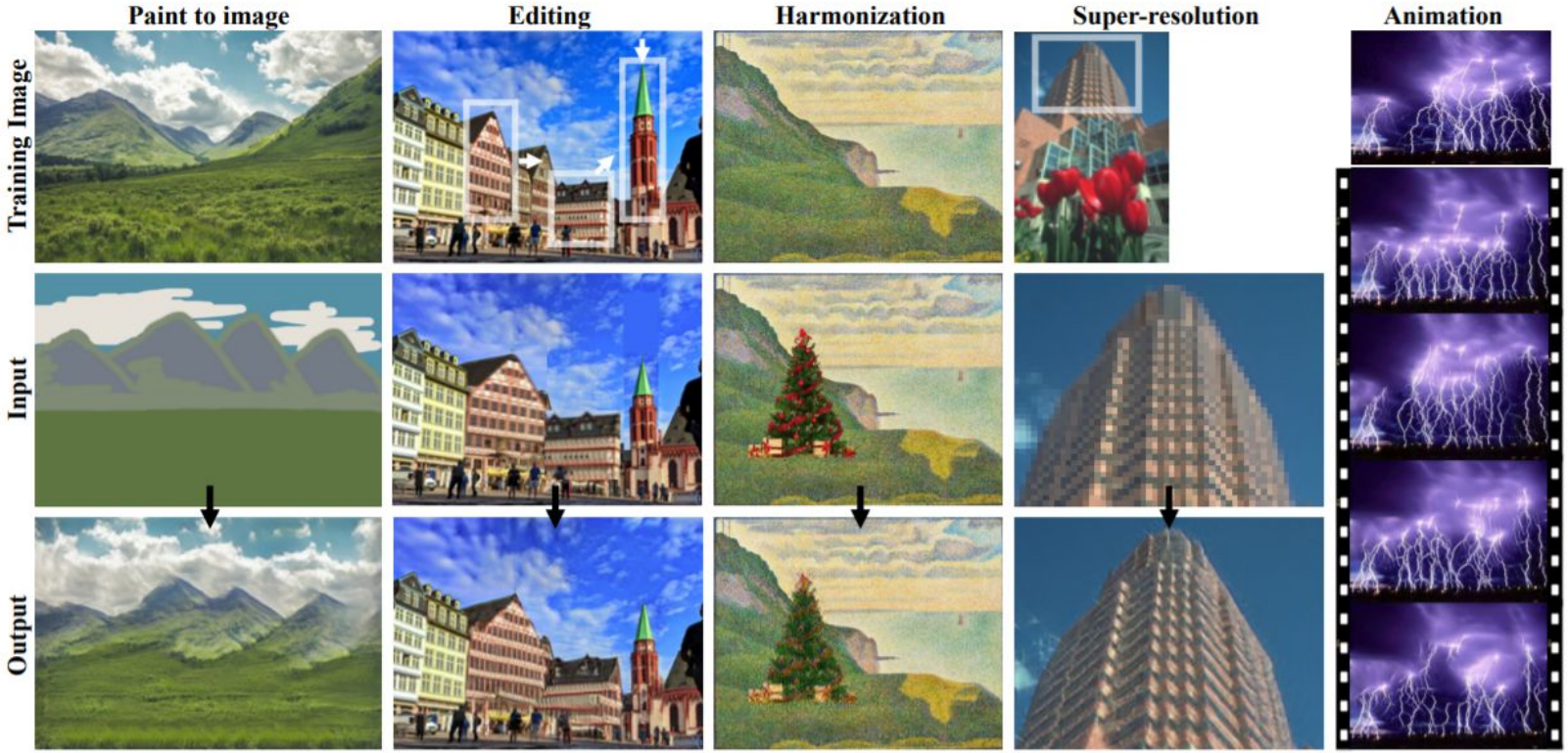


Figure 2: **Image manipulation.** SinGAN can be used in various image manipulation tasks, including: transforming a paint (clipart) into a realistic photo, rearranging and editing objects in the image, harmonizing a new object into an image, image super-resolution and creating an animation from a single input. In all these cases, our model observes only the training image (first row) and is trained in the same manner for all applications, with no architectural changes or further tuning (see Sec. 4).

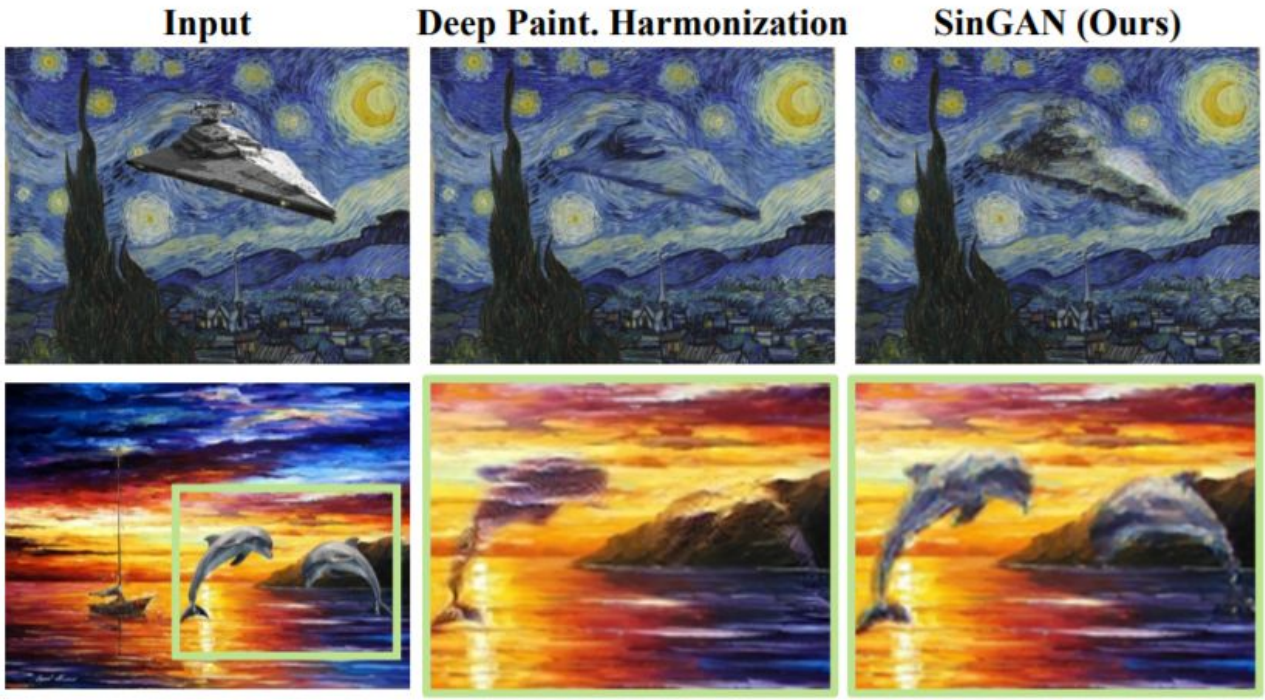


Figure 13: **Harmonization.** Our model is able to preserve the structure of the pasted object, while adjusting its appearance and texture. The dedicated harmonization method [34] overly blends the object with the background.

Image Super Resolution

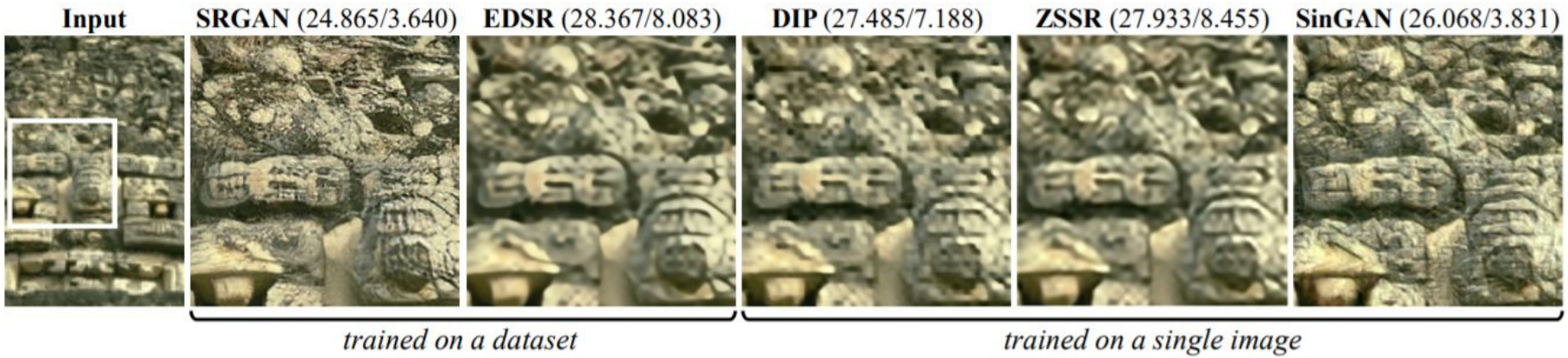
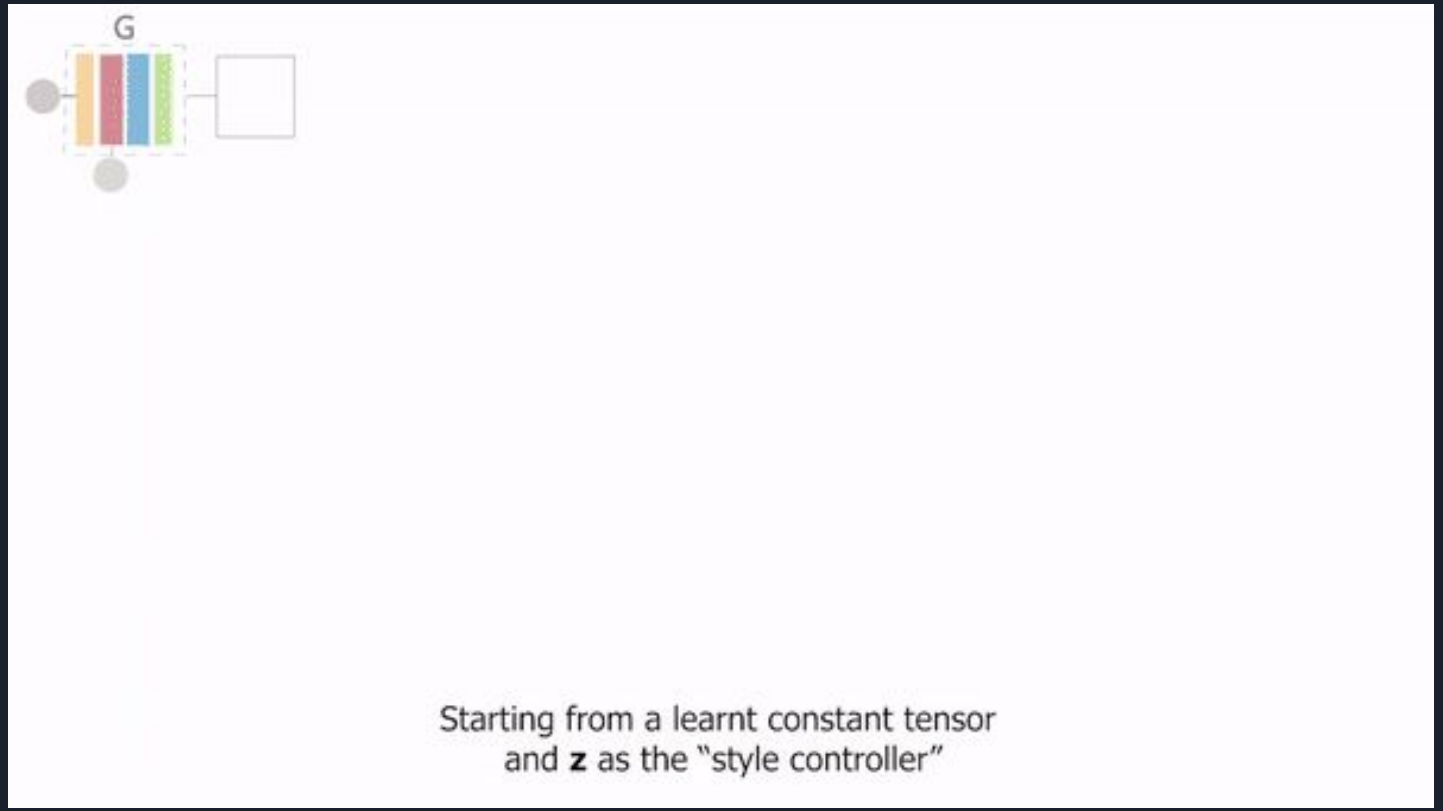


Figure 10: **Super-Resolution.** When SinGAN is trained on a low resolution image, we are able to super resolve. This is done by iteratively upsampling the image and feeding it to SinGAN's finest scale generator. As can be seen, SinGAN's visual quality is better than the SOTA internal methods ZSSR [45] and DIP [49]. It is also better than EDSR [32] and comparable to SRGAN [30], external methods trained on large collections. Corresponding PSNR and NIQE [40] are shown in parentheses.

