# A Gentle Introduction to Deep Generative Models

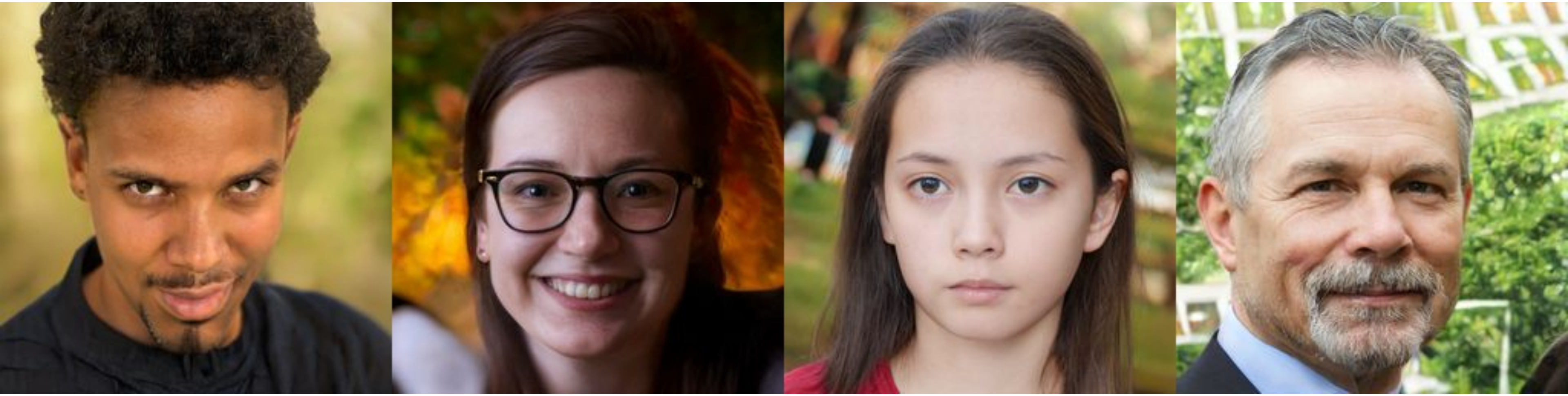
Mustafa Mustafa
Berkeley Lab

AI for Earth Systems Science (AI4ESS)
Summer School 2020

## Outline

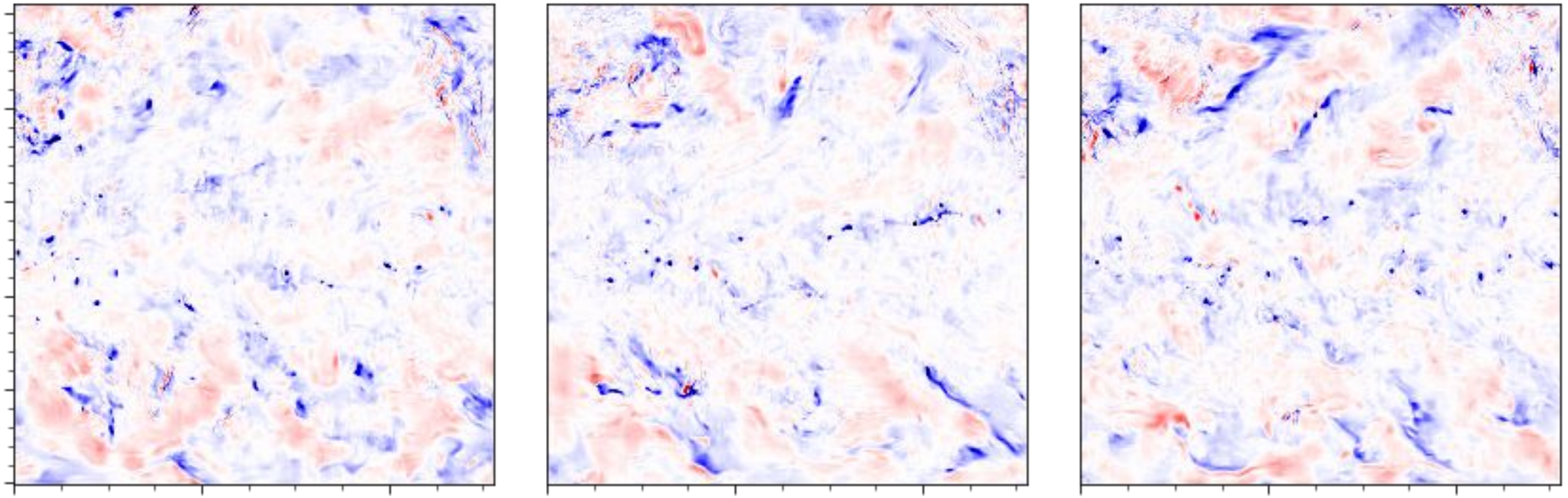
- Intro
- What are generative models?
- Latent variable models
- Variational AutoEncoders
- Generative Adversarial Networks

## Which of these faces is fake?



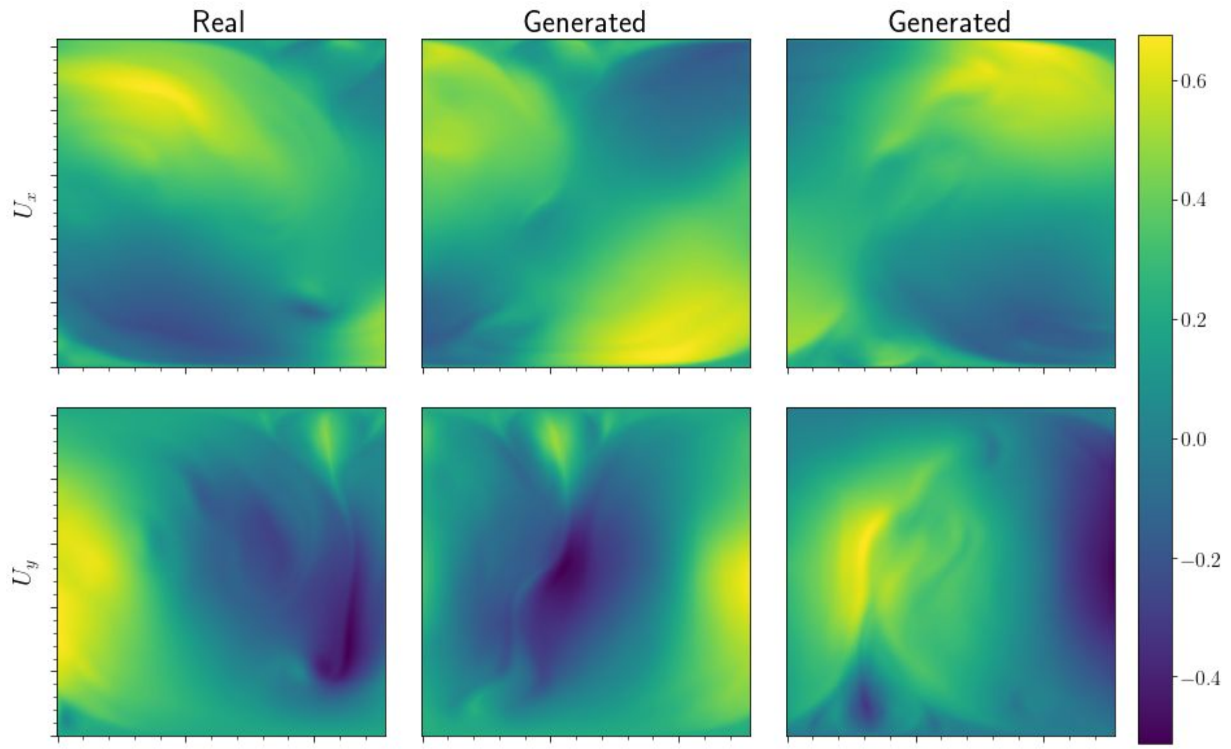
# Which of these simulations is fake?

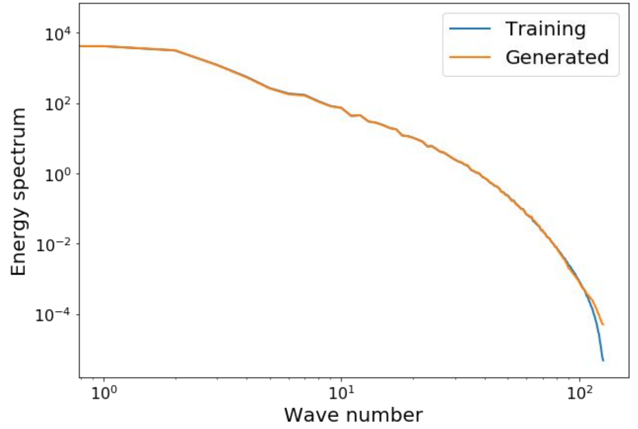
R. Gupta, M. Mustafa, K. Kashinath, manuscript to be submitted soon.



Atmospheric upwind velocity (ω500) fields over the pacific

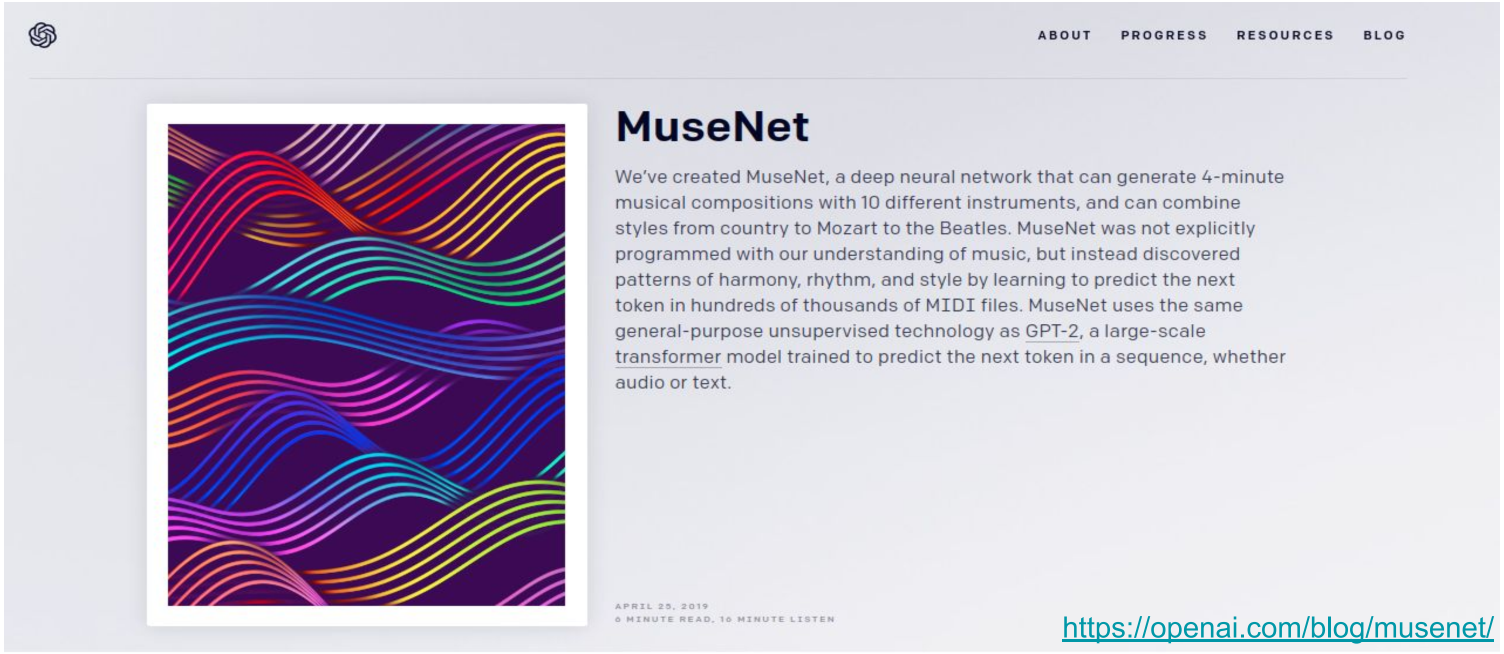
# It is not just pretty pictures (Rayleigh–Bénard convection)





R. Gupta, M. Mustafa, K. Kashinath, manuscript to be submitted soon.

# Generating music



ABOUT PROGRESS RESOURCES BLOG

## MuseNet

We've created MuseNet, a deep neural network that can generate 4-minute musical compositions with 10 different instruments, and can combine styles from country to Mozart to the Beatles. MuseNet was not explicitly programmed with our understanding of music, but instead discovered patterns of harmony, rhythm, and style by learning to predict the next token in hundreds of thousands of MIDI files. MuseNet uses the same general-purpose unsupervised technology as GPT-2, a large-scale transformer model trained to predict the next token in a sequence, whether audio or text.

APRIL 25, 2019
6 MINUTE READ, 16 MINUTE LISTEN

https://openai.com/blog/musenet/

## Samples

Prompt: First 5 notes of Chopin Op. 10, No. 9 SOUNDCLOUD

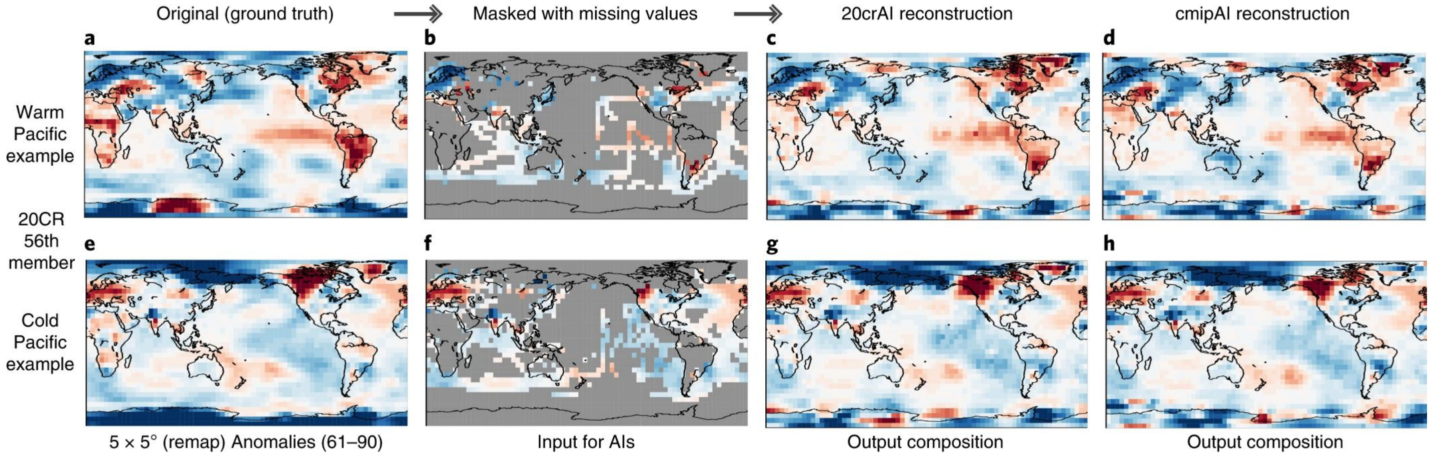
Prompt: Jazz Piano-Bass-Drums SOUNDCLOUD

# Denoising/ Image inpainting

nvidia.com/research/inpainting/

Generative models can be used to denoise or inpaint missing data.

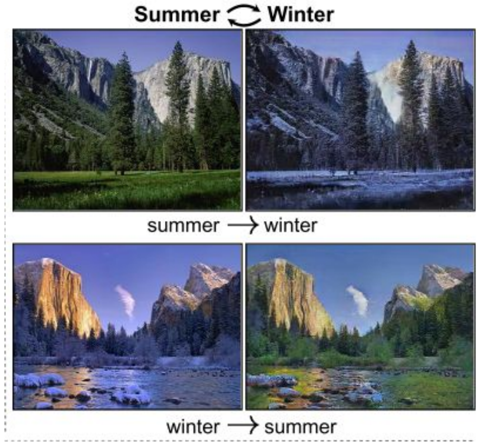
Generative models used to reconstruct observational gaps in historical temperature measurements.



"Artificial intelligence reconstructs missing climate information" , C. Kadow et al, Nature Geoscience, 13,408–413(2020)

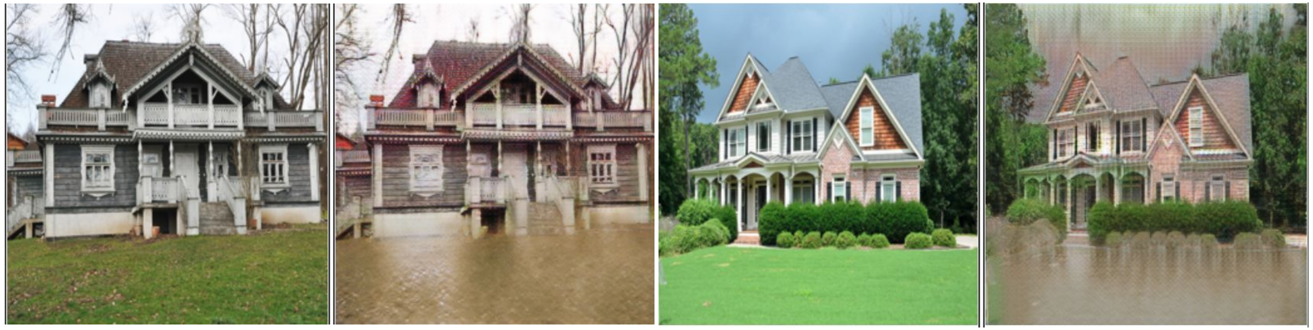
# Domain-to-domain translation

CycleGAN, Zhu et al, arXiv:1703.10593



Generative models can be used to “translate” data from domain-to-another.

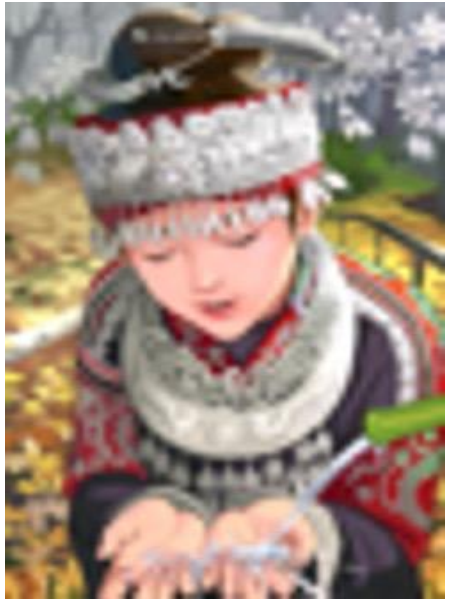
Schmidt et al. arXiv:1905.03709

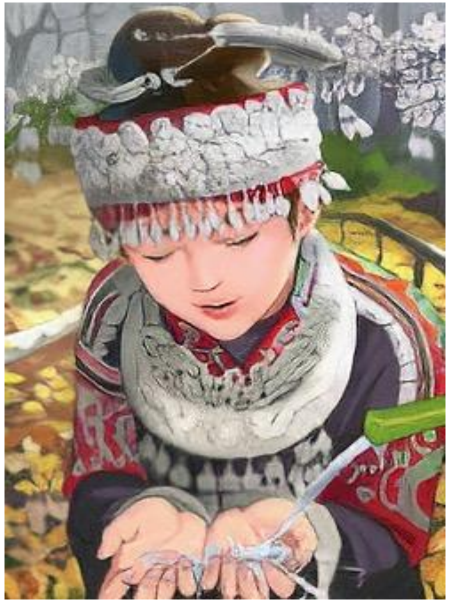


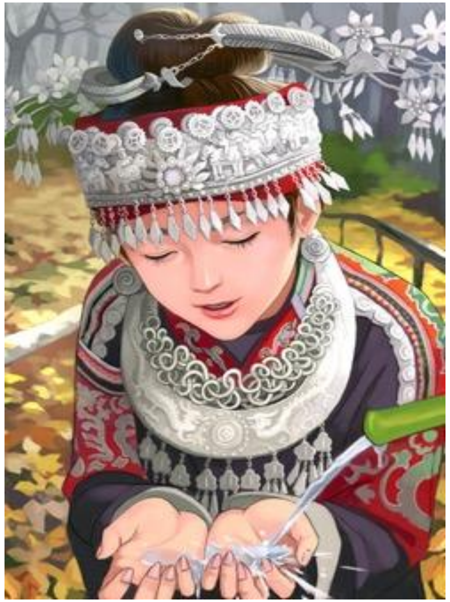
“Visualizing the Consequences of Climate Change Using Cycle-Consistent Adversarial Networks”

# Super-resolution

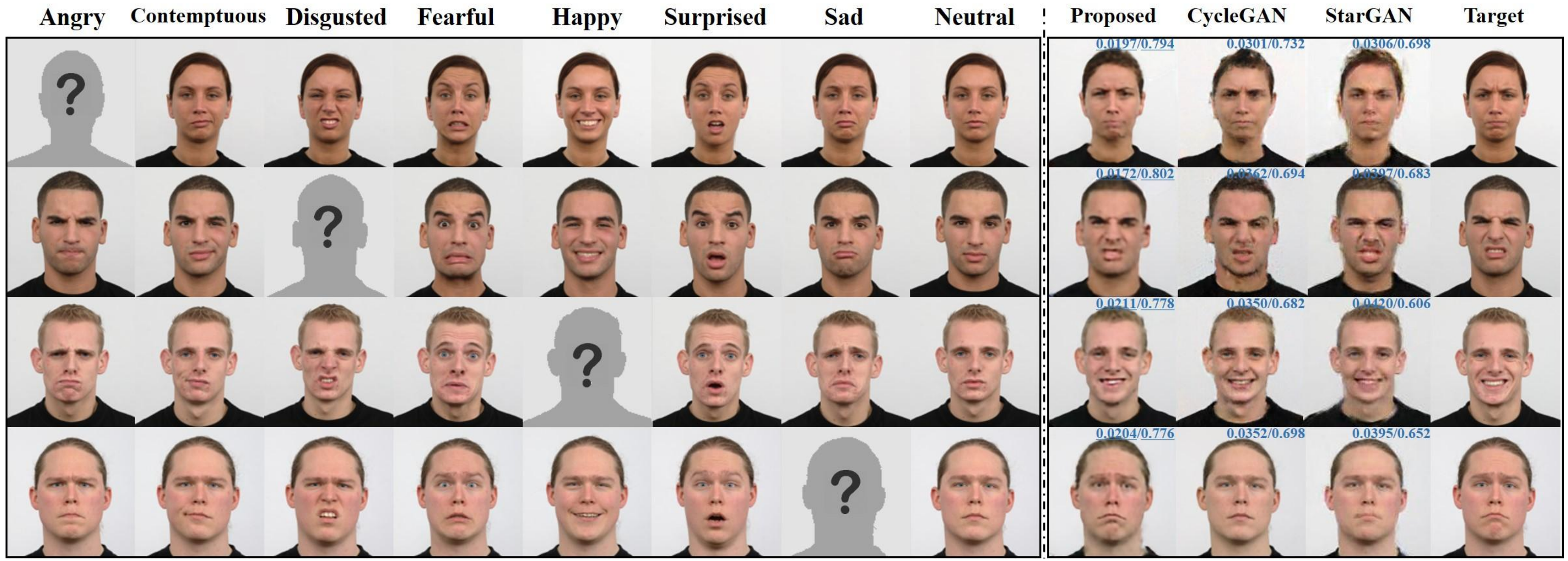
SRGAN, Ledig et al, [arXiv:1609.04802](arXiv:1609.04802)


bi-cubic


SR-GAN

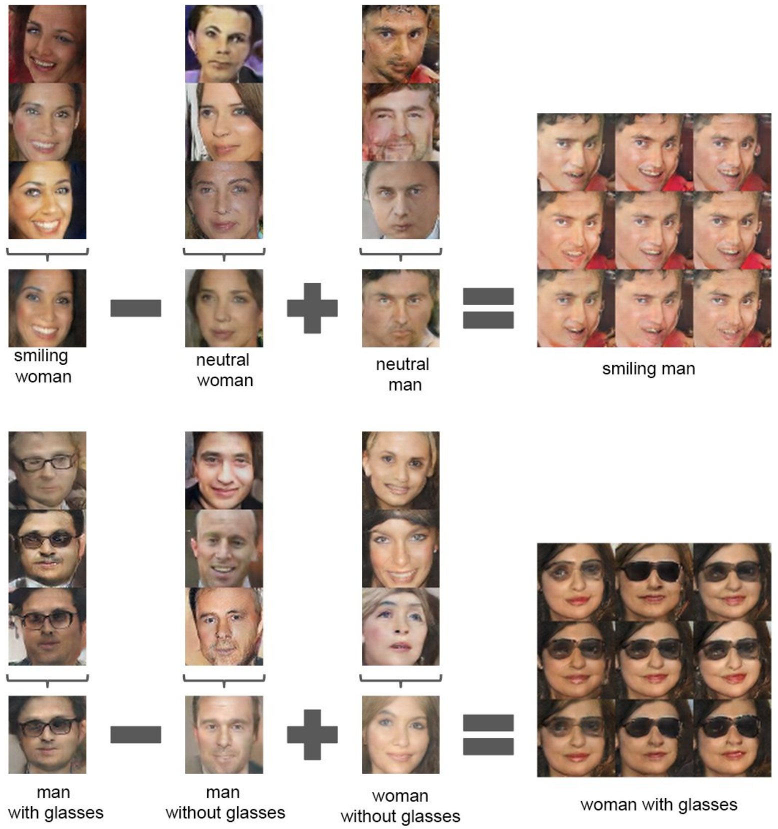

Original

# Missing data imputation



Lee et al. CVPR 2019

# Learning useful representations



## Supervised Learning

Given observed data $x_1, x_2, \cdots, x_n$ and labels $y_1, y_2, \cdots, y_n$, learn a mapping from X → Y, i.e. model the conditional distribution

$$P(Y|X = x)$$

**Examples:** classification, regression, object detection, segmentation, etc.

## Supervised Learning

Given observed data $x_1, x_2, \cdots, x_n$ and labels $y_1, y_2, \cdots, y_n$, learn a mapping from X → Y, i.e. model the conditional distribution

$$P(Y|X = x)$$

**Examples:** classification, regression, object detection, segmentation, etc.

## Unsupervised Learning

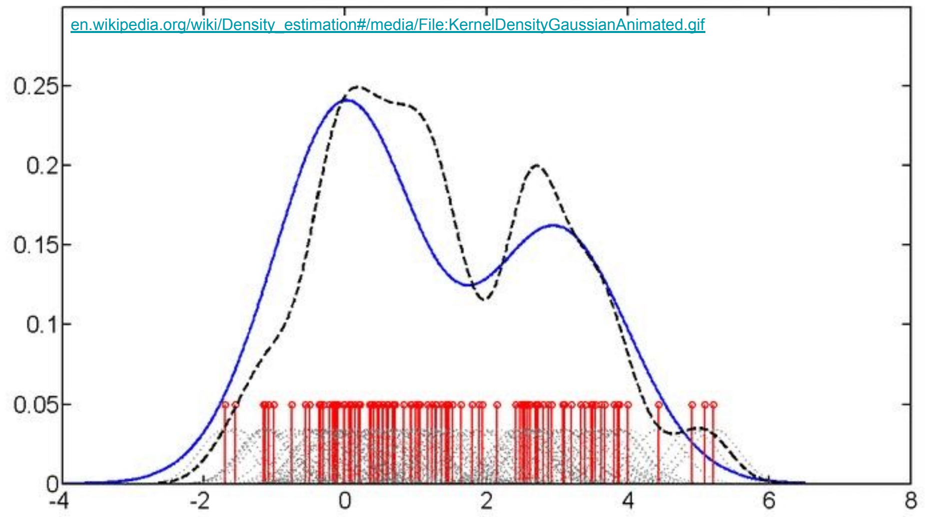
Given observed data $x_1, x_2, \cdots, x_n$ learn the underlying structure of the data, i.e. model the data distribution

$$P(x)$$

**Examples:** dimensionality reduction, data generation, data imputation, data completion, etc.

## Density estimation

An example of density estimation of observed data (red) using Kernel Density Estimation (dashed black). True distribution is a mixture of two-gaussians (blue).



Deep generative models are density estimators $P_{model}(x)$ of high-dimensional data $P_{data}(x)$ (e.g. images)

# Explicit density models: PixelCNN, PixelRNN, WaveNet

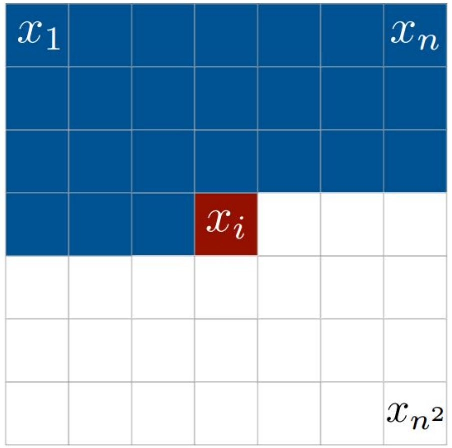
Decompose the data and model it auto-regressively. For example, the likelihood of an image $p(x)$ can be decomposed in terms of the other 1-d distributions:

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, ..., x_{i-1})$$

likelihood of image X

likelihood of the ith pixel

Figure credit: H. Sharma



Then use maximum-likelihood to maximize the likelihood of training data.

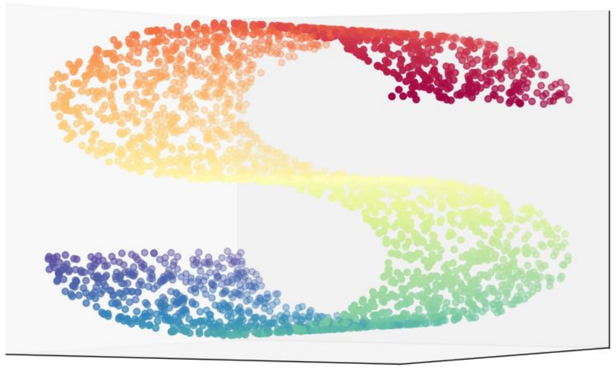
## Latent Variable Models

## The manifold hypothesis

Machine-learning is predicated on a hypothesis about the structure of real-data, namely:

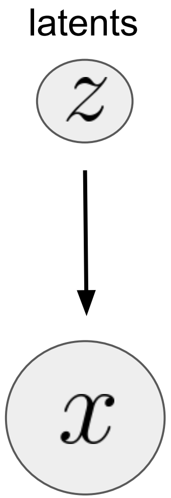
**real-world high-dimensional data lie on low-dimensional manifolds embedded within the high-dimensional space.**

We want to discover the complicated features of this lower-dimensional manifold, often with the downstream task in mind.


2D data manifold embedded within 3D space

## Latent variable models

Learn a mapping between a simpler (latent) manifold and the real data-manifold.

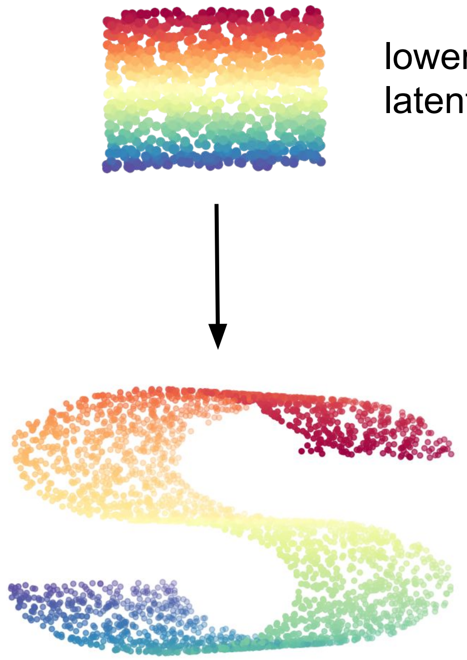


## Latent variable models

Learn a mapping between a simpler (latent) manifold and the real data-manifold.

latents

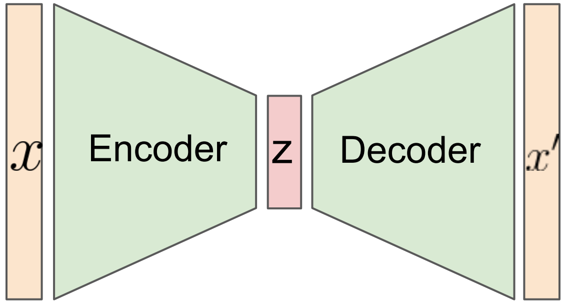
$$\begin{tikzcd} z \arrow[d] \\ x \end{tikzcd}$$



## Variational AutoEncoders

## AutoEncoders

Learn to encode data into a lower-dimensional latent representation by training the model to reconstruct the data passing through an information bottleneck.

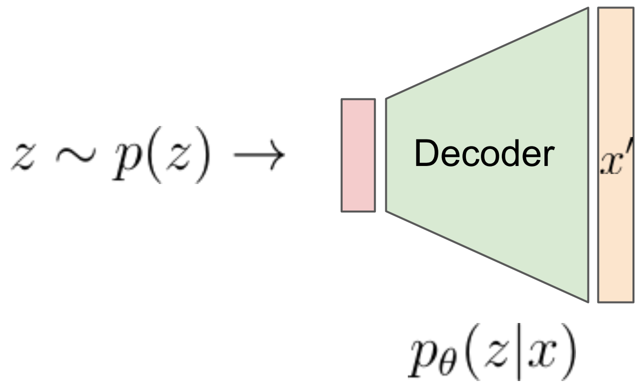


Reconstruction loss: $\mathcal{L}(x, x') = ||x - x'||^2$

# Variational AutoEncoder

Can we impose a prior on the latent representation that allows for inference?

i.e. we want to be able to sample new points from the latent space $z \sim p(z)$ and sample the model posterior distribution $p_\theta(z|x)$



## Variational AutoEncoder

To get the full picture of VAEs and the derivation of their loss as an Evidence Lower Bound on the log-likelihood of data, you need to consider the probabilistic interpretation. See the original VAE paper (Kingma & Welling, arXiv:1312.6114).

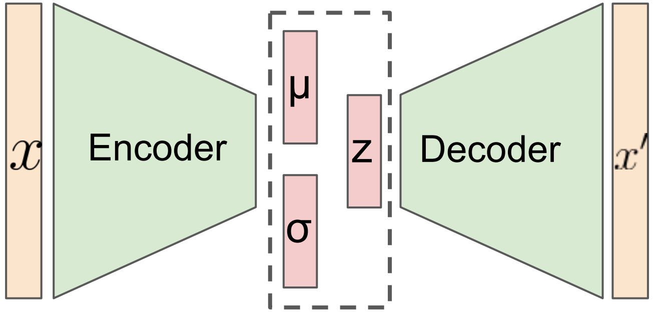
You can also find a full derivation at the excellent "Tutorial: Deriving the Standard Variational Autoencoder (VAE) Loss Function" by Stephen Odaibo.

In the following slides I will try to use a hacky/hand-wavy argument to conjure up the loss function. This follows the same line of development of Alexander Amini, MIT 6.s191 class.

# Variational AutoEncoder

Replace the deterministic latent representation with a stochastic sampler.

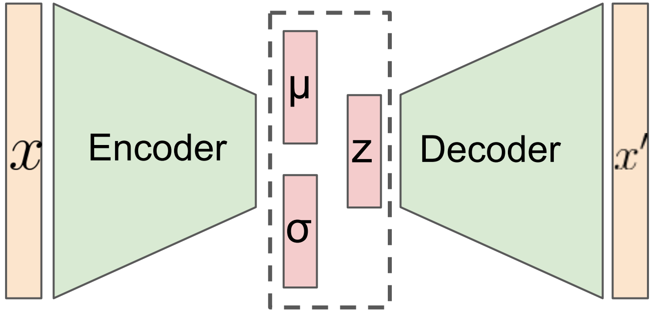
$$z \sim \mathcal{N}(\mu, \sigma)$$



# Variational AutoEncoder

Add a loss regularization term that “encourages” the econder to distribution to match a prior.

$$z \sim \mathcal{N}(\mu, \sigma)$$

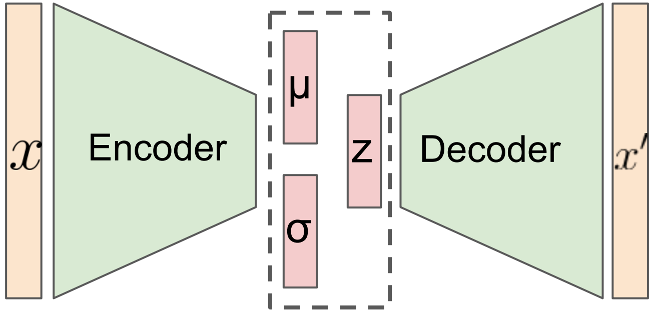


VAE loss = (reconstruction loss) + $D(p_\phi(z|x)||p(z))$

# Variational AutoEncoder

Add a loss regularization term that “encourages” the econder to distribution to match a prior.

$$z \sim \mathcal{N}(\mu, \sigma)$$



VAE loss = (reconstruction loss) + $D(p_\phi(z|x)||p(z))$

$$\downarrow$$

$$D(\mathcal{N}(\mu, \sigma)||\mathcal{N}(0, 1))$$

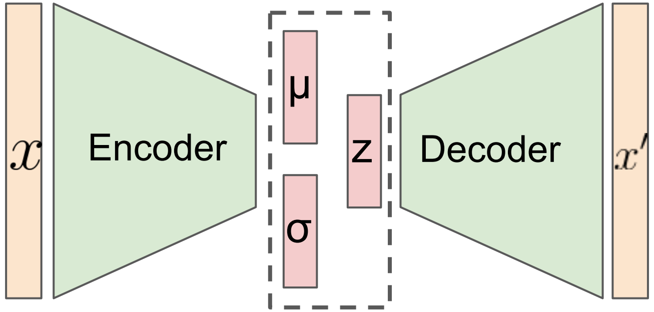
$$\downarrow$$

$$= -\frac{1}{2}\sum_j (\sigma_j + \mu_j^2 - 1 - \log \sigma_j)$$

# Variational AutoEncoder

You can't backprop throw a stochastic layer → use the reparameterization trick

$$z \sim \mathcal{N}(\mu, \sigma) \rightarrow \sigma * \epsilon + \mu \quad \text{where } \epsilon \sim \mathcal{N}(0,1)$$



VAE loss = (reconstruction loss) + $D(p_\phi(z|x)||p(z))$

$$\downarrow$$

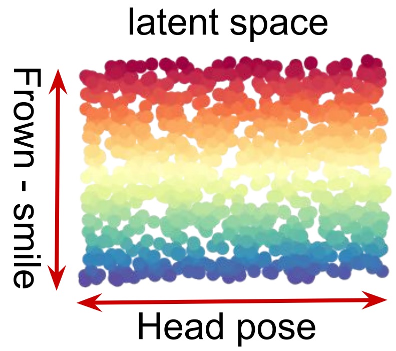
$$D(\mathcal{N}(\mu,\sigma)||\mathcal{N}(0,1))$$

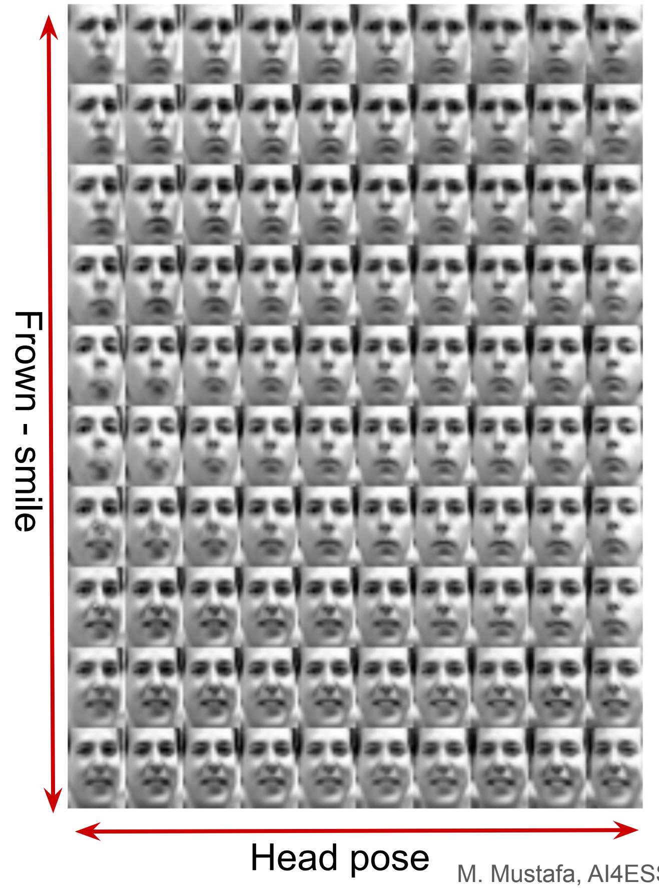
$$\downarrow$$

$$= -\frac{1}{2}\sum_j(\sigma_j + \mu_j^2 - 1 - \log\sigma_j)$$

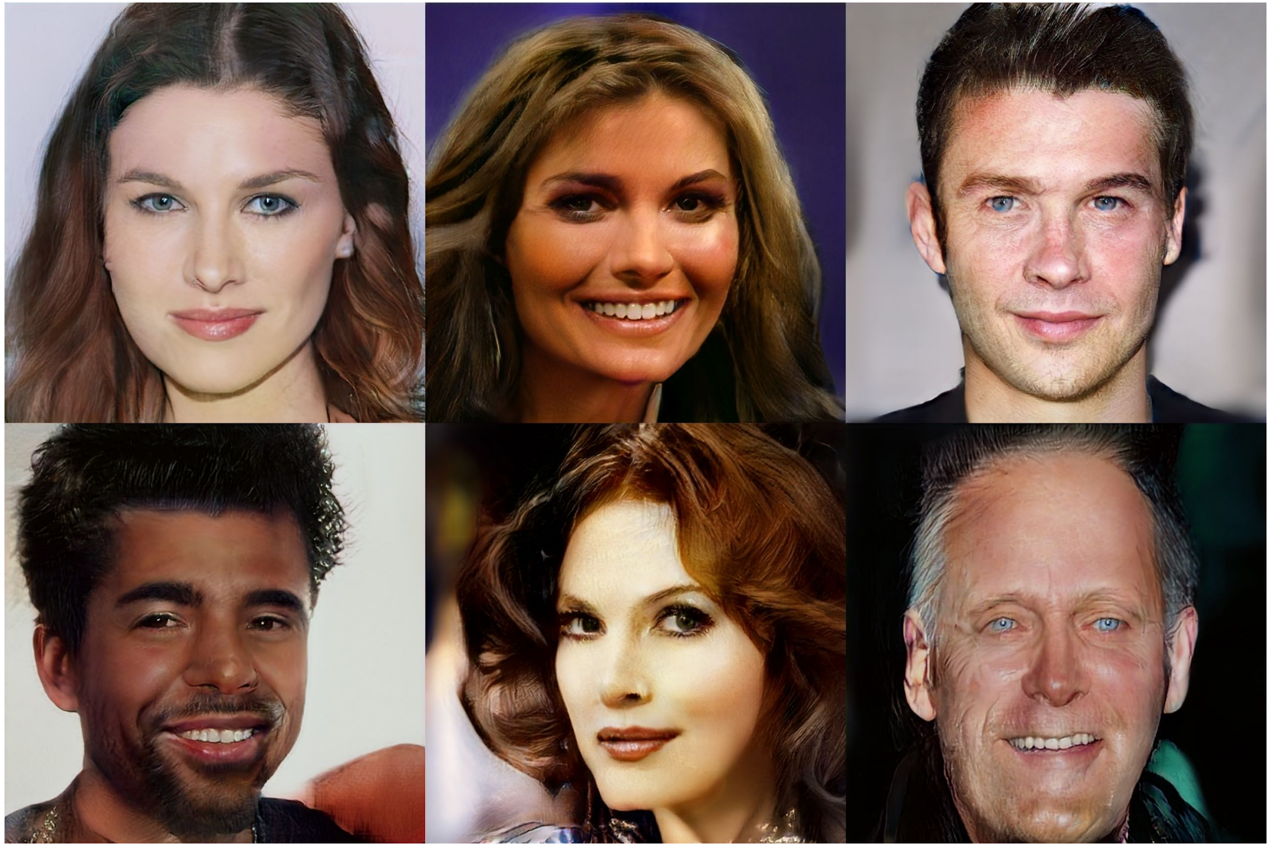
# Semantically meaningful directions in the learned latent space

Kingma & Welling, arXiv:1312.6114





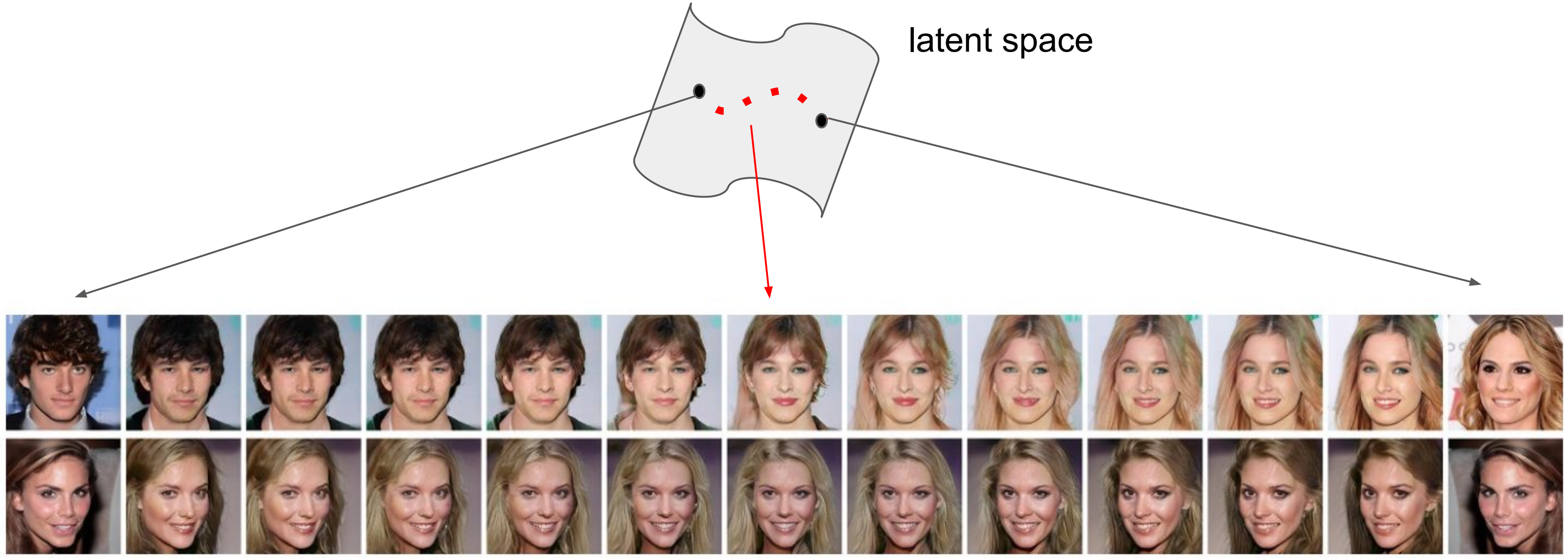
# Performance of modern VAEs: IntroVAE



“IntroVAE: Introspective VAE for Photographic Image Synthesis”, Huang et al., NeurIPS 2018, code: github.com/hhb072/IntroVAE

# Exploring the learned latent representation of IntroVAE

Interpolations in the latent space correspond to smooth transitions over the generated data manifold

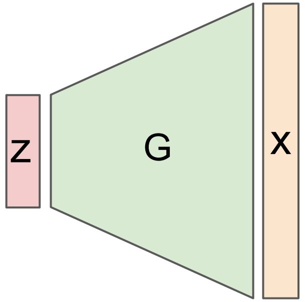


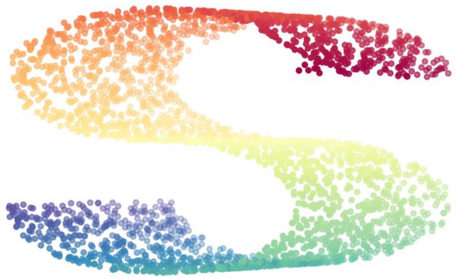
"IntroVAE", Huang et al., NeurIPS 2018, code: github.com/hhb072/IntroVAE

## Generative Adversarial Networks

# Generative Adversarial Networks (GANs)

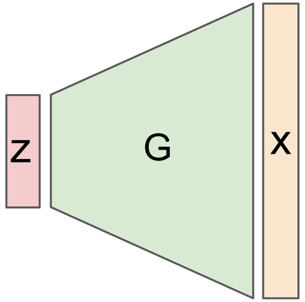
We want to learn how to generate how to generate samples that look similar to the real-data; approximately from the real data manifold.

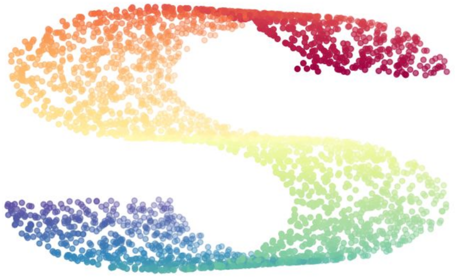




# Generative Adversarial Networks (GANs)

We want to learn how to generate how to generate samples that look similar to the real-data; approximately from the real data manifold.

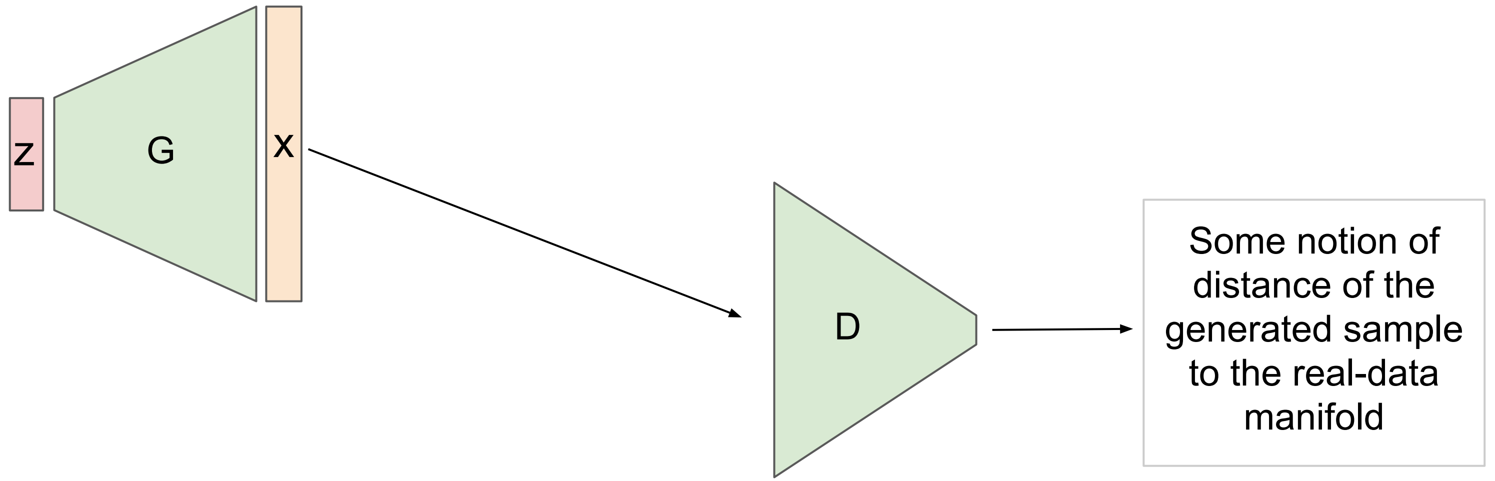




In other words, we a need a loss function that tells us how close the generated data is to the real-data manifold: $\mathcal{L} = f(G(z))$

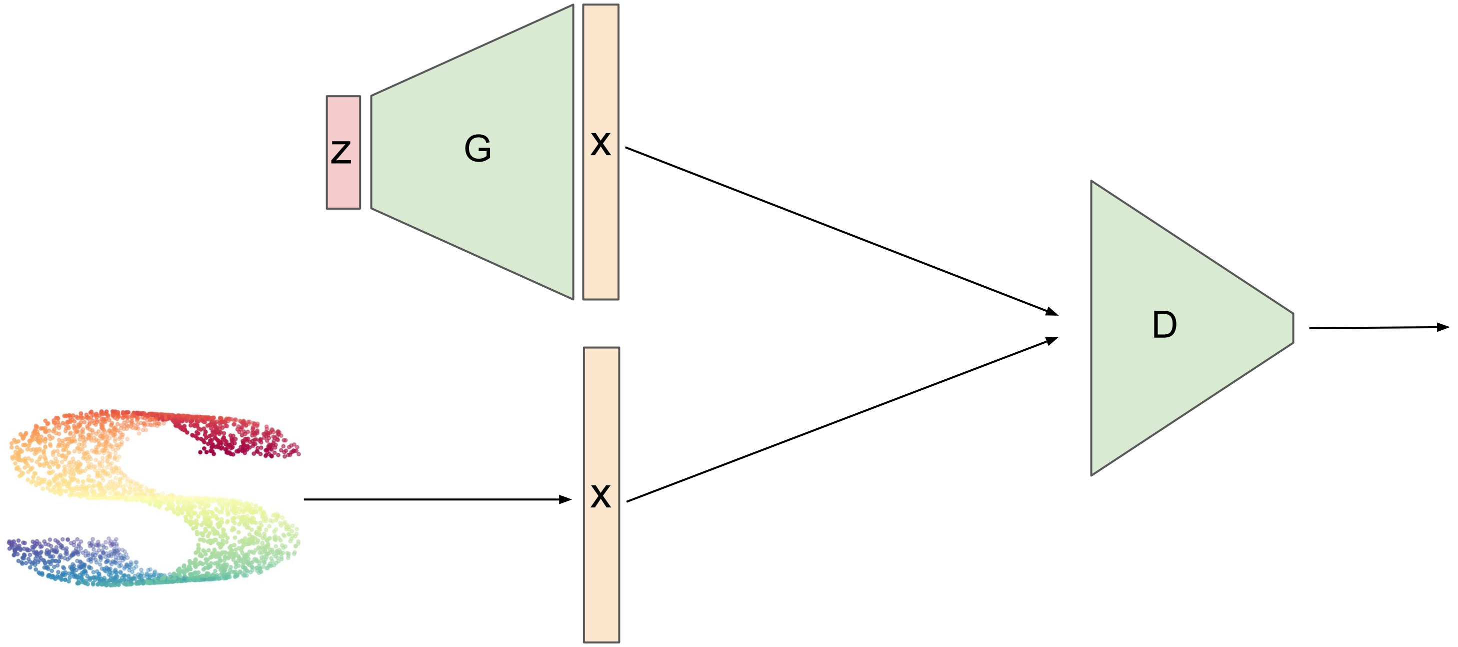
Can we learn such an $f$ ?

# Generative Adversarial Networks (GANs)


Z
G
X
D
Some notion of distance of the generated sample to the real-data manifold

Yes, we can learn a function that evaluates how close a generated sample is to real data.

# Generative Adversarial Networks (GANs)



Discriminator should learn how to "discriminate" between real and fake data samples.

The generator should learn how to "fool" the discriminator by creating more realistic samples"

The new network (discriminator/critic) is trained "adversarially.

## GANs' loss (original loss)

The discriminators is trained to minimize:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_{z\sim p_z} \log(1 - D((G(z)))$$

learn to give a high score to samples from realdata

learn to give a low score to samples from generated data

## GANs’ loss (original loss)

The discriminators is trained to minimize:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_{z\sim p_z} \log(1 - D((G(z)))$$

In the minimax game formulation, you train the generator to maximize the discriminators loss (to fool the discriminator):

$$J^{(G)} = -J^{(D)}$$

# GANs' loss (original loss)

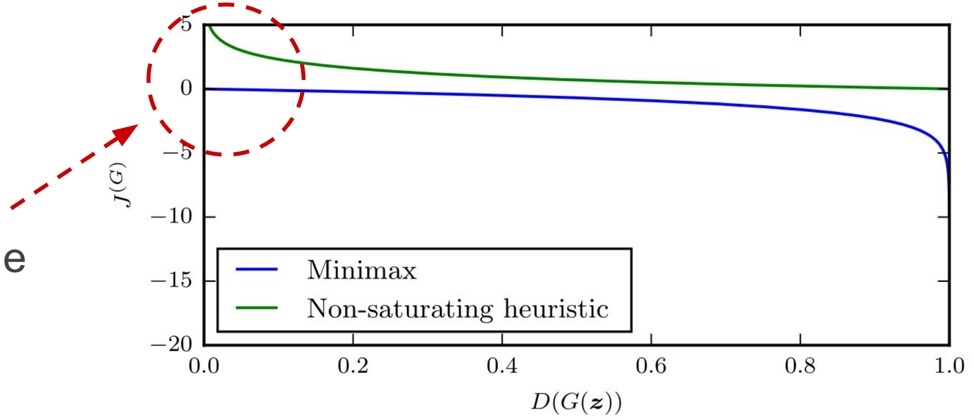
The discriminators is trained to minimize:

$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_{z\sim p_z} \log(1 - D((G(z)))$$

In the minimax game formulation, you train the generator to maximize the discriminators loss (to fool the discriminator):

$$J^{(G)} = -J^{(D)}$$

But this is has vanishing-gradients when the discriminator is confused about generated samples; vanishing gradients are bad for gradient descent optimization



# GANs' loss (original loss)

The discriminators is trained to minimize:

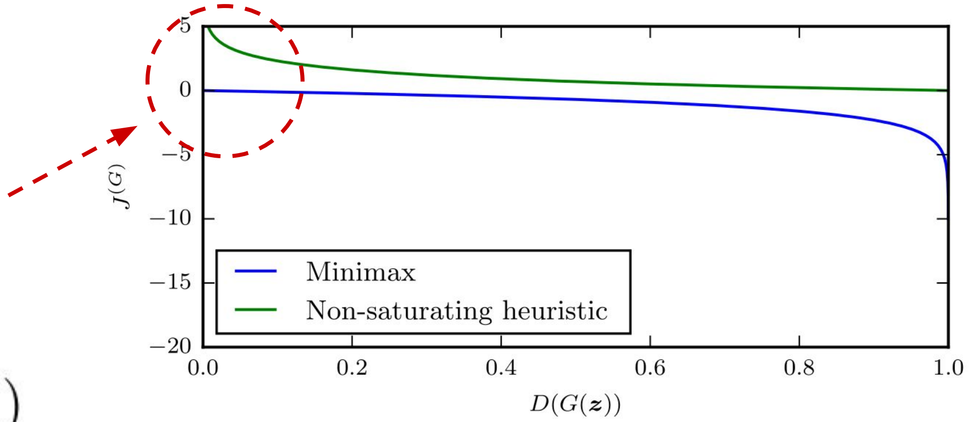
$$J^{(D)} = -\frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_{z\sim p_z} \log(1 - D((G(z)))$$

In the minimax game formulation, you train the generator to maximize the discriminators loss (to fool the discriminator):

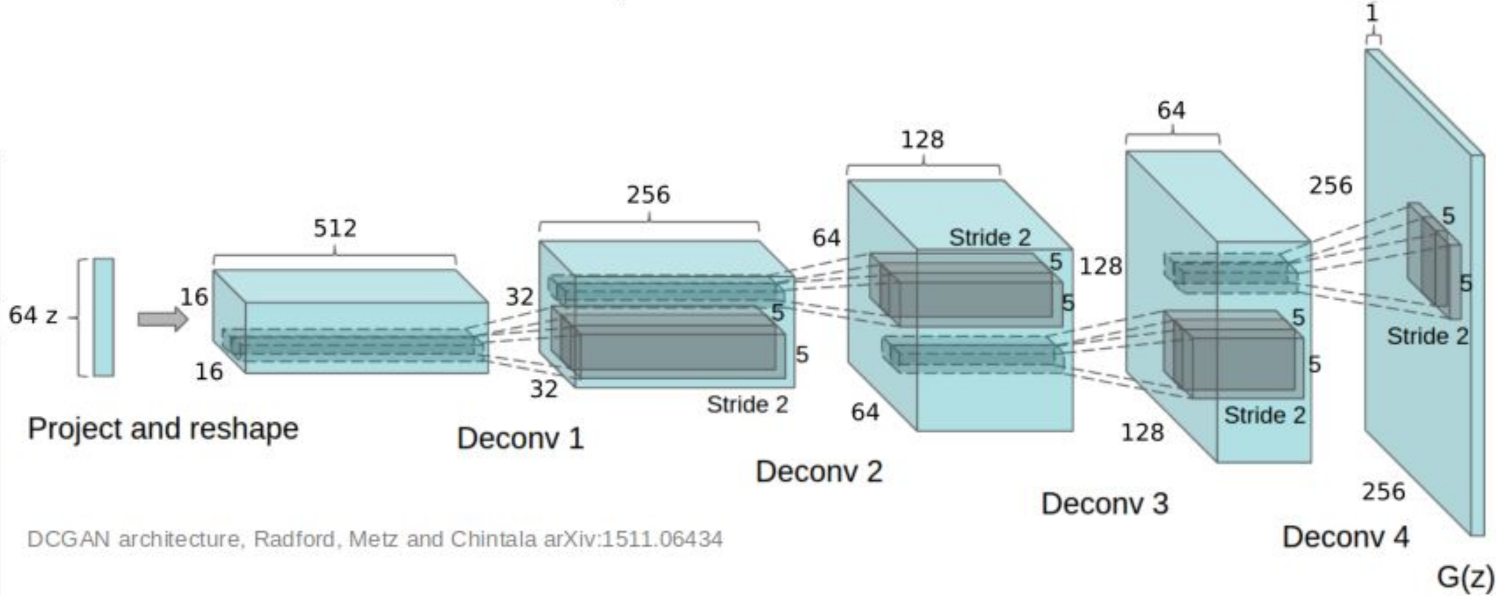
$$J^{(G)} = -J^{(D)}$$

The original GAN paper proposes to use a non-saturating "heuristic" loss to train the generator

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_{z\sim p_z} \log D(G(z))$$

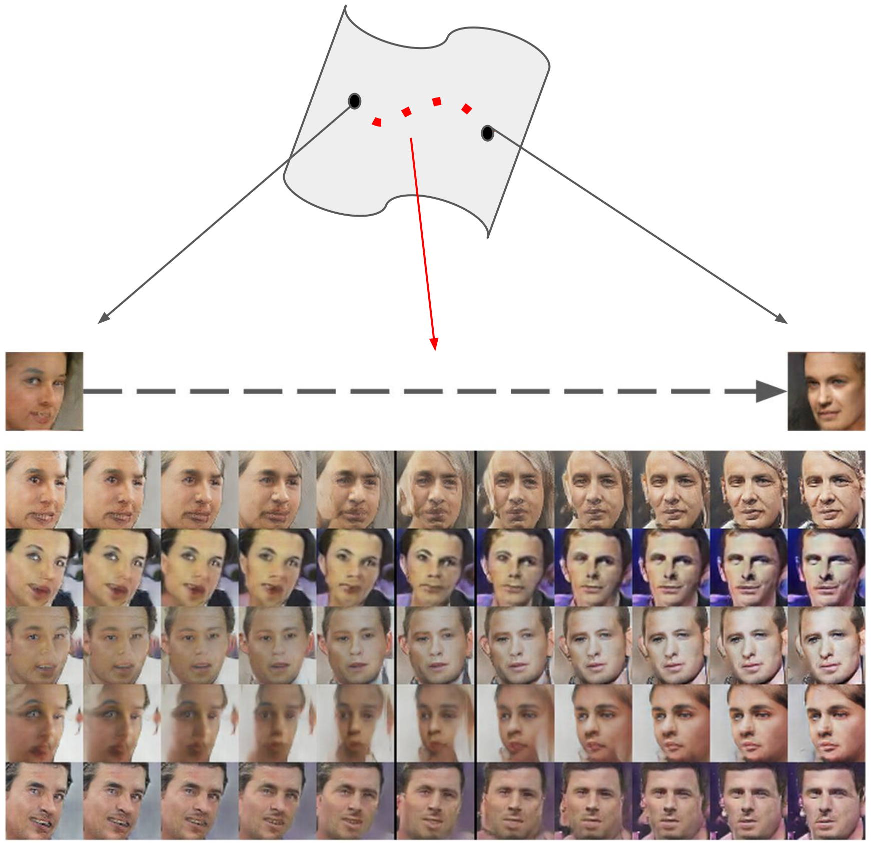


# DCGAN generator architecture



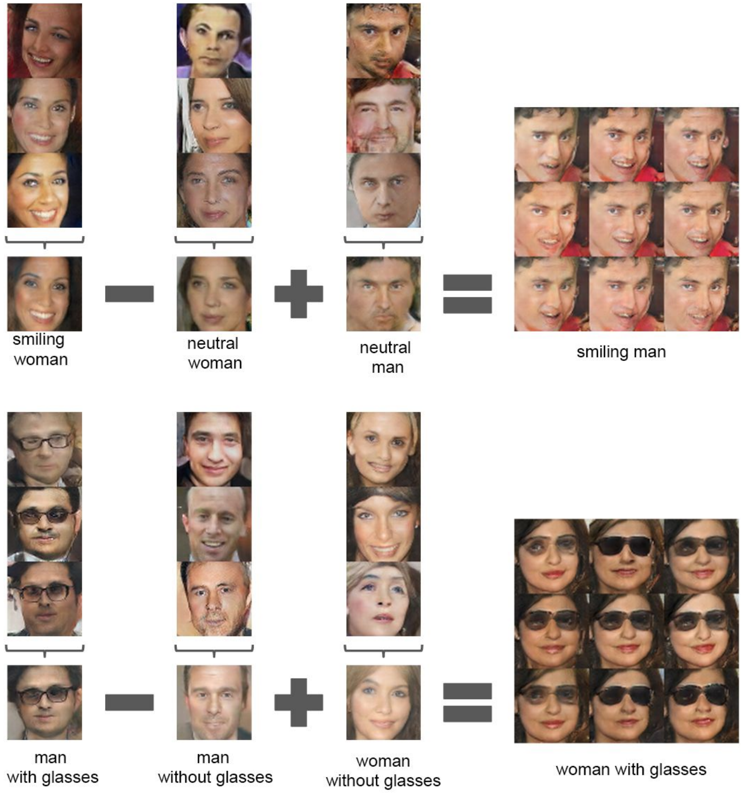
DCGAN architecture, Radford, Metz and Chintala arXiv:1511.06434

# Exploring the learned latent representation: Interpolation



DCGAN, Radford, Metz and Chintala, arXiv:1511.06434

## Exploring the learned latent representation: directions in the latent space are semantically meaningful (simple arithmetics)



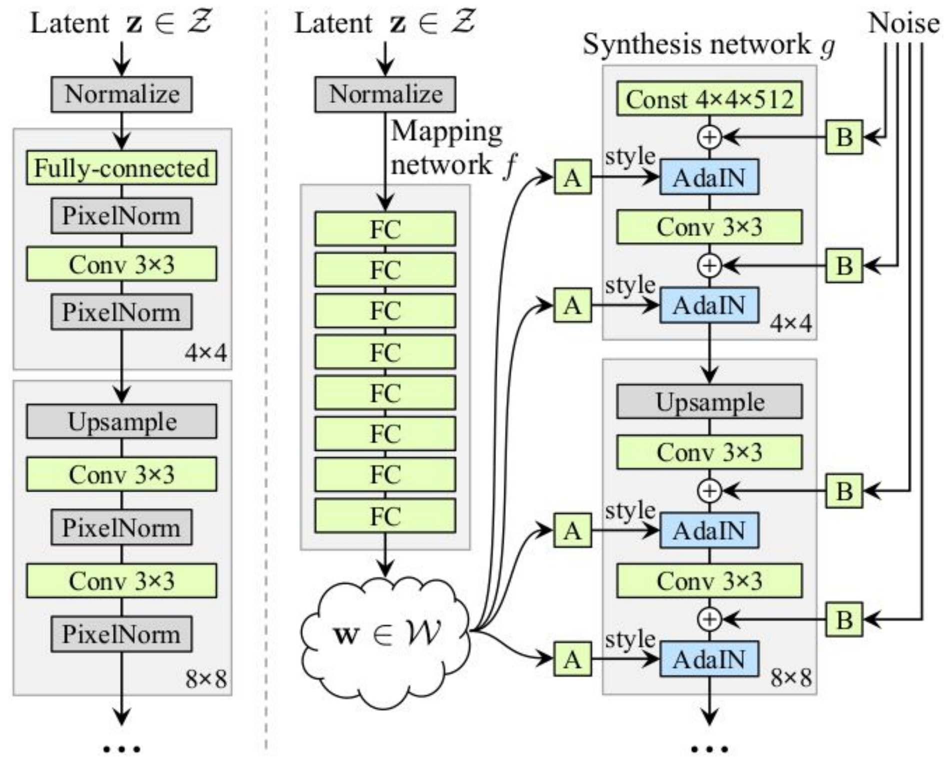
DCGAN, Radford, Metz and Chintala, arXiv:1511.06434

## Performance of more recent GAN models: StyleGAN-II

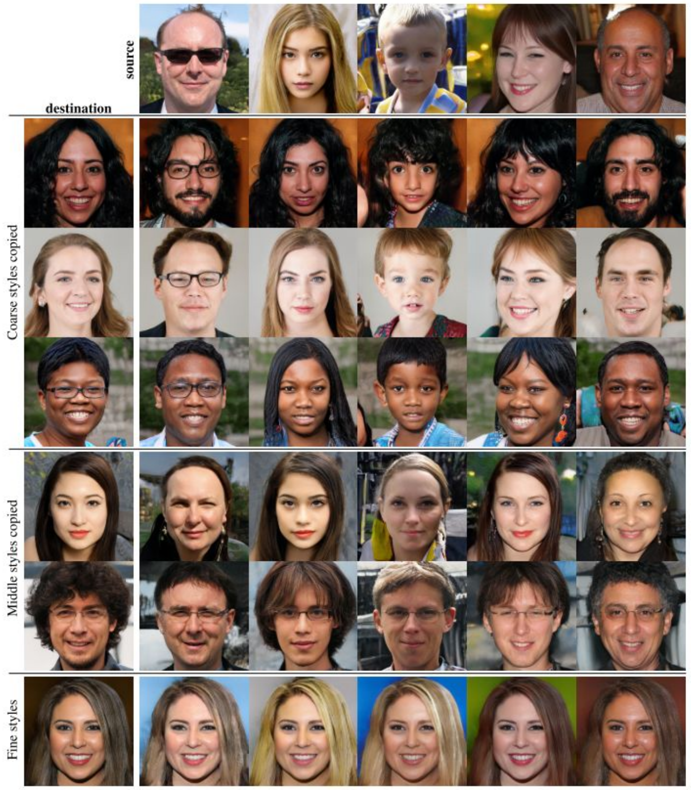
StyleGAN-II, T. Karras et al, arXiv:1912.04958

# StyleGAN

StyleGAN, Karras, arXiv:1812.04948



## Visualizing the effect of styles in StyleGAN “Disentangled representations”?

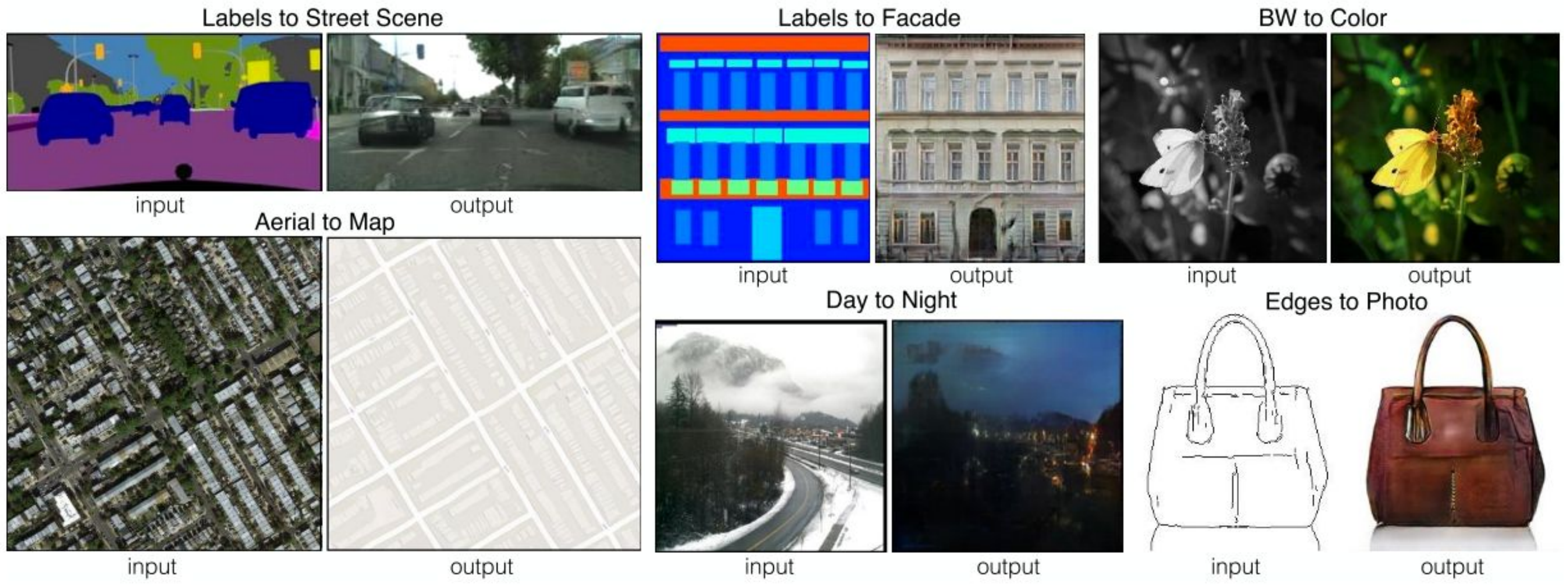


StyleGAN, Karras, arXiv:1812.04948

## Other tasks that make use of an Adversarial Loss

# Image-to-image translation (domain-to-domain)

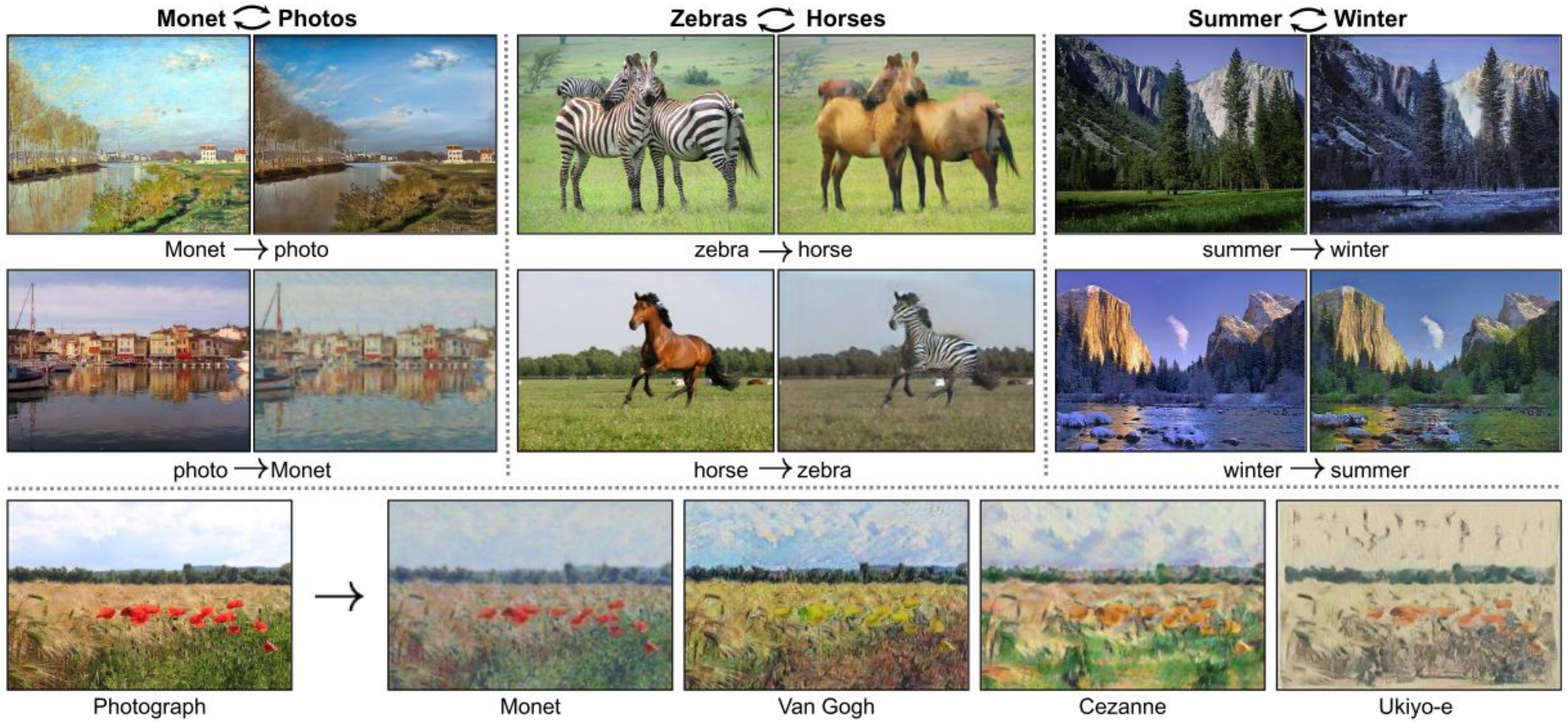
Learns a one-to-one mapping between domain X and Y, training with paired images.



Pix2Pix, Isola et. al, arXiv:1611.07004

# Unpaired Image to Image Translation (CycleGAN)

Learns a one-to-one mapping between domain X and Y without paired images.

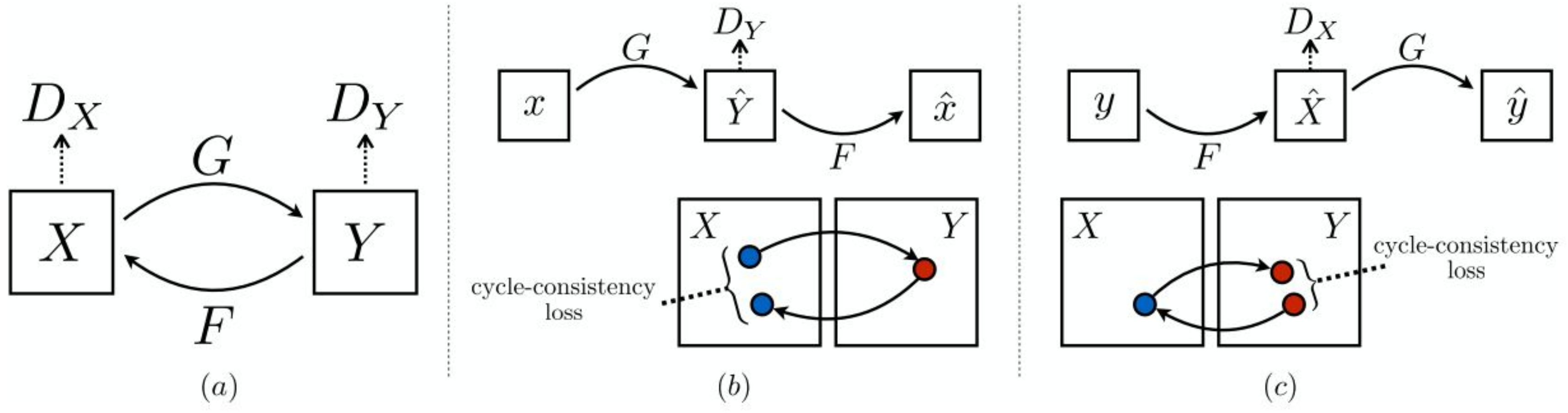


CycleGAN, Zhu et al, [arXiv:1703.10593](https://arxiv.org/abs/1703.10593)

BERKELEY LAB

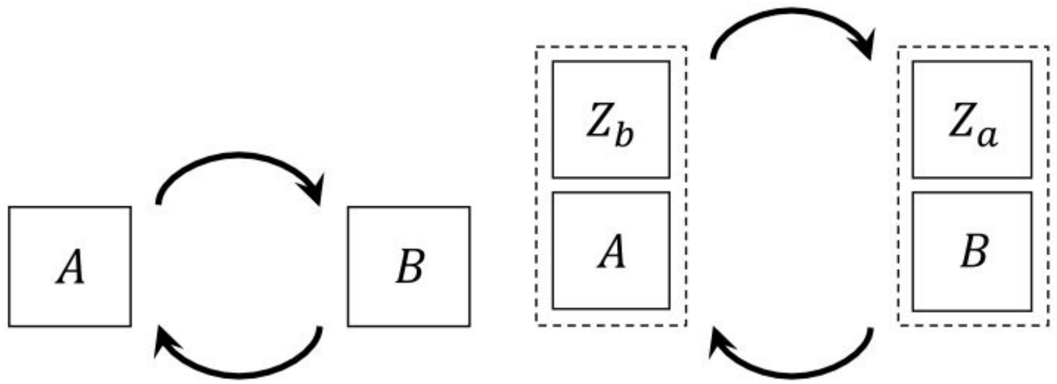
# Cycle Consistency

CycleGAN, Zhu et al, [arXiv:1703.10593](https://arxiv.org/abs/1703.10593)



$$x \to G(x) \to F(G(x)) \approx x$$

# Augmented CycleGAN (many-to-many)



(a) CycleGAN (b) Augmented CycleGAN

Augmented CycleGAN, Almahairi et. al arXiv:1802.10151

# Augmented CycleGAN (many-to-many)

Augmented CycleGAN, Almahairi et. al [arXiv:1802.10151](arXiv:1802.10151)

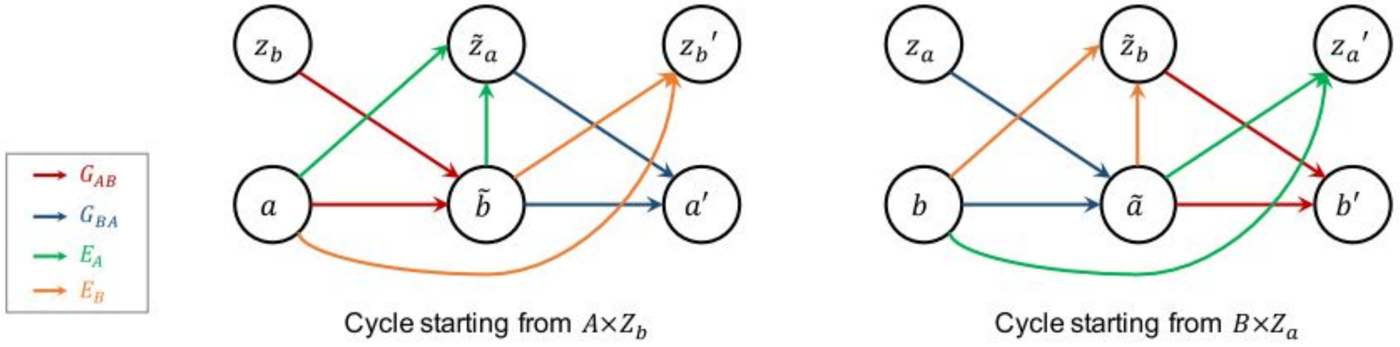
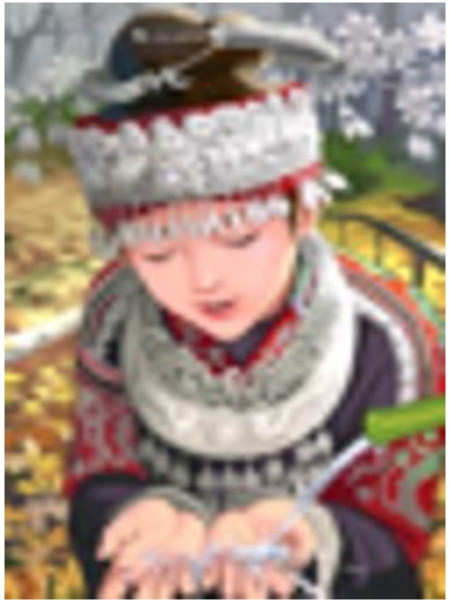


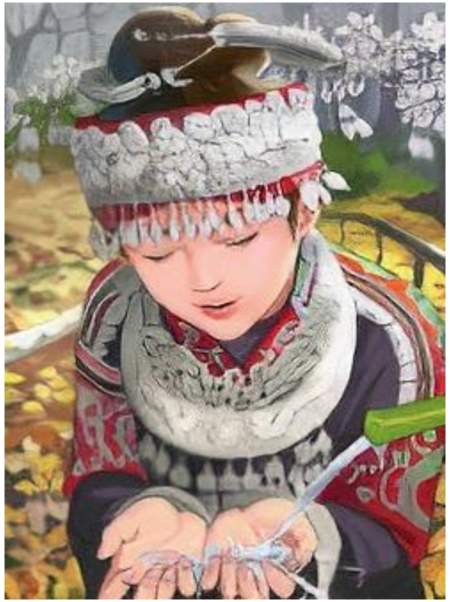
Figure 2: Cycles starting from augmented spaces in Augmented CycleGAN. Model components identified with color coding.

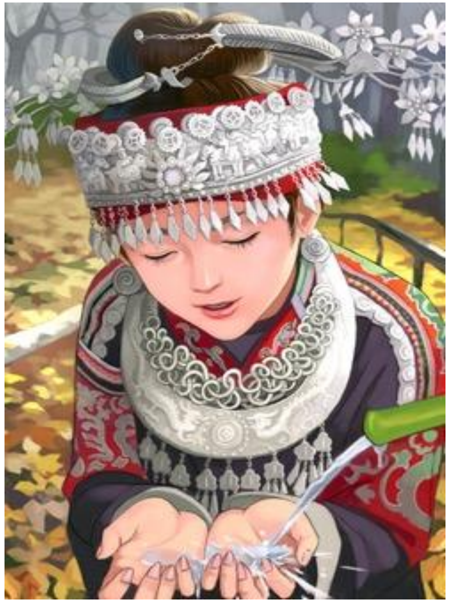
# Super-resolution

Given a low-resolution image, learn how to create a high-fidelity high-resolution image.

SRGAN, Ledig et al, arXiv:1609.04802


bi-cubic


SR-GAN

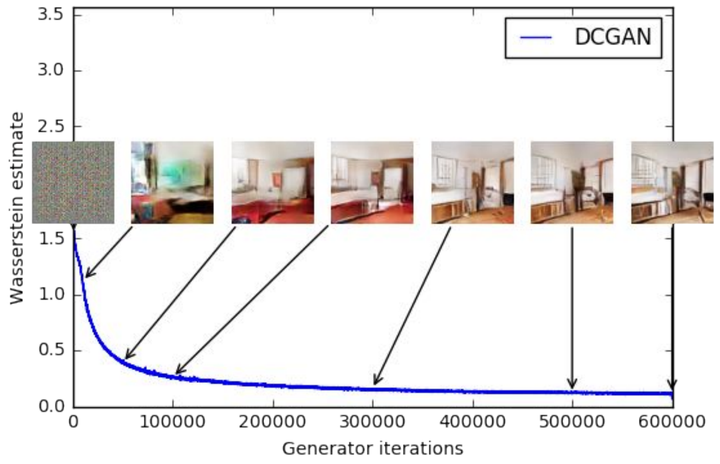

Original

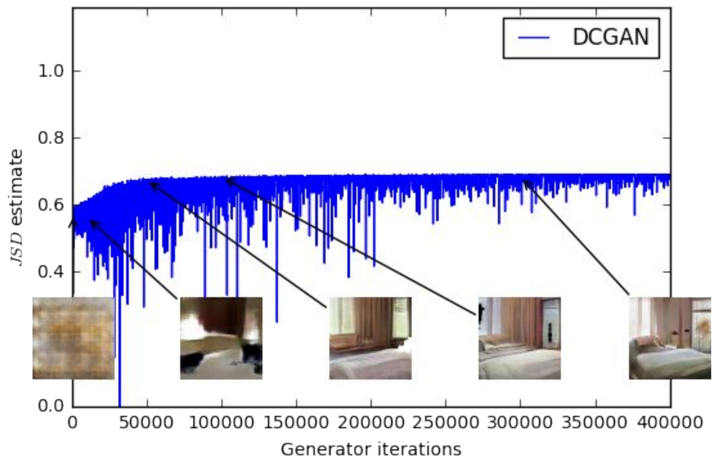
## Challenges with training GANs in practice

- No evaluation metric that can capture all desired data properties
  - generator loss doesn’t correlate with image quality/ desired properties
    - optimal transport based losses/metrics have been proposed (e.g. Wasserstein-GAN)
  - for evaluation, we resort to domain specific metrics that correspond to the desired property
    - for vision tasks the community has developed proxy metrics for human perception
- GAN training dynamics is finicky

# Challenges with training GANs in practice

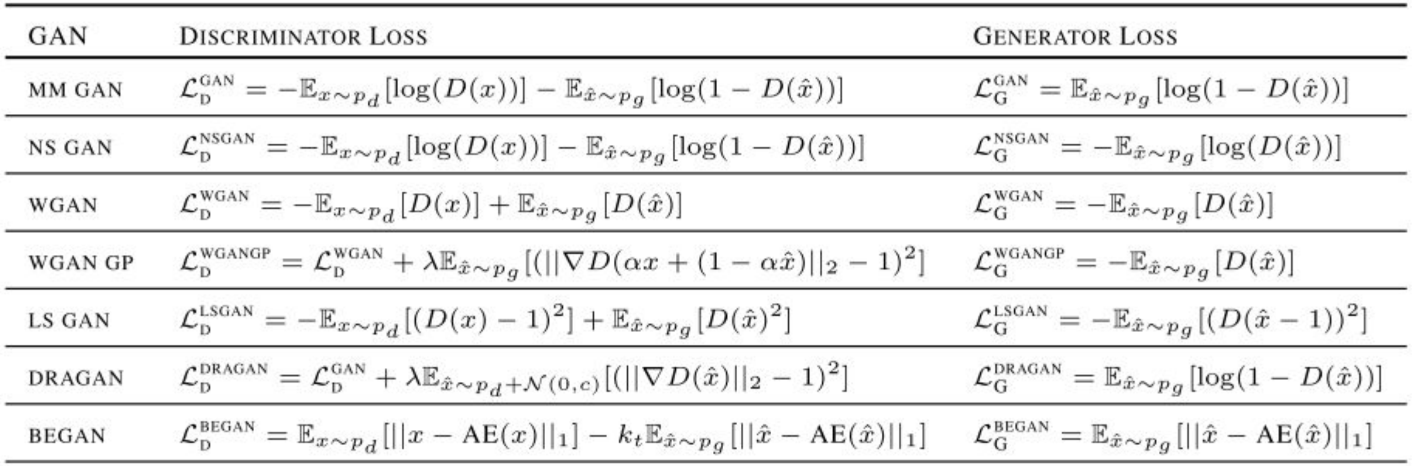
- No evaluation metric that can capture all desired data properties
  - generator loss doesn't correlate with image quality/ desired properties
    - optimal transport based losses/metrics have been proposed (e.g. Wasserstein-GAN)
  - for evaluation, we resort to domain specific metrics that correspond to the desired property
    - for vision tasks the community has developed proxy metrics for human perception
- GAN training dynamics is finicky





Wasserstein GAN, Arjovsky, Chintala and Bottou arXiv:1701.07875

# A “zoo” of proposed GAN losses



| GAN | Discriminator Loss | Generator Loss |
|---|---|---|
| MM GAN | $\mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$ | $\mathcal{L}_{\mathrm{G}}^{\mathrm{GAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$ |
| NS GAN | $\mathcal{L}_{\mathrm{D}}^{\mathrm{NSGAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$ | $\mathcal{L}_{\mathrm{G}}^{\mathrm{NSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$ |
| WGAN | $\mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$ | $\mathcal{L}_{\mathrm{G}}^{\mathrm{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$ |
| WGAN GP | $\mathcal{L}_{\mathrm{D}}^{\mathrm{WGANGP}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\|\nabla D(\alpha x + (1 - \alpha\hat{x})\|\|_2 - 1)^2]$ | $\mathcal{L}_{\mathrm{G}}^{\mathrm{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$ |
| LS GAN | $\mathcal{L}_{\mathrm{D}}^{\mathrm{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x) - 1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$ | $\mathcal{L}_{\mathrm{G}}^{\mathrm{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x} - 1))^2]$ |
| DRAGAN | $\mathcal{L}_{\mathrm{D}}^{\mathrm{DRAGAN}} = \mathcal{L}_{\mathrm{D}}^{\mathrm{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\|\nabla D(\hat{x})\|\|_2 - 1)^2]$ | $\mathcal{L}_{\mathrm{G}}^{\mathrm{DRAGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1 - D(\hat{x}))]$ |
| BEGAN | $\mathcal{L}_{\mathrm{D}}^{\mathrm{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x - \mathrm{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \mathrm{AE}(\hat{x})\|\|_1]$ | $\mathcal{L}_{\mathrm{G}}^{\mathrm{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \mathrm{AE}(\hat{x})\|\|_1]$ |

“Are GANs Created Equal? A Large-Scale Study”, Lucic et al, arXiv:1711.10337

BERKELEY LAB

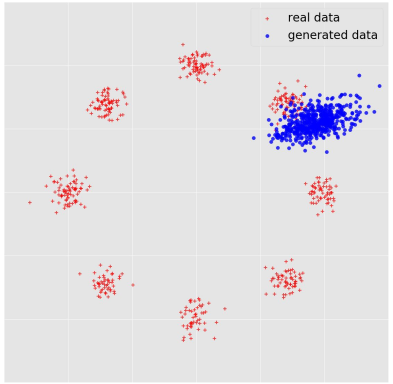
# Challenges with training GANs in practice

- No evaluation metric that can capture all desired data properties
  - generator loss doesn’t correlate with image quality/ desired properties
    - optimal transport based losses/metrics have been proposed (e.g. Wasserstein-GAN)
  - for evaluation, we resort to domain specific metrics that correspond to the desired property
    - for vision tasks the community has developed proxy metrics for human perception
- GAN training dynamics is finicky
  - the discriminator could win the game before the generator starts to produce high-fidelity samples
    - introduce some noise in the discriminator (e.g. label flipping, label smoothing, etc)

# Challenges with training GANs in practice

- No evaluation metric that can capture all desired data properties
  - generator loss doesn’t correlate with image quality/ desired properties
    - optimal transport based losses/metrics have been proposed (e.g. Wasserstein-GAN)
  - for evaluation, we resort to domain specific metrics that correspond to the desired property
    - for vision tasks the community has developed proxy metrics for human perception

- GAN training dynamics is finicky
  - the discriminator could win the game before the generator starts to produce high-fidelity samples
    - introduce some noise in the discriminator (e.g. label flipping, label smoothing, etc)
  - generators samples from a single mode of the real data (mode collapse)

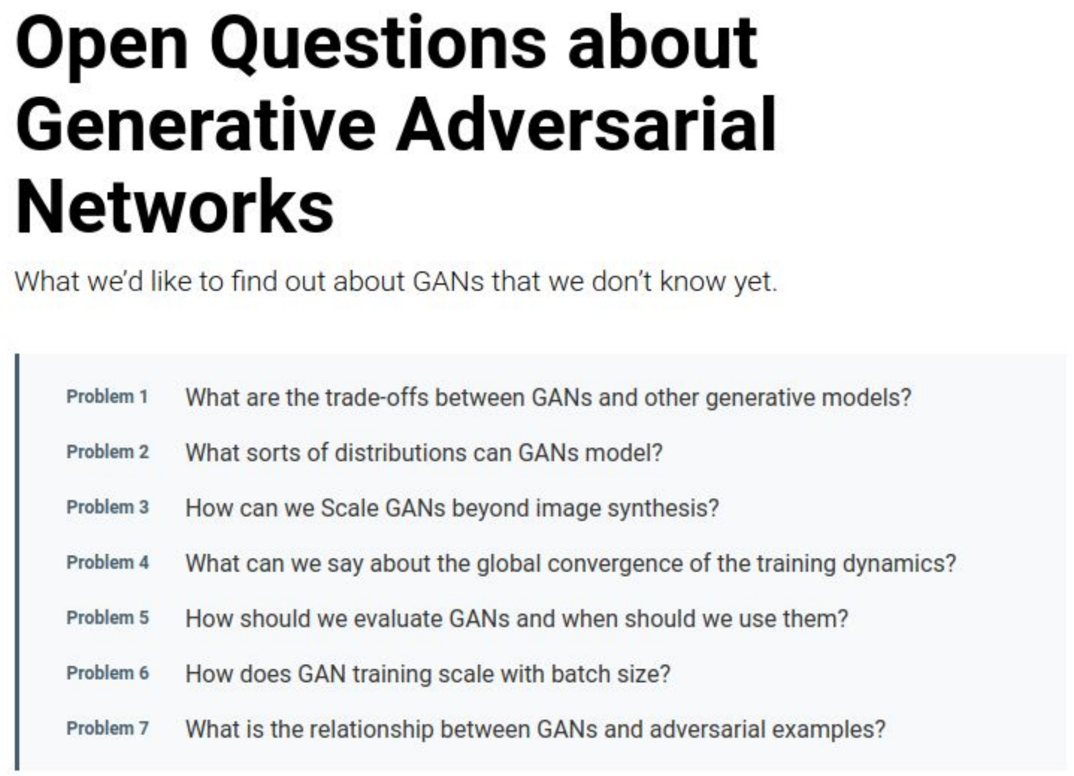


MGAN, Hoang et al, arXiv:1708.02556

## Challenges with training GANs in practice

- No evaluation metric that can capture all desired data properties
  - generator loss doesn't correlate with image quality/ desired properties
    - optimal transport based losses/metrics have been proposed (e.g. Wasserstein-GAN)
  - for evaluation, we resort to domain specific metrics that correspond to the desired property
    - for vision tasks the community has developed proxy metrics for human perception

- GAN training dynamics is finicky
  - the discriminator could win the game before the generator starts to produce high-fidelity samples
    - introduce some noise in the discriminator (e.g. label flipping, label smoothing, etc)
  - generators samples from a single mode of the real data (mode collapse)
    - still an open problem, partial solutions and heuristics that don't generalize across problems
    - we might need to resort to multi-mode priors for more complex data
  - the training could be highly unstable (strongly sensitive to hyper-parameters, especially to those of the optimizer). Many proposals for ameliorating this situation:
    - spectral-normalization schemes have been proposed (mixed results)
    - generalization of gradient-descent to two-player games might become necessary (very computationally expensive so far); see : Schäfer et al, arXiv:1910.05852.



## Open Questions about Generative Adversarial Networks

What we'd like to find out about GANs that we don't know yet.

| | |
|---|---|
| Problem 1 | What are the trade-offs between GANs and other generative models? |
| Problem 2 | What sorts of distributions can GANs model? |
| Problem 3 | How can we Scale GANs beyond image synthesis? |
| Problem 4 | What can we say about the global convergence of the training dynamics? |
| Problem 5 | How should we evaluate GANs and when should we use them? |
| Problem 6 | How does GAN training scale with batch size? |
| Problem 7 | What is the relationship between GANs and adversarial examples? |

Augustus Odena, 2019: distill.pub/2019/gan-open-problems/, DOI: 10.23915/distill.00018

## Normalized Flow Models

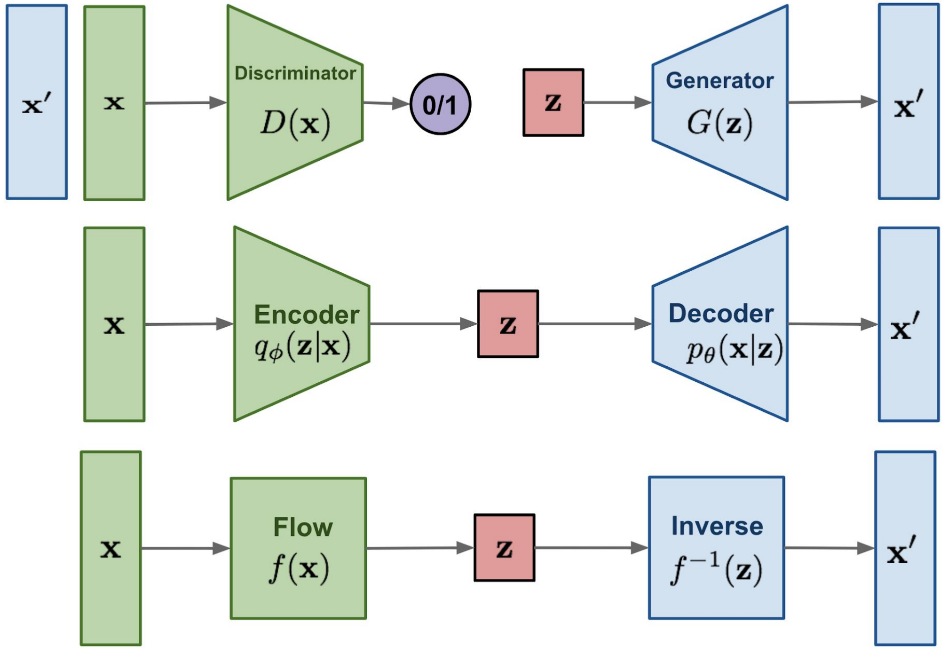
# Normalized Flow Models

Using invertible transformations (layers), flow models try to explicitly learn the true data distribution, thus they are optimized using maximum-likelihood.

**GAN:** minimax the classification error loss.

**VAE:** maximize ELBO.

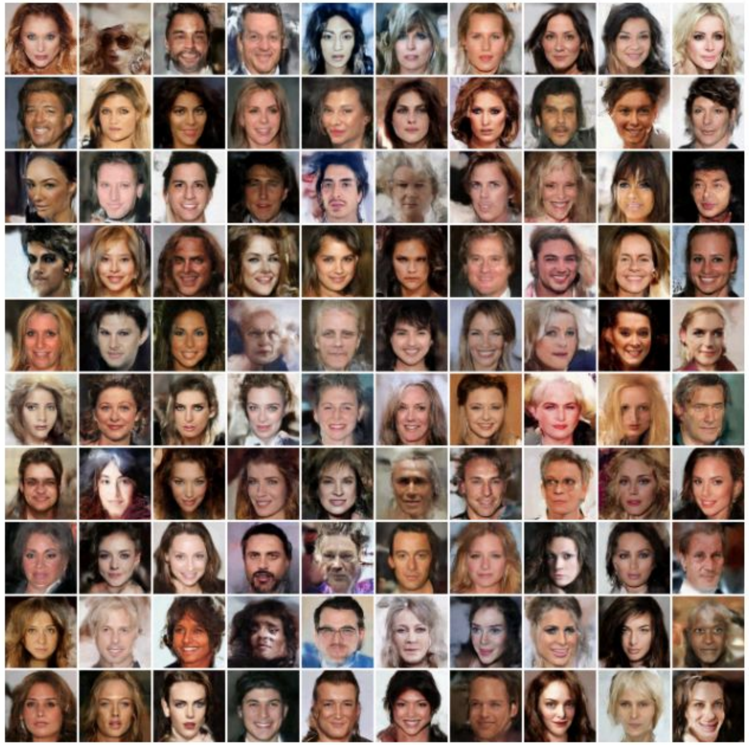
**Flow-based generative models:** minimize the negative log-likelihood


$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
$\mathbf{x}$
Flow $f(\mathbf{x})$
$\mathbf{z}$
Inverse $f^{-1}(\mathbf{z})$
$\mathbf{x}'$

"Flow-based Deep Generative Models", Lilian Weng

# Flow++

Ho et al. arXiv:1902.00275



## More resources

In addition to all the links in these slides, I have enjoyed reading/viewing the following material while I am preparing this lecture:

- "Introduction to generative models", Mihaela Rosca
- "Deep Generative Models", Ava Soleimany, MIT, 6.S191
- "GANs & Unsupervised Representation Learning" , Yoshua Bengio
- "Energy-based GANs & Other Adversarial Training", Yann Lecun
- "Deep Unsupervised Learning", UCB, CS294-158

Practical tutorials with code:

- DCGAN: PyTorch, TensorFlow/Keras
- VAE: PyTorch, TensorFlow/Keras

NERSC

## Thank You

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory