- HW4 due Sunday March 13th Midnight
- At most 3 days of late days allowed (Even if you have more remaining)

https://www.whichfaceisreal.com/

Lecture 27: Deep Generative Models

- Unsupervised learning
  - Dimensionality reduction
    - PCA
    - Auto-encoder
  - Clustering
    - k-means
    - Spectral,t-SNE,UMAP
  - **Generative models**
  - Density estimation

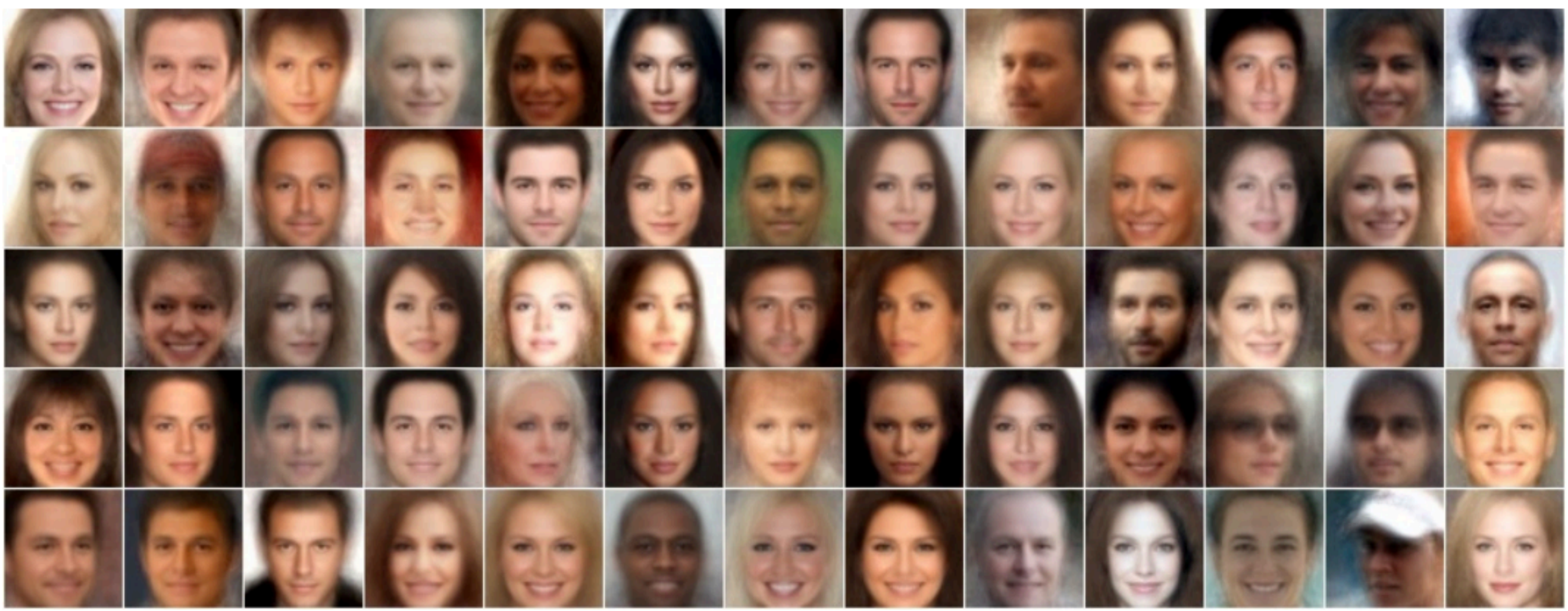
- traditional parametric generative model
  - Gaussian:

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - Gaussian Mixture Models (GMM)

$$f_{\{\mu_i\},\{\sigma_i\},\{\pi_i\}}(x) = \sum_{i=1}^{k} \pi_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$$

Images from “on GANs and GMMs”, 2018, Richardson &Weiss



Deep generative model

- traditional parametric generative model
  - Gaussian:

$$f_{\mu,\sigma}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - Gaussian Mixture Models (GMM)

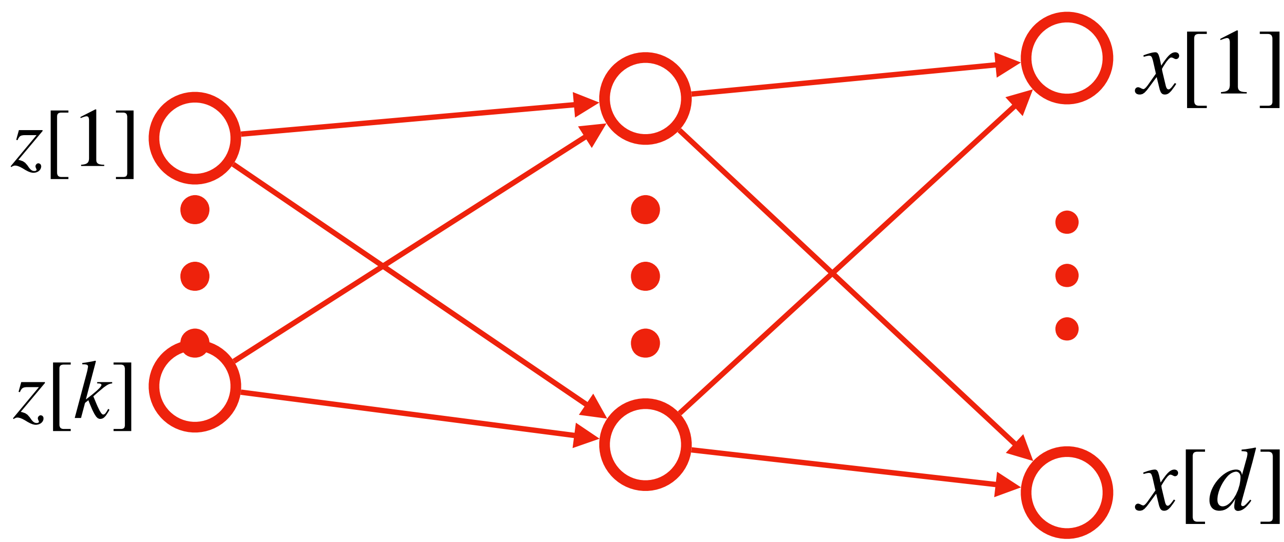
$$f_{\{\mu_i\},\{\sigma_i\},\{\pi_i\}}(x) = \sum_{i=1}^{k} \pi_i \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{(x-\mu_i)^2}{2\sigma_i^2}}$$

  - Because we have the explicit p.d.f,
easy to train with expectation-maximization

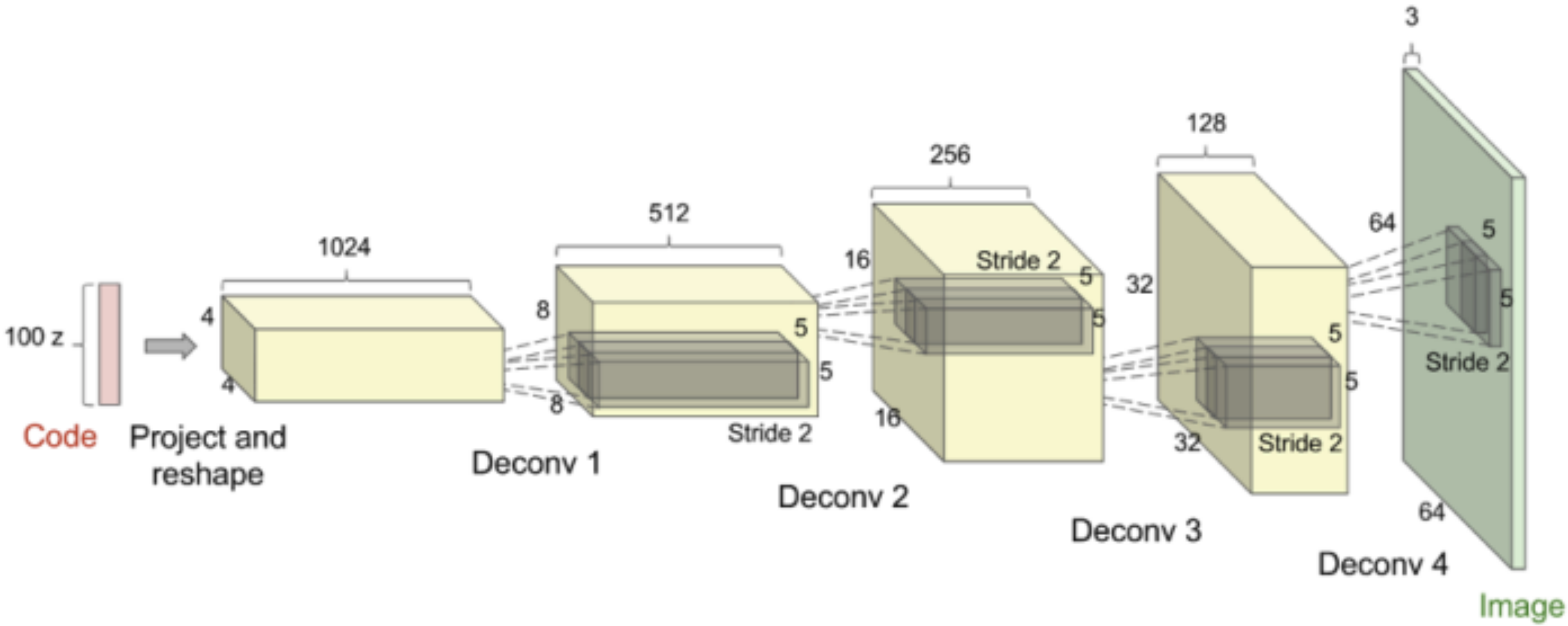
- **Deep generative model**
  - high representation power with sharp images
  - easy to sample
  - but no tractable evaluation of the density (i.e. p.d.f.)

Deep generative model

- sampling from a deep generative model, parametrized by w
  - first sample a **latent code** $z \in \mathbb{R}^k$ of small dimension $k \ll d$, from a simple distribution like standard Gaussian $N(0, \mathbf{I}_{k \times k})$
  - pass the code through a neural network of your choice, with parameter w
  - the output sample $x \in \mathbb{R}^d$ is the sample of this deep generative model



Deep generative model using deep deconvolutional layers



Generative model

- a task of importance in unsupervised learning is fitting a generative model so that we can sample from it
- classically, if we fit a parametric model like mixture of Gaussians, we write the likelihood function explicitly in terms of the model parameters, and maximize it using some algorithms
- $$\text{maximize}_w \sum_{i=1}^{n} \log \Big(\underbrace{P_w(x_i)}_{\text{p.d.f.}} \Big)$$
- deep generative models use neural networks, but the likelihood of deep generative models cannot be evaluated easily, so we use alternative methods

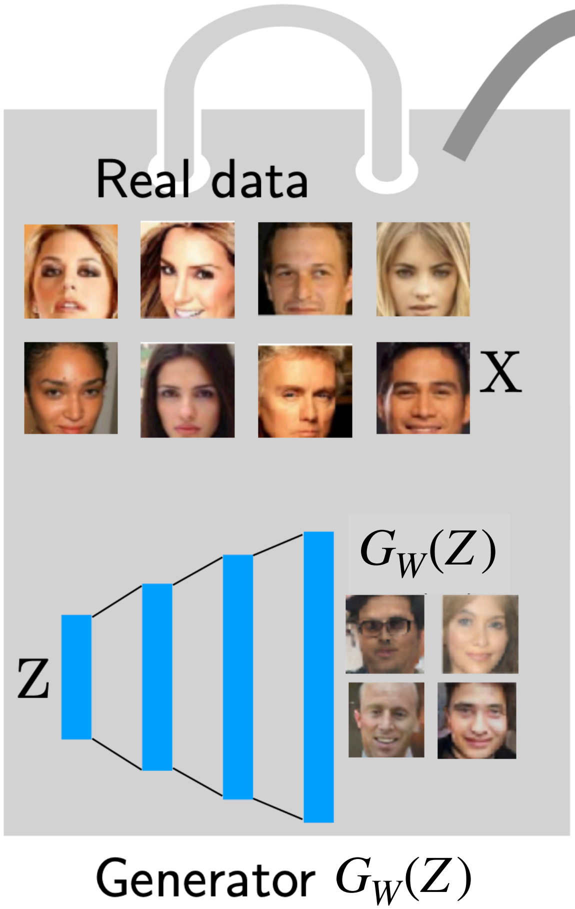
Goal

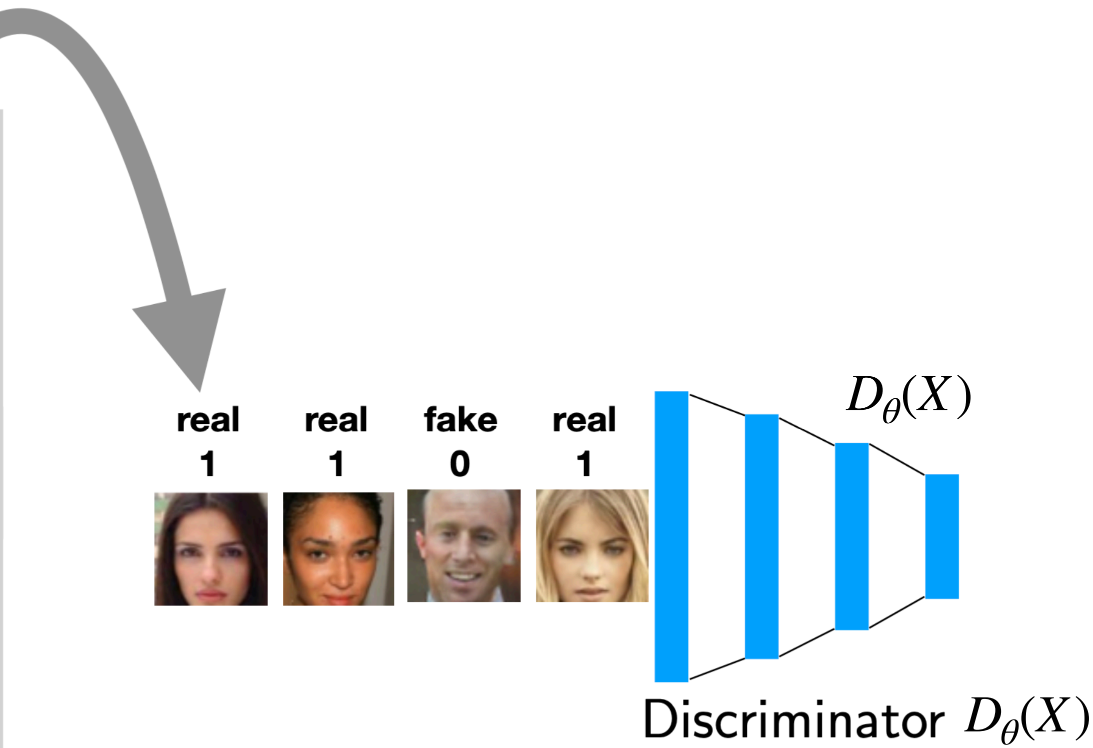
- Given examples $\{x_i\}_{i=1}^n$ coming i.i.d from an unknown distribution $P(x)$, train a generative model that can generate samples from a distribution close to $P(x)$

These are computer generated images from the "bigGAN".

- Any idea how to train $f_w(Z)$ for Gaussian Z, so that it is close to the samples $\{x_i\}_{i=1}^n$?

Adversarial training: a new way to train a deep generative model





- The discriminator tries to identify which is real and which is fake
- The generator tries to fool the discriminator

$$\min_W \max_\theta \; V(G_W, D_\theta)$$

“Generative Adversarial Nets”, Goodfellow et al.

Adversarial training

- Classification by a discriminator
  - Consider the example of SPAM detection
  - Each sample x_i is an email
  - Distribution of **true email** is $P(x)$
  - Suppose spammers generate **spams** with distribution $Q(x)$
  - Training a Spam detector: Typical classification task
    - Generate samples from true emails and label them $y_i = 1$
    - Generate samples from spams and label them $y_i = 0$
    - Using these as training data, train a classifier that outputs

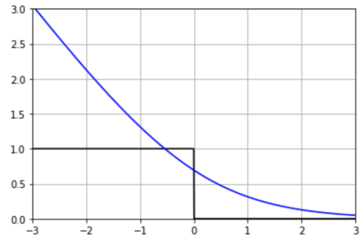
$$\mathbb{P}(y_i = 1 \,|\, x_i) \simeq \frac{1}{1 + e^{-f_\theta(x)}}$$

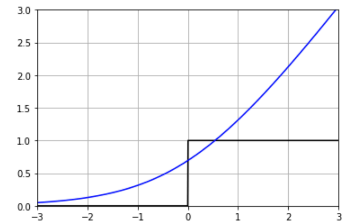
for some neural network $f_\theta(\,\cdot\,)$ with parameter θ
(this is the **logistic regression** for a binary classification)

- Applying **logistic regression**, we want to solve

$$\max_{\theta} \sum_{i:y_i=1} \log\left(\frac{1}{1+e^{-f_\theta(x_i)}}\right) + \sum_{i:y_i=0} \log\left(1 - \frac{1}{1+e^{-f_\theta(x_i)}}\right)$$

$$\min_{\theta} \log(1+e^{-f_\theta(x_i)})$$





$$\min_{\theta} \log(1+e^{f_\theta(x_i)})$$

$$= \log\left(\frac{1}{1 - \frac{1}{1+e^{-f_\theta(x_i)}}}\right)$$

- in **adversarial training**, it is customary to write

$$\frac{1}{1+e^{-f_\theta(x)}} = D_\theta(x),$$ which is called a **discriminator**

- and find the "best" discriminator by solving for

$$\max_{\theta} \mathscr{L}(\theta) = \sum_{x_i \sim P_{\text{real}}(\cdot)} \log D_\theta(x_i) + \sum_{x_i \sim Q_{\text{gen}}(\cdot)} \log(1 - D_\theta(x_i))$$

as 1 labelled examples come from real distribution $P_{\text{real}}(\,\cdot\,)$
and 0 labelled examples come from spam distribution $Q_{\text{gen}}(\,\cdot\,)$

Adversarial training

- Suppose now that the **spam detector (i.e. the discriminator)** is fixed, then the spammer's job is to generate spams that can fool the detector by making the likelihood of the "spams being classified as spams" **small**:

$$\min_{Q_{\text{gen}}(\cdot)} \mathscr{L}(\theta) = \underbrace{\sum_{x_i \sim P_{\text{real}}(\cdot)} \log D_\theta(x_i)}_{\text{does not depend on } Q_{\text{gen}}(\cdot)} + \sum_{x_i \sim Q_{\text{gen}}(\cdot)} \log(1 - D_\theta(x_i))$$

- where 0 labelled examples are coming from the distribution $Q_{\text{gen}}(\cdot)$, which is modeled by a **deep neural network generative model,** i.e.

$x_i = G_w(z_i)$, where $z_i \sim N(0, \mathbf{I}_{k \times k})$

- The minimization can be solved by finding. The "best" generative model that can fool the discriminator

$$\min_{w} \mathscr{L}(w, \theta) = \underbrace{\sum_{x_i \sim P(\cdot)} \log D_\theta(x_i)}_{\text{does not depend on } w} + \sum_{z_i \sim \mathscr{N}(0, \mathbf{I}_{k \times k})} \log\Big(1 - D_\theta\big(G_w(z_i)\big)\Big)$$

Adversarial training

- Now we have a game between the spammer and the spam detector:

$$\min_{w} \max_{\theta} \sum_{x_i \sim P(\cdot)} \log D_\theta(x_i) + \sum_{z_i \sim N(0,\mathbf{I})} \log(1 - D_\theta(G_W(z_i)))$$

- Where $P(\cdot)$ is the distribution of real data (true emails), and $f_W(z_i) \sim Q(\cdot)$ is the distribution of the generated data (spams) that we want to train with a **deep generative model**
- jointly training the discriminator and the generator is called **adversarial training**
- Alternating method is used to find a solution of this non-convex minimax optimization

Alternating gradient descent for adversarial training

- Gradient update for the **discriminator** (for fixed generator w)

$$\max_{\theta} \sum_{x_i \sim P(\cdot)} \log D_\theta(x_i) + \sum_{x_i \sim Q(\cdot)} \log(1 - D_\theta(x_i))$$

- First sample n examples from real data (in the training set) and the generator data $x_i \sim G_w(z_i)$
(for the current iterate of the generator weight w)
- compute the gradient for those $2n$ samples using back-propagation
- Update the discriminator weight θ by adding the gradient with a choice of a step size

$$\theta \leftarrow \theta + \eta \nabla \mathscr{L}(w, \theta)$$

Alternating gradient descent for adversarial training

- gradient update for the **generator** (for fixed discriminator θ)

$$\min_{w} \sum_{x_i \sim P(\cdot)} \log D_\theta(x_i) + \sum_{z_i \sim N(0,\mathbf{I})} \log(1 - D_\theta(G_w(z_i)))$$

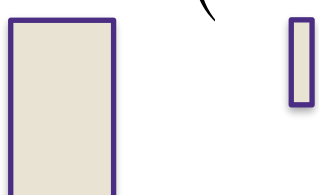
- Consider the gradient update on a single sample

$$\min_{w} \mathscr{L}(w, z_i) \;=\; \log(1 - D_\theta(G_w(z_i)))$$

for a single $z_i \sim N(0,\mathbf{I})$ sampled from a Gaussian

- The gradient update is

$$\begin{aligned} w \;&=\; w - \eta \nabla_w \mathscr{L}(w, z_i) \\ &=\; w - \eta \, \nabla_w G_w(z_i) \left(\nabla_x D_\theta(x) \, \frac{-1}{1 - D_\theta(x)} \right) \end{aligned}$$



by the chain rule with $x = G_w(z_i)$

Not only is GAN amazing in generating realistic samples

http://whichfaceisreal.com

It opens new doors to exciting applications

- Cycle-GAN

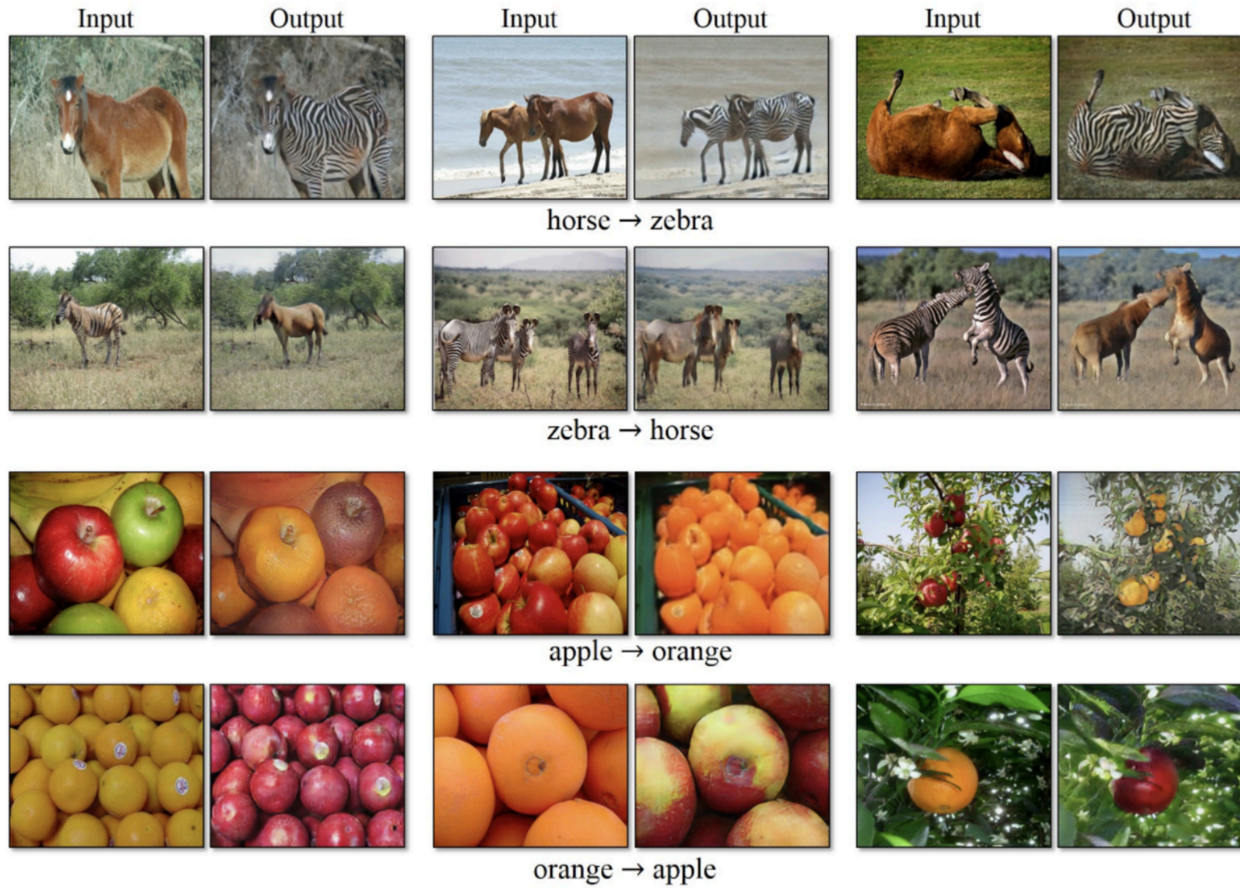


Figure 3: Street scene image translation results. For each pair, left is input and right is the translated image.

Any idea how to do this?

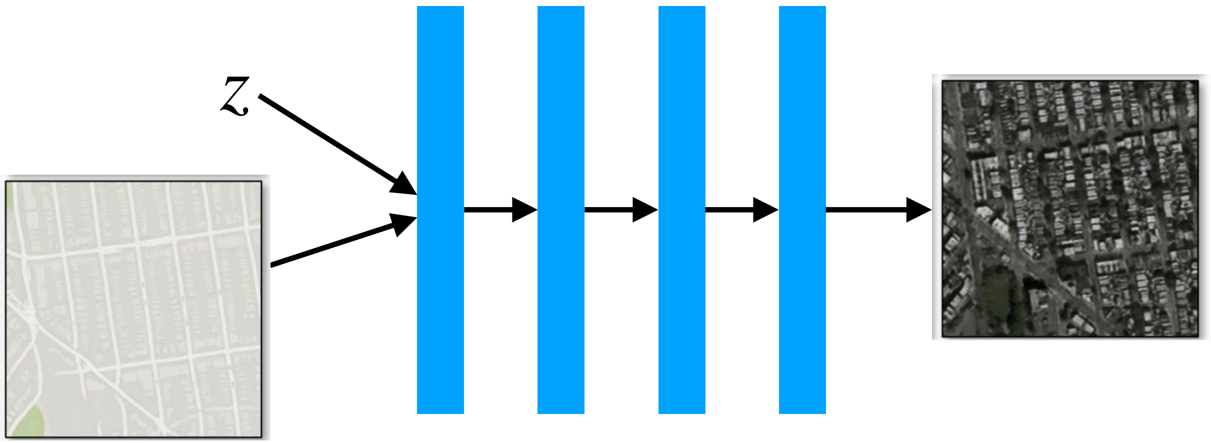
https://www.youtube.com/watch?v=PCBTZh41Ris

Style transfer with generative model

- If we have paired training data,

- And want to train a generative model G(x,z)=y,

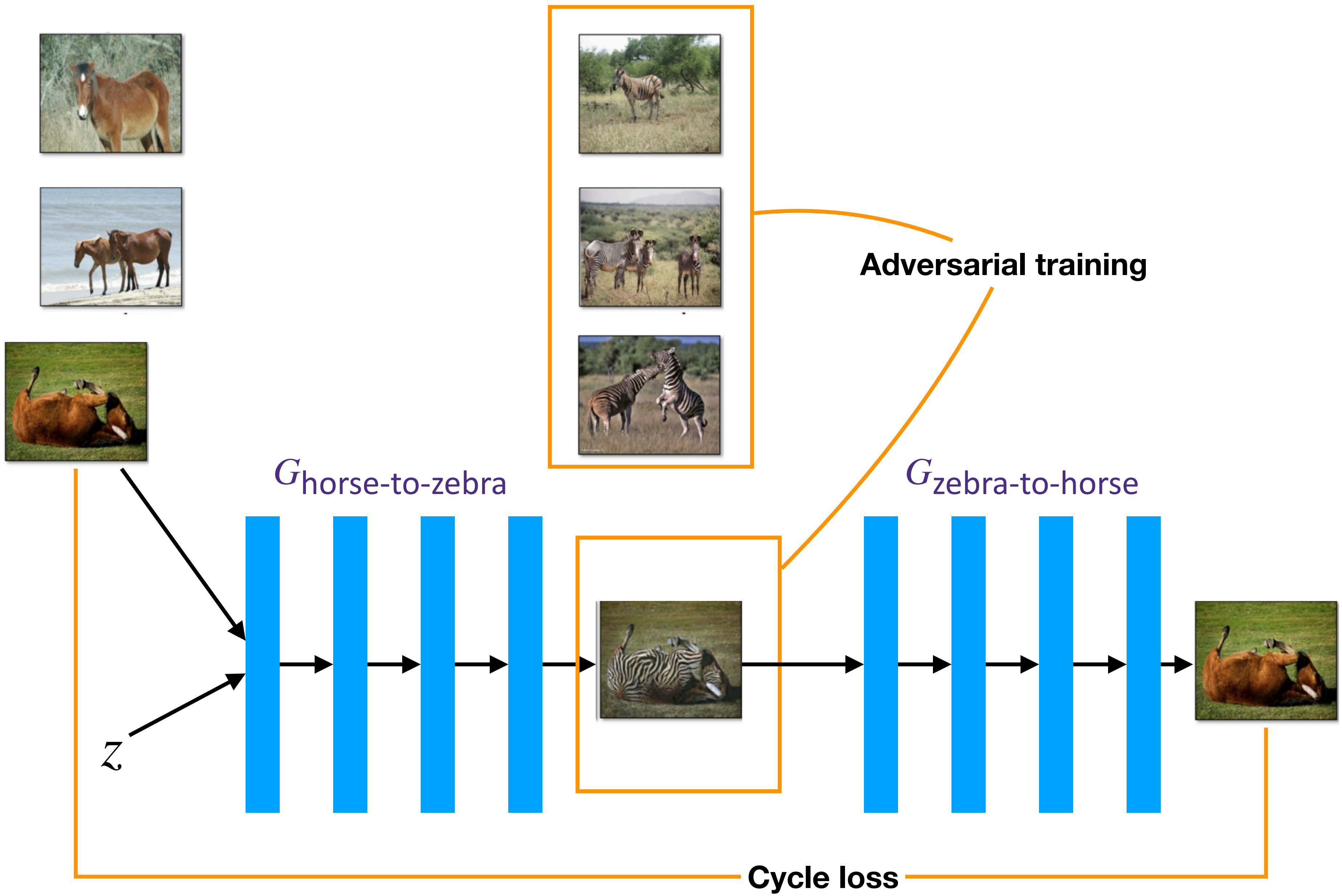
- This can be posed as a regression problem



- What do we do when we do not have paired data?

How do we do style transfer without paired data? Cycle-GAN

How do we do style transfer without paired data? Cycle-GAN



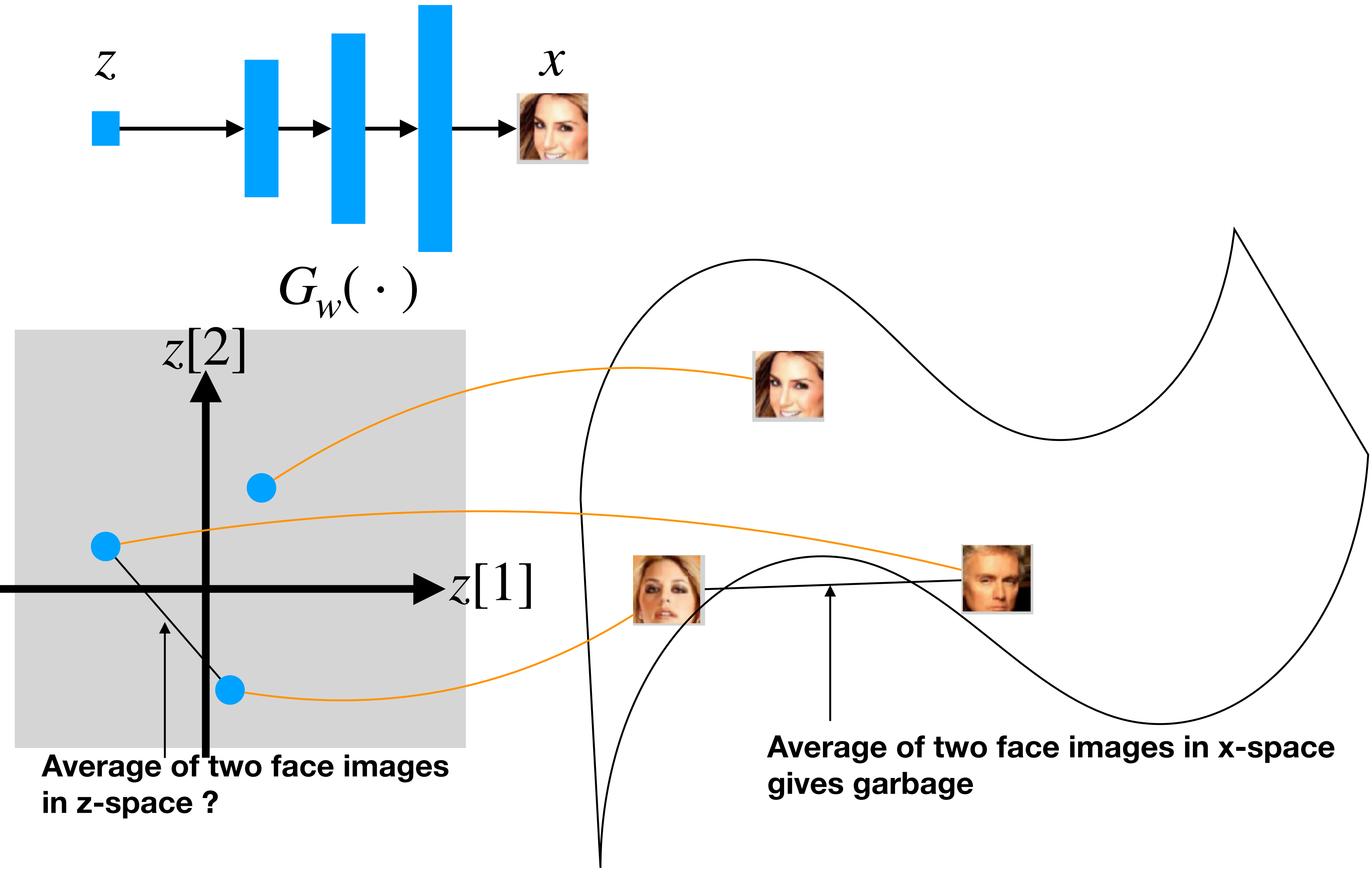
Super resolution

Low resolution image

Estimated
high resolution image

True
high resolution image

The learned latent space is important

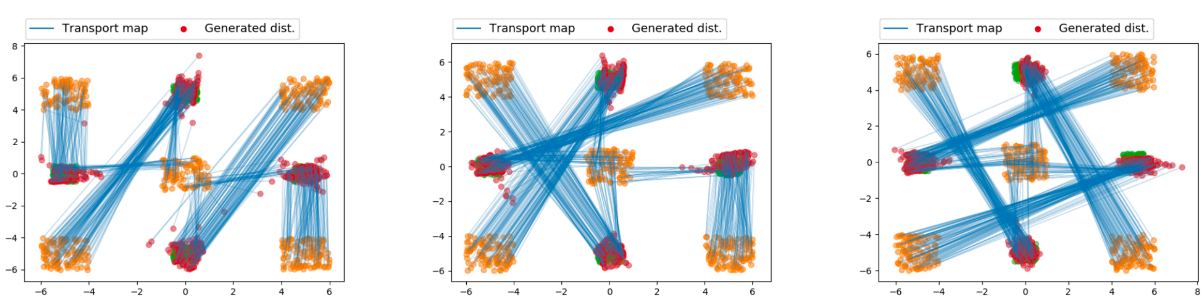


How do we check if we found the right manifold (of faces)?

- **latent traversal**

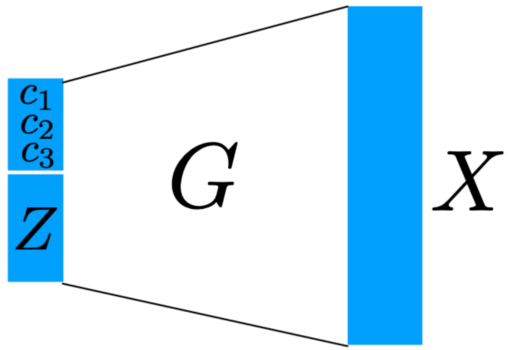
Can we make the relation between the latent space and the image space more meaningful?

- **Disentangling**
    - **GANs learn arbitrary mapping from z to x**
    - **As the loss only depends on the marginal distribution of x and not the conditional distribution of x given z (how z is mapped to x)**



Disentangling seeks meaningful mapping from z to x

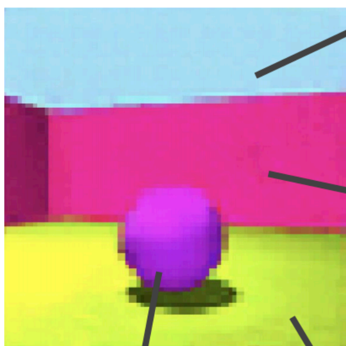
- there is no formal (mathematical) universally agreed upon definition of disentangling



$\Rightarrow$ change c_1 $\Rightarrow$ change c_2 $\Rightarrow$ change c_3

- informally, we seek latent codes that
  - are "informative" or make "noticeable" changes
  - are "uncorrelated" or make "distinct" changes

Decompose data into a set of underlying **human-interpretable** factors of variation



Explainable models

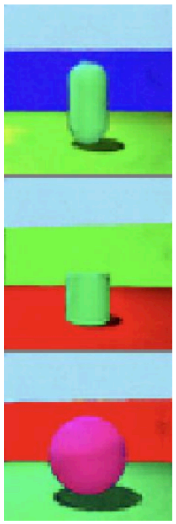
What is in the scene?

Controllable generation

Generate a red ball instead

Fully-supervised case

Strategy: Label everything



c_1 c_2 c_3

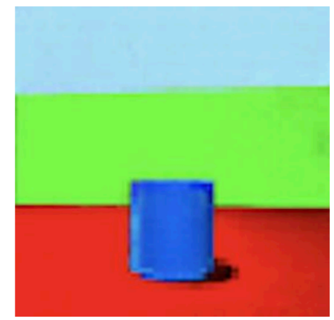
{dark blue wall, green floor, green oval}

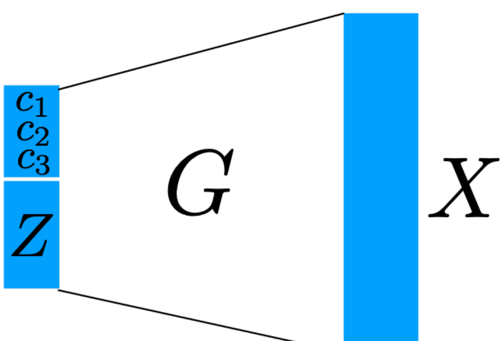
{green wall, red floor, green cylinder}

{red wall, green floor, pink ball}

Controllable generation as **label-conditional generative modeling**

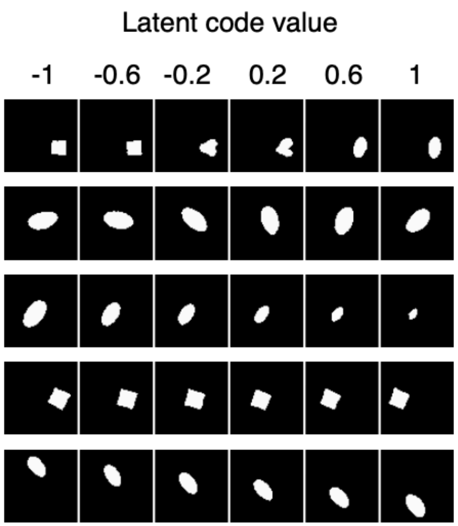
green wall, red floor, blue cylinder

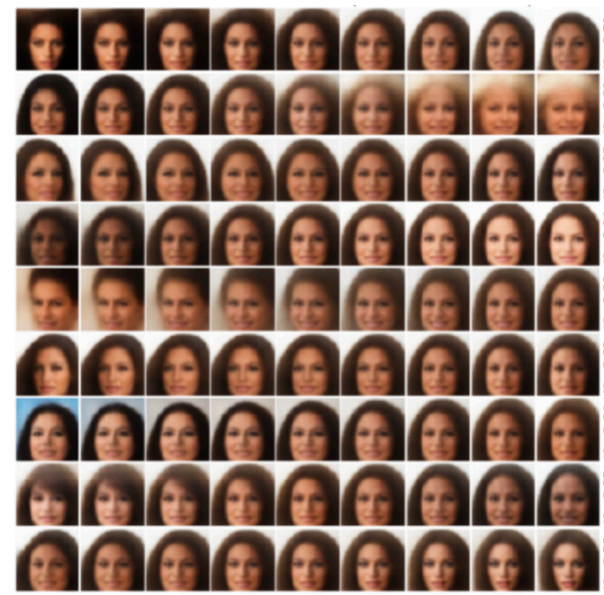


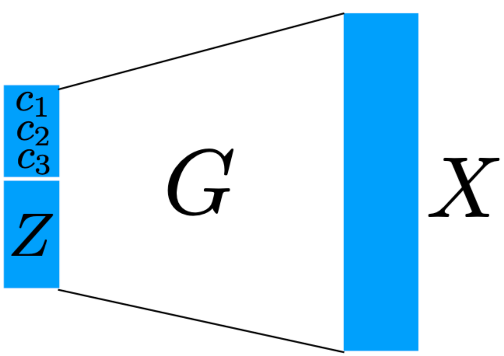


Train a **conditional GAN**, where (c_1, c_2, c_3) is a numerical representation of the **labels** given in the training data, and z is drawn from Gaussian

Unsupervised training of Disentangled GAN

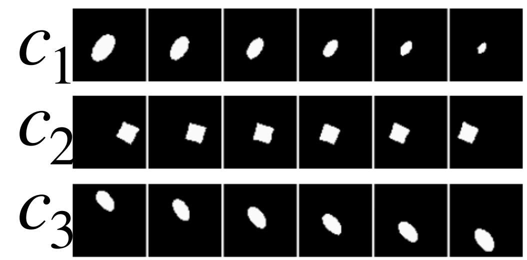






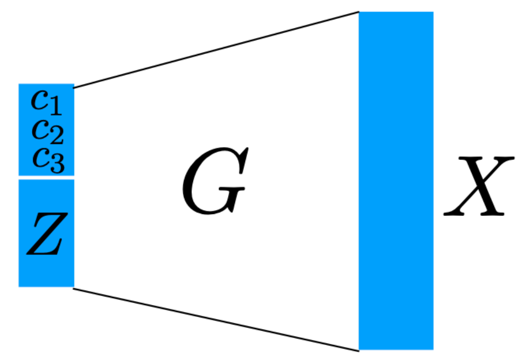
Disentangled GAN training: InfoGAN-CR, 2019

- 1. As in standard GAN training, we want $G_w(z)$ to look like training data (which is achieved by adversarial loss provided by a discriminator)



D( **) = {real,fake}**

- 2. We also want the controllable latent code c to be predictable from the image
  - add a NN regressor that predicts $\hat{c}(x)$, and train the generator that makes the prediction accuracy high (note that both this predictor and the generator works to make the prediction accurate, unlike adversarial loss)



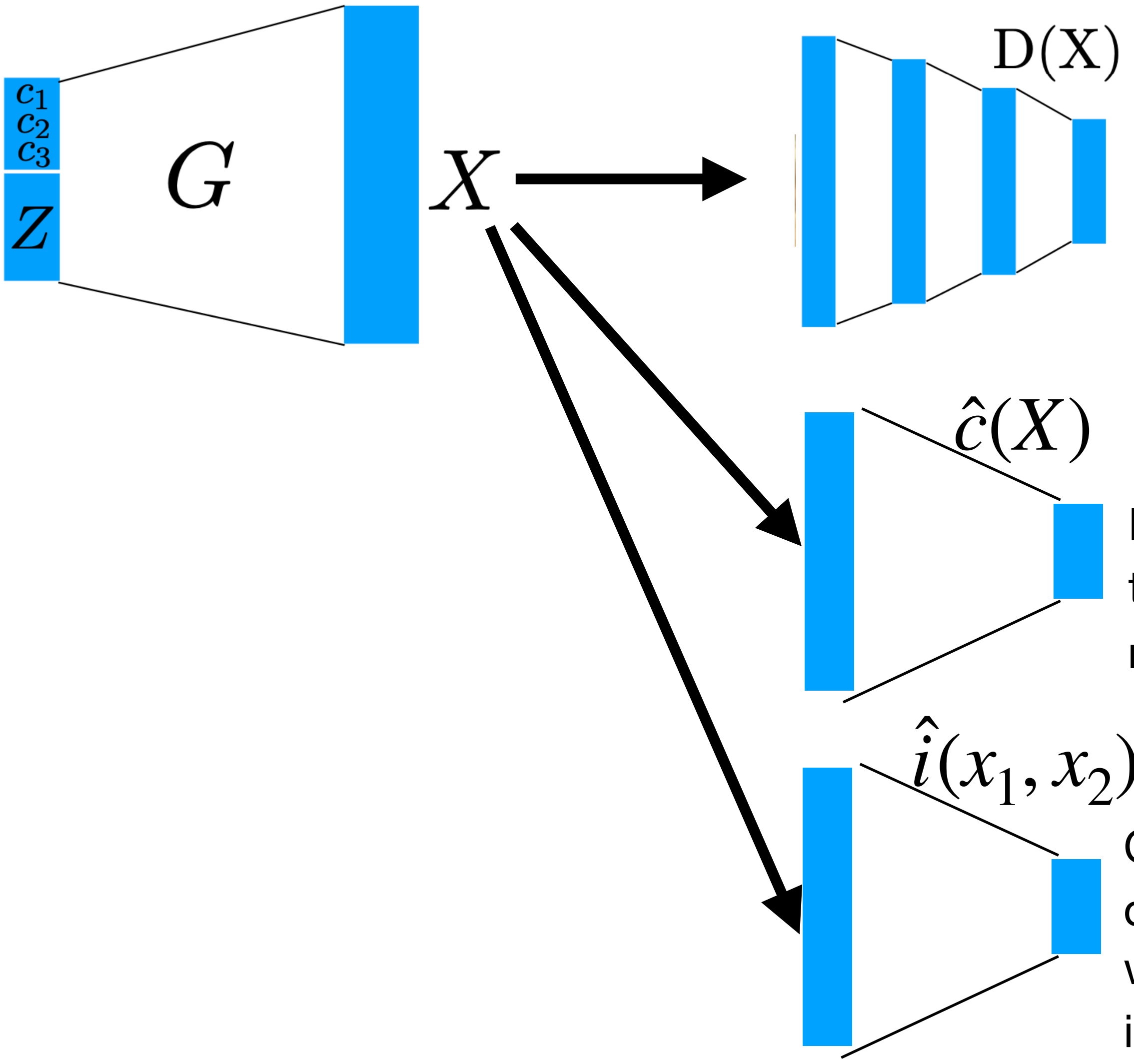
$$\text{minimize} \quad \| \hat{c}(\quad) - c \|^2$$

- 3. We also want each code to control distinct properties
  - add a NN that predicts which code was changed

$$\hat{i}(\quad) \simeq i$$

Disentangling with contrastive regularizer

- To train a disentangled GAN, we use contrastive regularizer



Discriminator encourages output x to be **realistic**

Predictor makes sure that the changes in c make **noticeable** changes in x

Contrastive regularizer detects which latent code c_i was the same/different in a paired (x_1, x_2)

Questions?