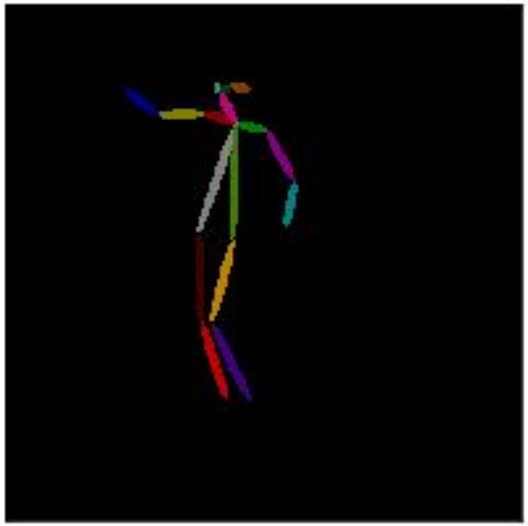
Semantic manipulation: HoloGAN

Semantic manipulation: HoloGAN



Semantic manipulation: HoloGAN

Pose-to-Body Results



Latent space interpolations

INFO GAN

FaceApp

BigGAN (DeepMind, Oktober 2018)

BigGAN (DeepMind, Oktober 2018)

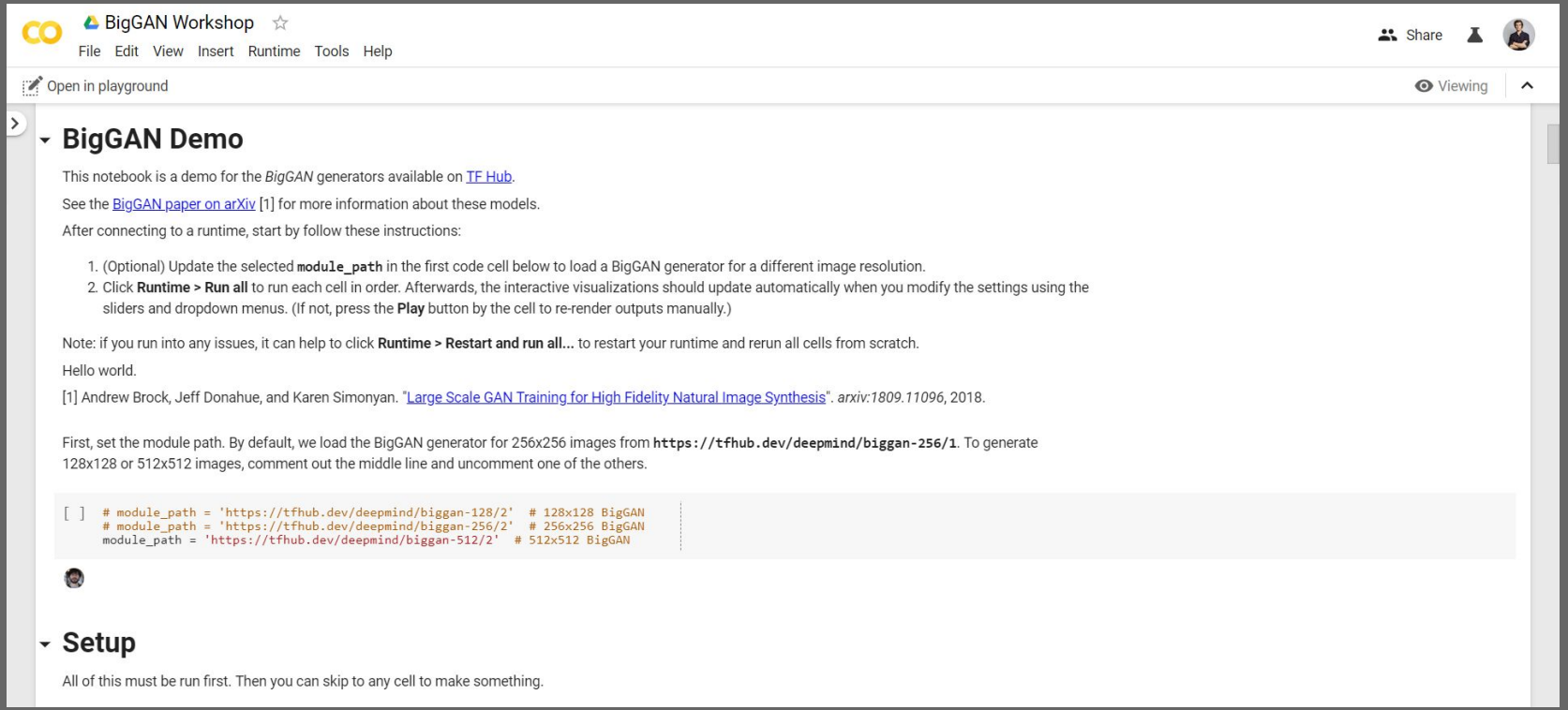
BigGAN (DeepMind, Oktober 2018)

BigGAN interpolations



BigGAN - Interpolation LIVE DEMO

Link: https://colab.research.google.com/drive/1rqDwIddy0eunhhV8yrznG4SNiB5XWFJJ



This “simple” idea…

is creating a totally new ‘generative’ industry

Practical Applications:

- Image denoising
- Image generation (sampling)
- Image inpainting
- Attribute manipulation (VAE, VQ-VAE2)
- Learned compression (eg streaming) (GAN-based television anchor)
- Data efficiency: unsupervised pre-training
- Antibody design
- Hyperspectral sensing
- ...

More efficient Image Classifiers through GANs:

- https://arxiv.org/abs/1907.07484 (related paper)
- http://gradientscience.org/adv/ (Blog)

- First there was styletransfer for creating fun images & art
- Now, styletransfer is used to create more robust & generalizing classifiers!

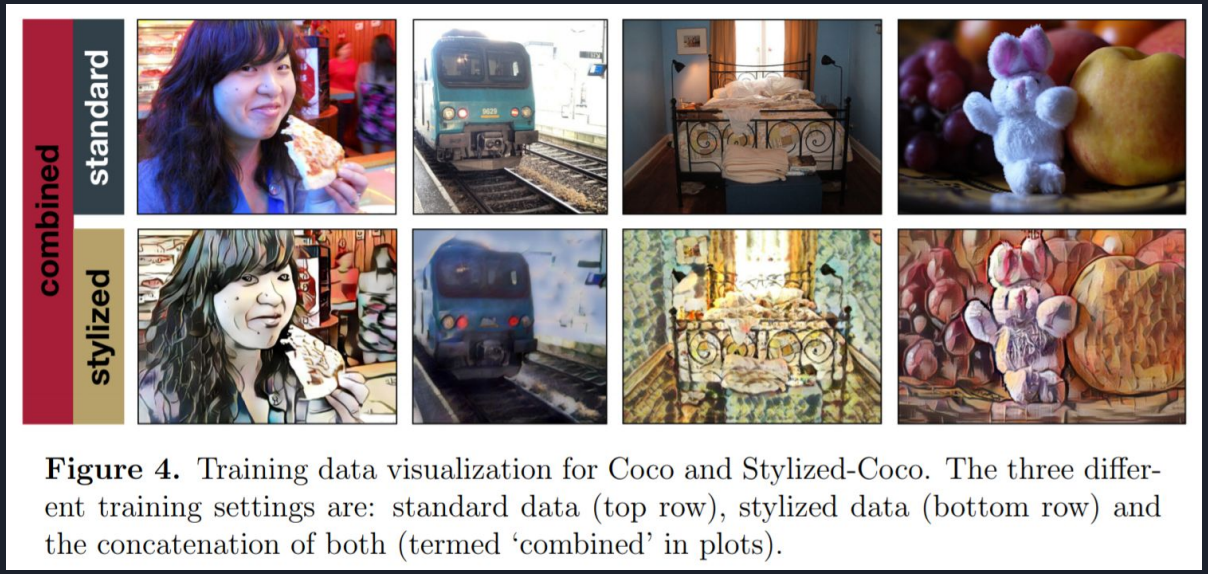
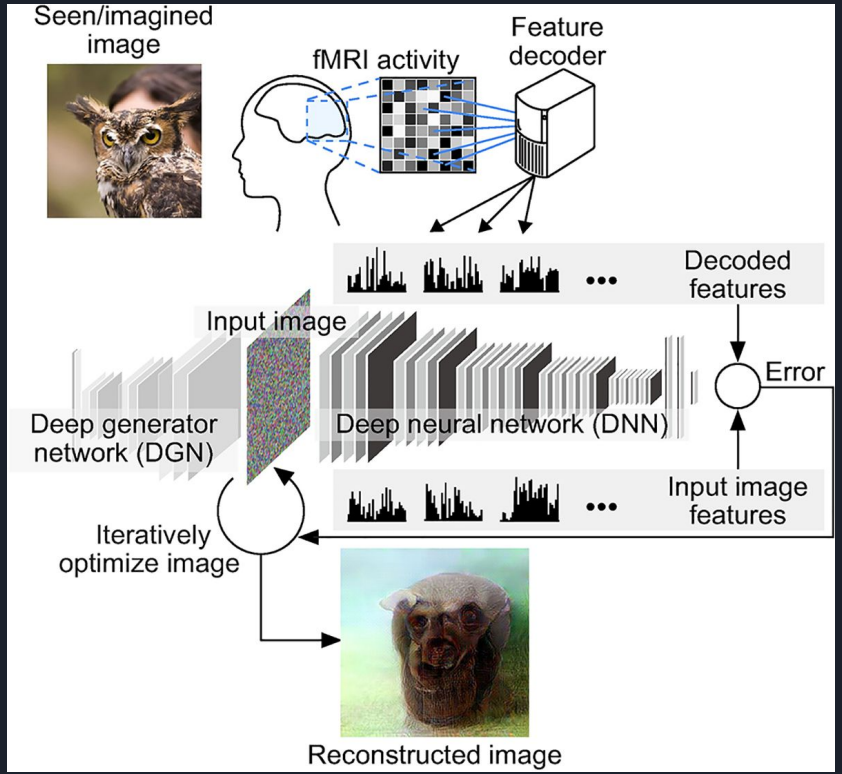


Figure 4. Training data visualization for Coco and Stylized-Coco. The three different training settings are: standard data (top row), stylized data (bottom row) and the concatenation of both (termed 'combined' in plots).

Deep image reconstruction from human brain activity

Link: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006633

Pdf: https://journals.plos.org/ploscompbiol/article/file?id=10.1371/journal.pcbi.1006633&type=printable



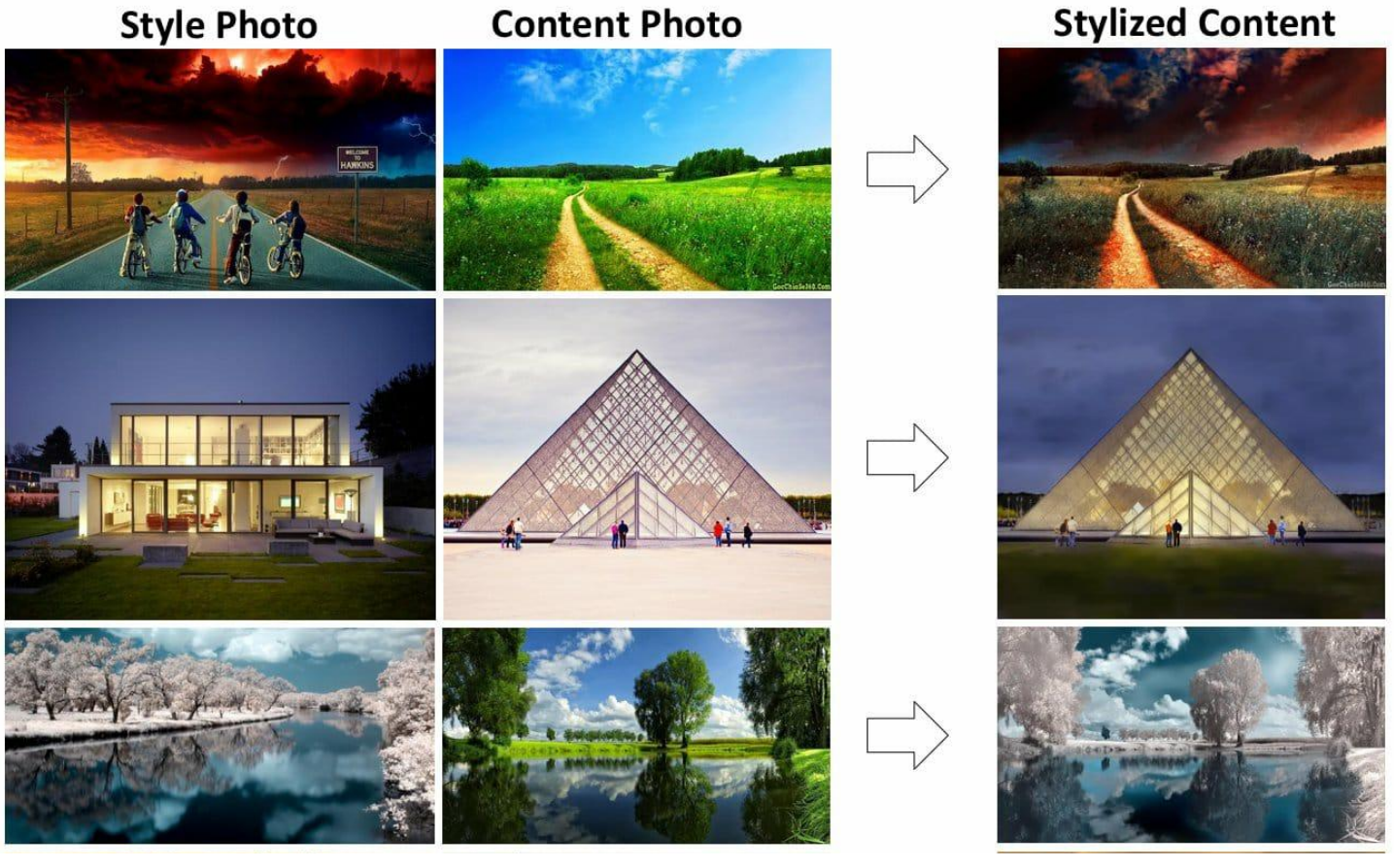
Generative 4x super resolution

original

bicubic
(21.59dB/0.6423)

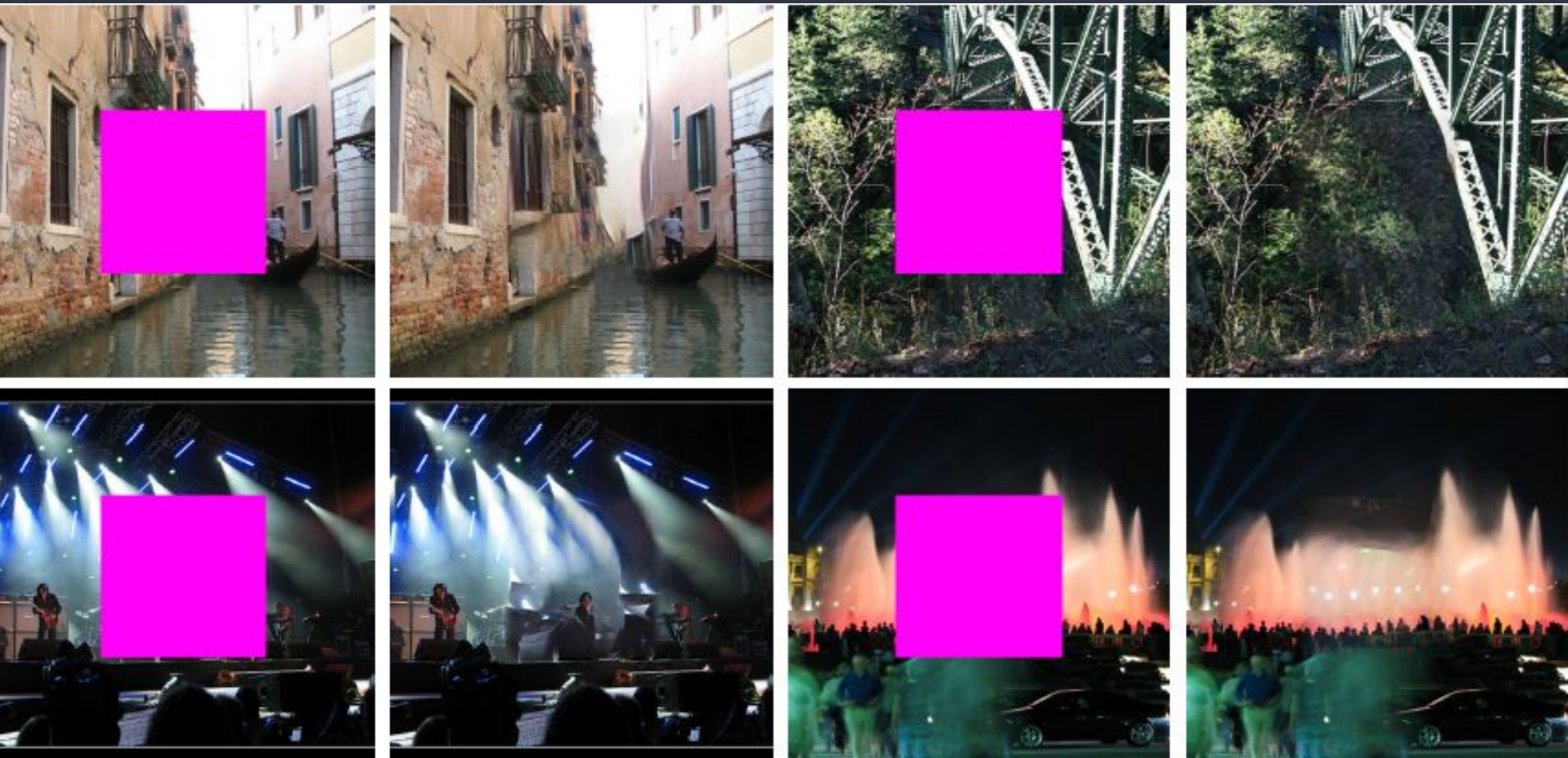
SRGAN
(20.34dB/0.6562)

Photo style transfer



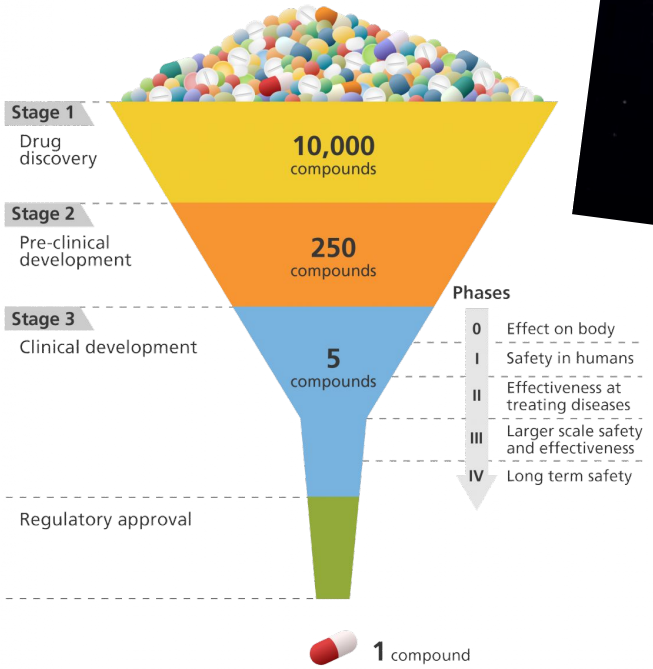
Sketch to image

Neural inpainting → Photoshop 2.0



Sim2Real Transfer for testing self-driving cars

Huge application potential...



Many emerging applications

- Antibody design for treating incurable diseases
- Discovering new antibiotics
- Protein folding
- Bank fraud
- Hyperspectral sensing in space applications
- Data augmentation
- Image denoising
- Data compression
- ...

Startpagina Over

Heb je al een account? Inloggen

Silviu Pirvu
@UrbanPlanningAI

Digital Urbanism and AI for Cities on Earth and Beyond. #PlanTech and Music Production when rainy.

London, England

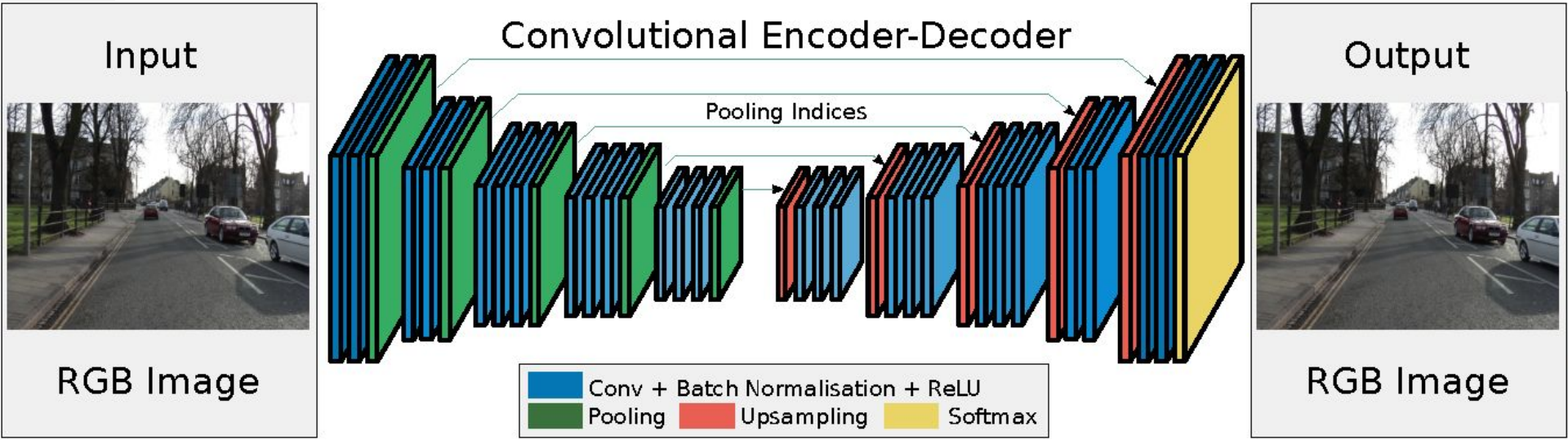
Silviu Pirvu
@UrbanPlanningAI

Volgen

Als antwoord op @DeepMindAI

For an architect or designer, this has massive implications in generating stylistic transformations, as well as diversified combinations of materials and designs. I'm intrigued to explore how #BigGAN can be combined with pix2pix and #cycleGAN to generate controlled designs.

Deep Image Prior



‘Blind’ denoising

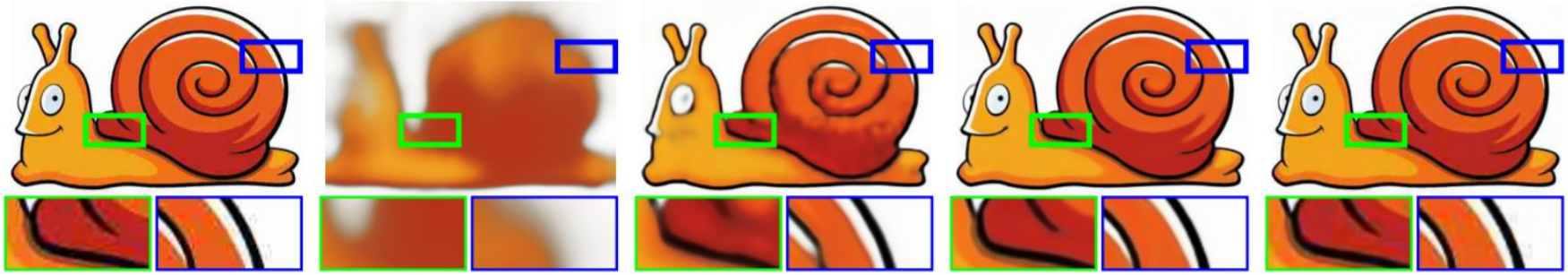
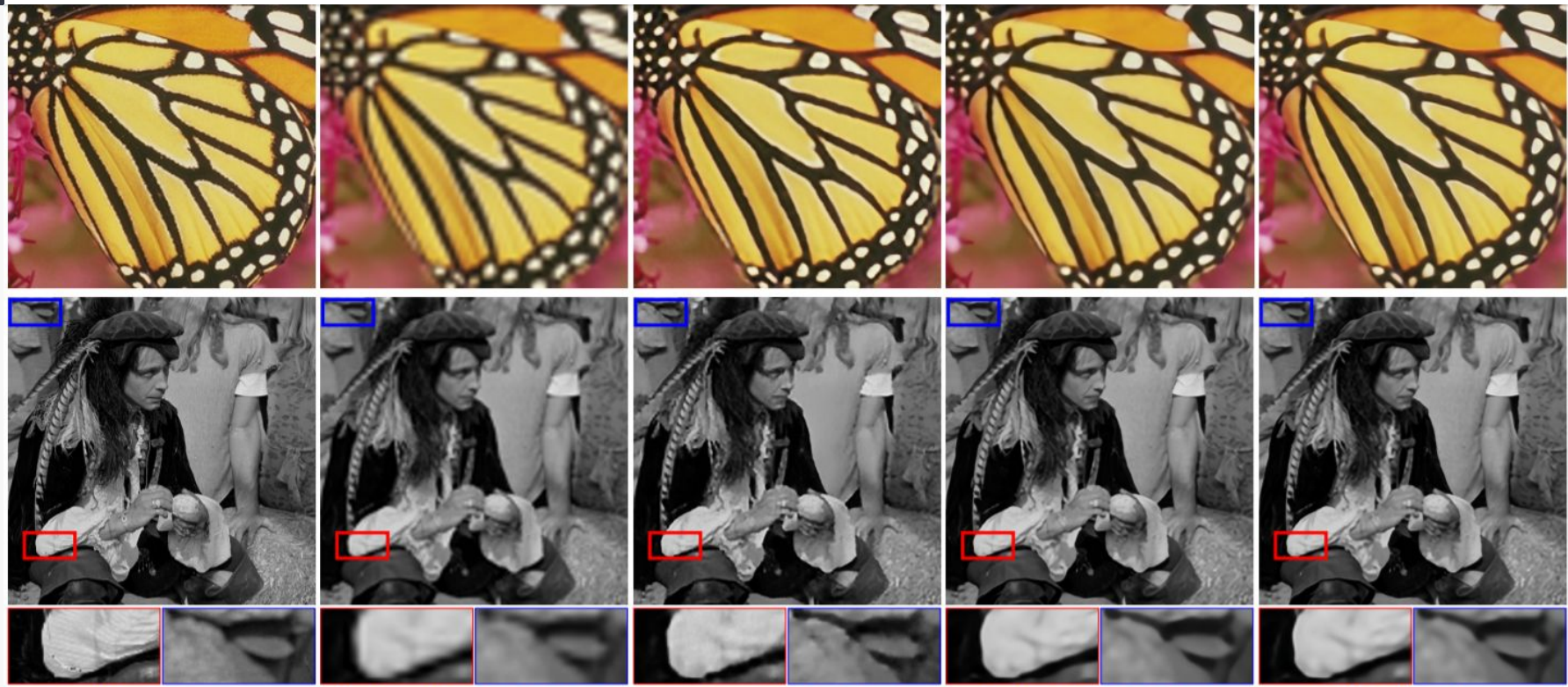


Figure 3: **Blind restoration of a JPEG-compressed image.** (*electronic zoom-in recommended*) Our approach can restore an image with a complex degradation (JPEG compression in this case). As the optimization process progresses, the deep image prior allows to recover most of the signal while getting rid of halos and blockiness (after 2400 iterations) before eventually overfitting to the input (at 50K iterations).

Super resolution



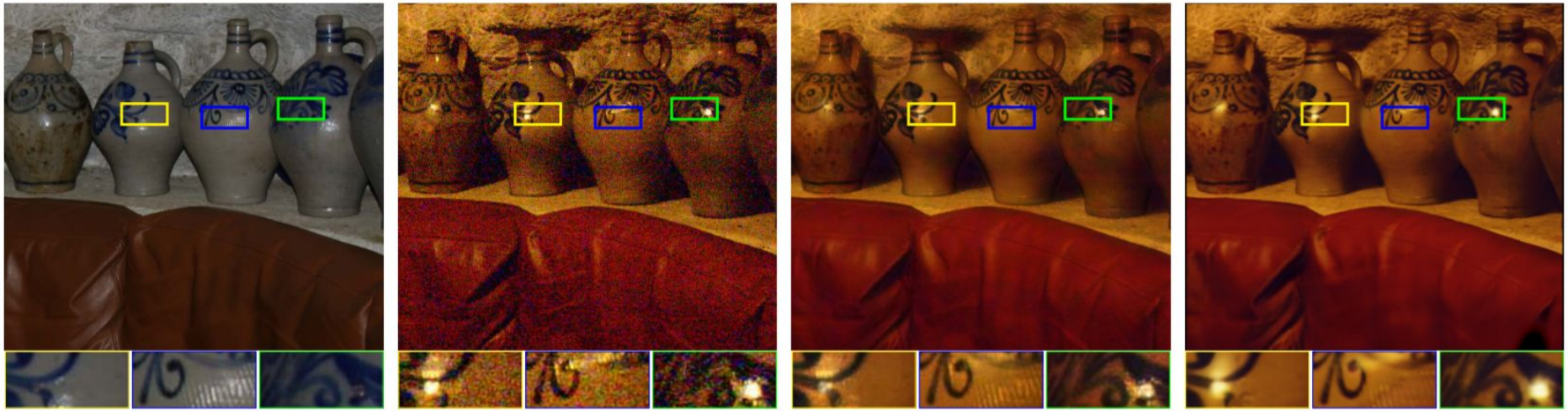
(a) Original image (b) Bicubic, **Not trained** (c) Ours, **Not trained** (d) LapSRN, **Trained** (e) SRResNet, **Trained**

Inpainting

(a) Input (white=masked)

(b) Encoder-decoder, depth=6

Flash-no flash restoration



(a) Flash (b) No flash (c) Joint bilateral [25] (d) Deep image prior

Figure 10: **Reconstruction based on flash and no-flash image pair.** The deep image prior allows to obtain low-noise reconstruction with the lighting very close to the no-flash image. It is more successful at avoiding "leaks" of the lighting patterns from the flash pair than joint bilateral filtering [25] (c.f. blue inset).

GAN’s and generative ‘Art’

Ganbreeder

dolores a month ago

Children Crossbreed Edit-Genes

Make Children

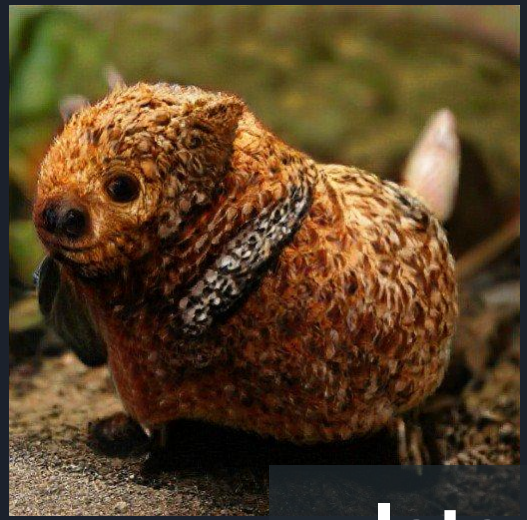
< Similar Different >

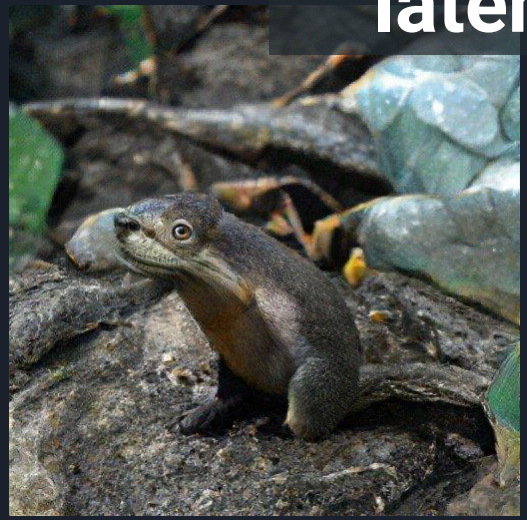
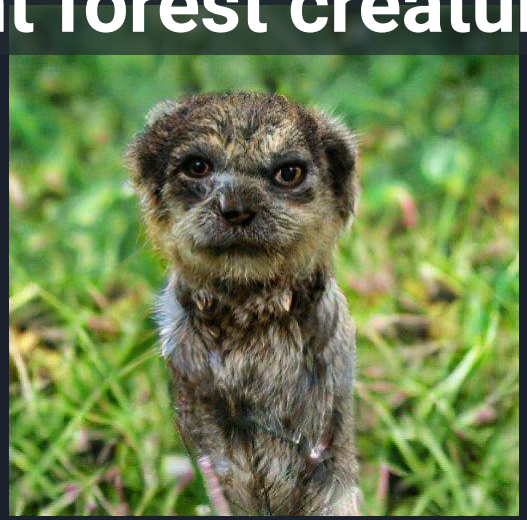
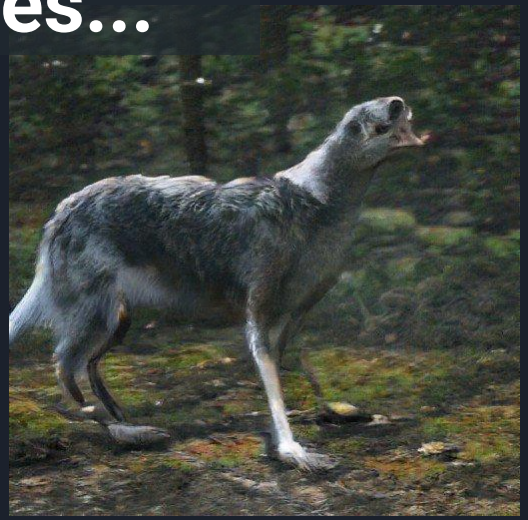


‘Latent animals’

“Firefox”

“Flowerdog”



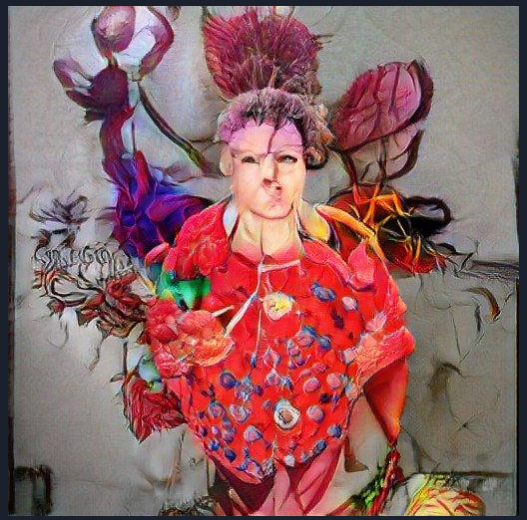
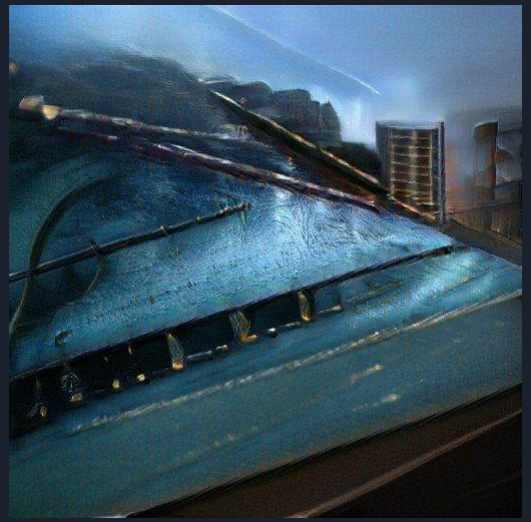
latent forest creatures...



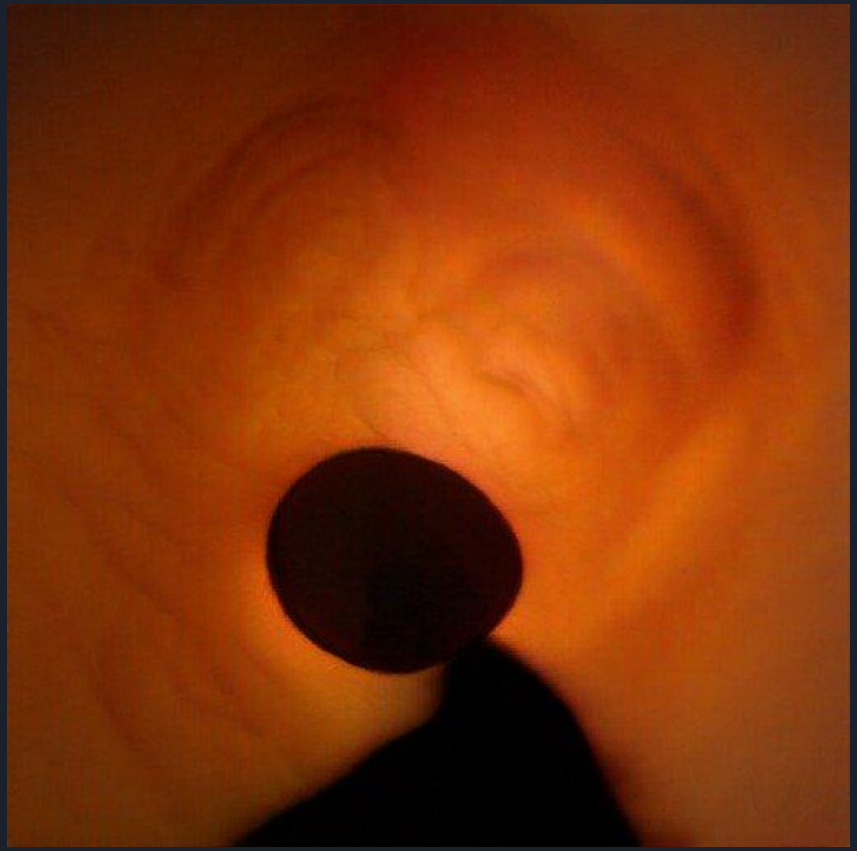







Generated artworks



Generated artworks

Generated artworks

What is Creativity?

"Building universes out of nothing."

Danny Sullivan

"Copying smarter."

Lisa Barone

"Giving the world something it didn't know it was missing."

Daniel Pink

"Seeing something that doesn't exist and then making it so."

Hugh Howey

"Going to unexpected places."

Shane Snow

"Seeing the intersection of seemingly unrelated topics and combining them into something new."

Brian Clark

"Tapping into your soul and your intuition and allowing them to guide what you make."

Bernadette Jiwa

"It's our brains doing what they do."

Michael Grybko

"Interpreting something you saw or experienced and processing it so it comes out different than how it went in."

Henry Rollins

"One part inspiration, one part motivation."

Ann Handley

"Living in possibility and abundance rather than limitation and scarcity."

CJ Lyons

TECH | ARTIFICIAL INTELLIGENCE | CULTURE

Christie's sells its first AI portrait for $432,500, beating estimates of $10,000

8

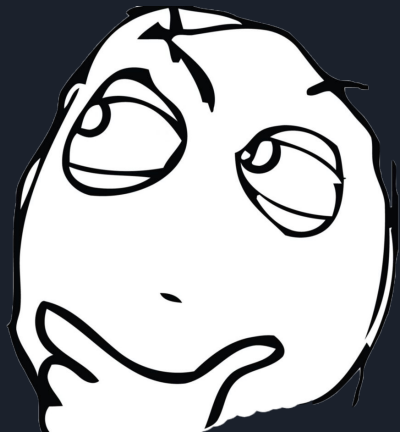
The image was created using a machine learning algorithm that scanned historical artwork

By James Vincent | Oct 25, 2018, 1:03pm EDT

SHARE

Who ‘owns’ it?

- The researchers who open-sourced the **algorithm**?
- The people who created the **dataset**?
- The company who **trained** the model?
- The person who ‘**sampled**’ the image?
- ...



Personal projects with GANs (+ live demo's)

- Music visualization through GANs:
- Google Earth GAN: