Computer Visions
Generative Machine Learning Tools

MAS.S68 F’19
Roy Shilkrot, Fluid Interfaces

Class 2: GANs

Today

Catch up

- HW0, Logistics and Tooling
- Convolutions

Content

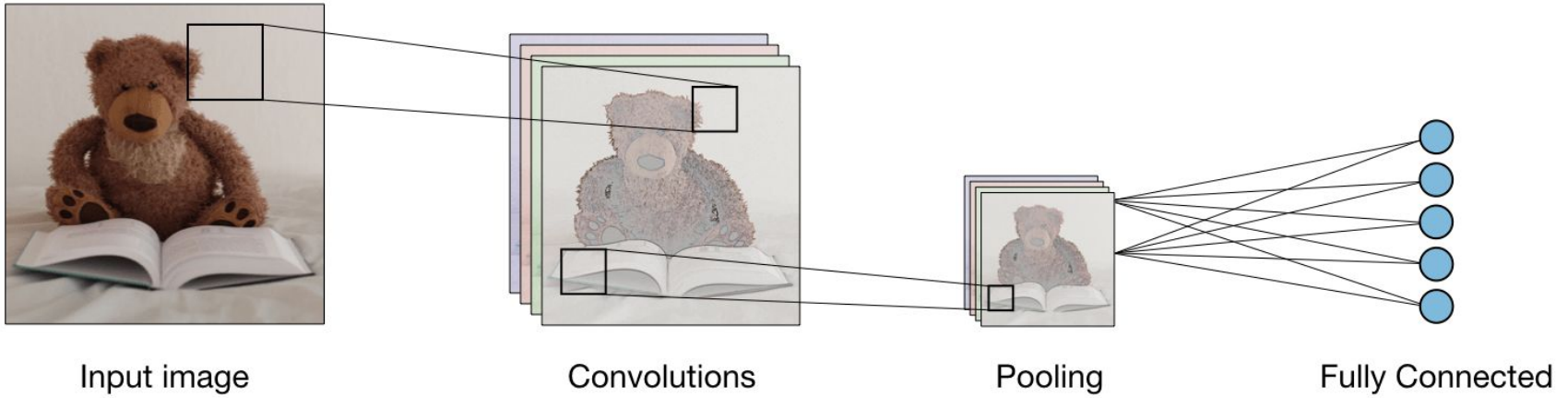
- Generative Models and Unsupervised Learning
- Variational Autoencoders
- Generative Adversarial Networks

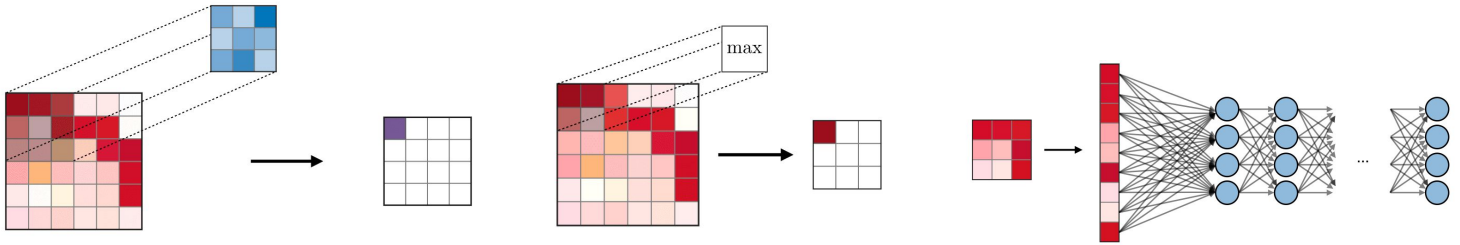
Application

- VAE, DCGAN, C GAN / InfoGAN
- BigGAN, ProGAN
- pix2pix
- CycleGAN

Tooling and HW0

Convolutional Networks





Generative Models and Unsupervised Learning

Supervised vs Unsupervised Learning

Clustering

Dimension Reduction

Generative Models

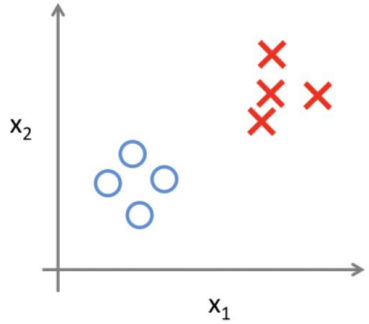
Supervised vs. Unsupervised Learning

Supervised Learning

Given data X, predict output Y

Example: Classification, Regression, Detection, Segmentation

Requires: *Annotated* data

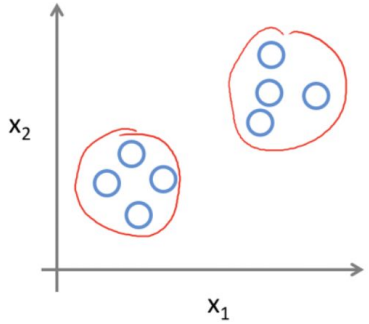


Unsupervised Learning

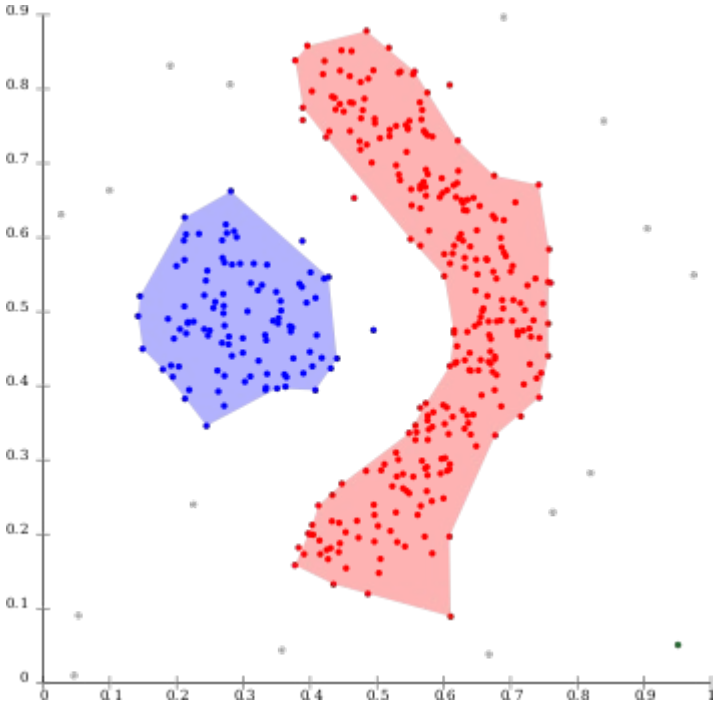
Given data X, uncover underlying (“**latent**”) structure

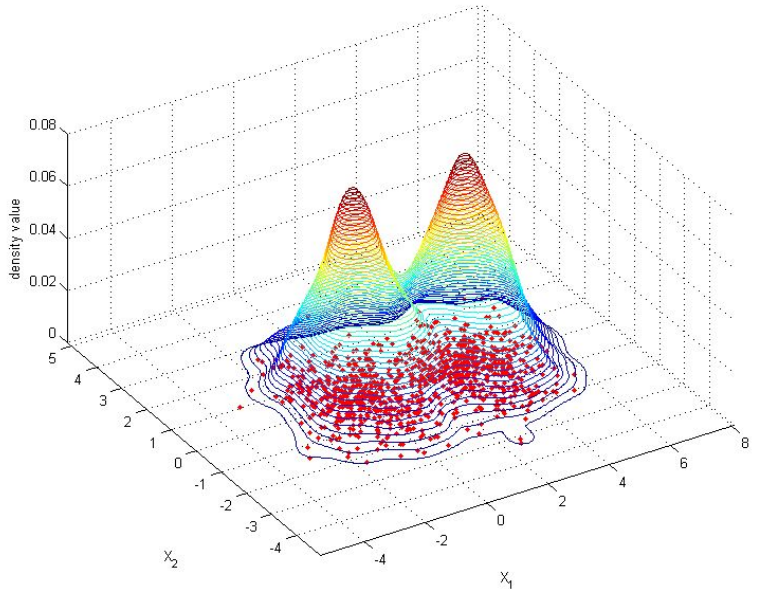
Example: Clustering / Density, Compression, Anomaly detection

Requires: Data.



Clustering and Density Estimation



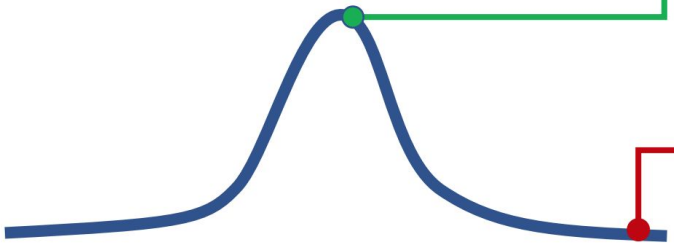


Why is it important?

- **Problem:** How can we detect when we encounter something new or rare?
- **Strategy:** Leverage generative models, detect outliers in the distribution
- Use outliers during training to improve even more!

95% of Driving Data:
(1) sunny, (2) highway, (3) straight road

Detect outliers to avoid unpredictable behavior when training

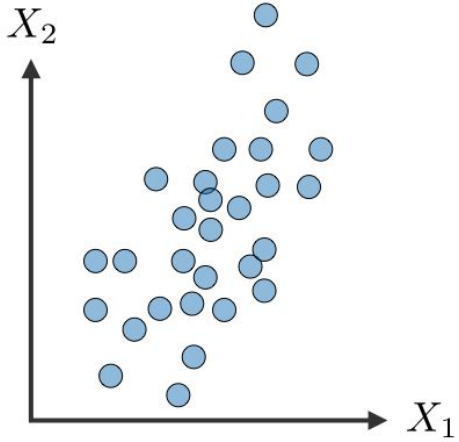


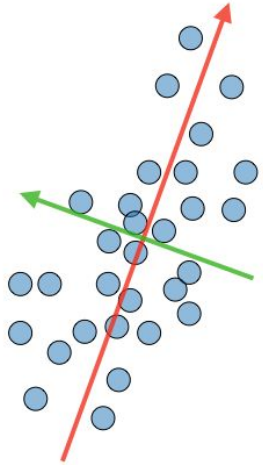
Edge Cases

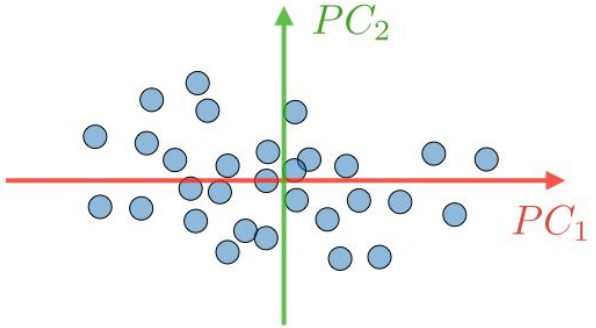
Harsh Weather

Pedestrians

Dimensionality Reduction





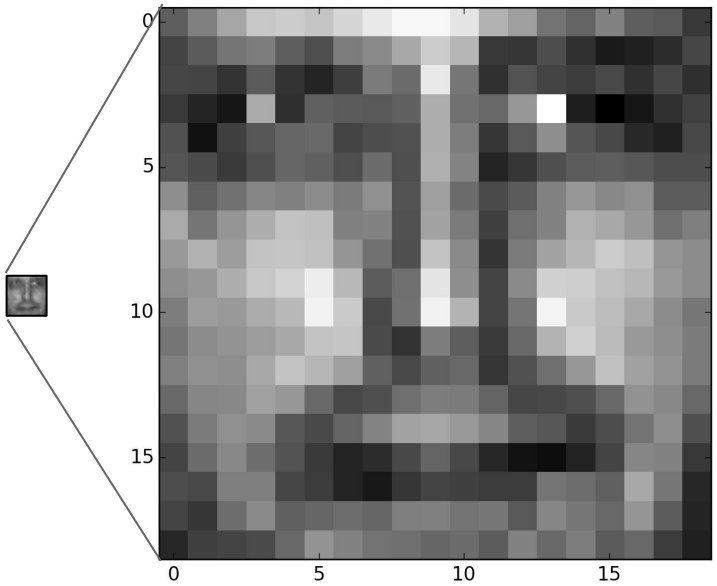


Data in feature space → Find principal components → Data in **p**rincipal **c**omponents space

Compact representation for the data using latent variables

Why is it important?

Keep in mind, our images are **high** dimension:



19x19 pixels = 361D vector (flat)

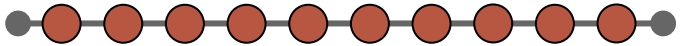
From a sampling viewpoint:

- As the number of dimension (linearly) grows the number of samples to cover the space grows exponentially!

We want to guarantee to always find a sample 0.1 away from our query.

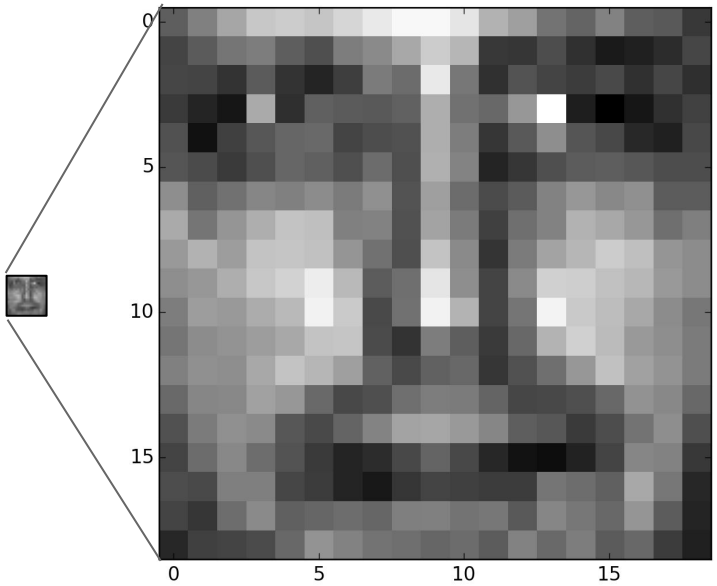
1D: we need 10 samples to cover the space [0,1].

From a k-NN perspective we can guarantee we will get a sample that's 0.1 away.



Curse of Dimensionality

Keep in mind, our images are high dimension:

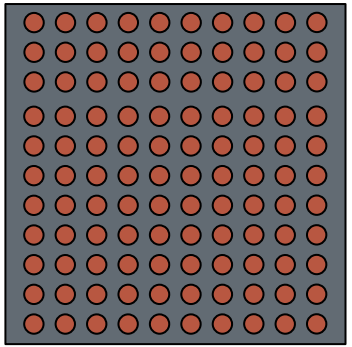


19x19 pixels = 361D vector (flat)

From a sampling viewpoint:

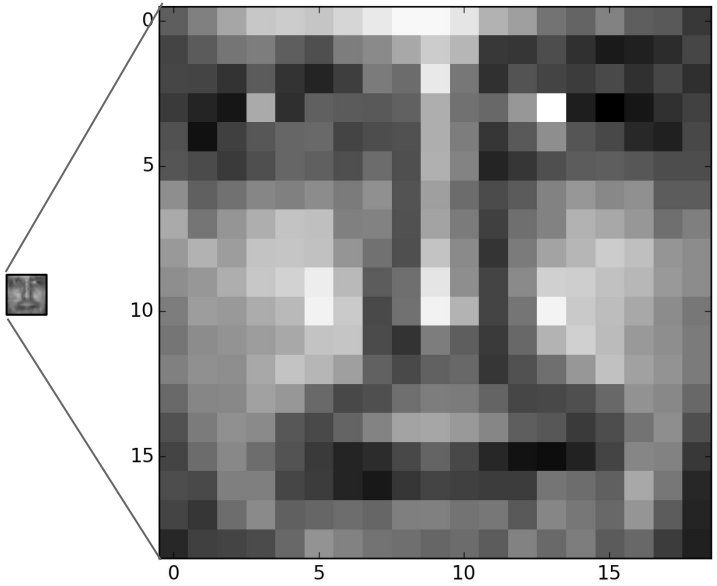
- As the number of dimension (linearly) grows the number of samples to cover the space grows exponentially!

2D: now we need **100** (10^2) samples to cover with a 0.1 margin guarantee.



Curse of Dimensionality

Keep in mind, our images are high dimension:

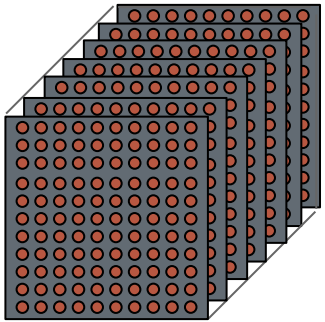


19x19 pixels = 361D vector (flat)

From a sampling viewpoint:

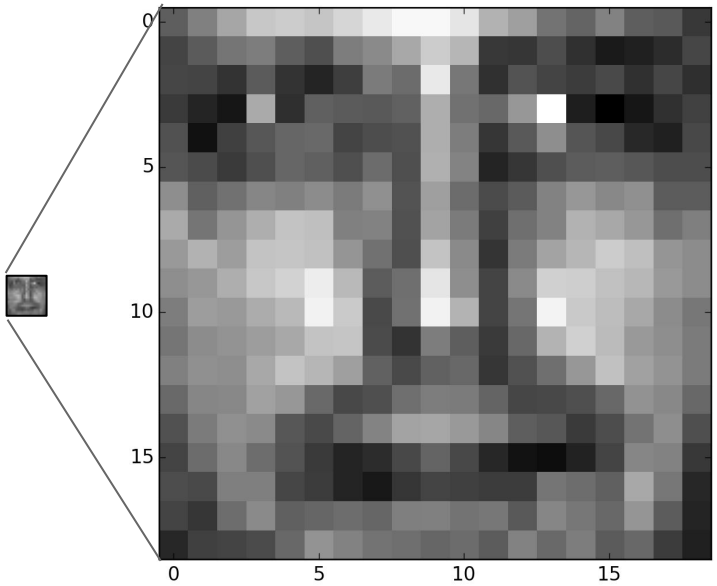
- As the number of dimension (linearly) grows the number of samples to cover the space grows exponentially!

3D: now we need **1000** (10^3) samples to cover with a 0.1 margin.



Curse of Dimensionality

Keep in mind, our images are high dimension:

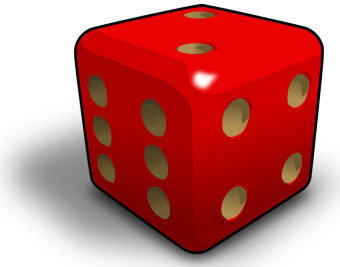


19x19 pixels = 361D vector (flat)

From a combinatorics viewpoint:

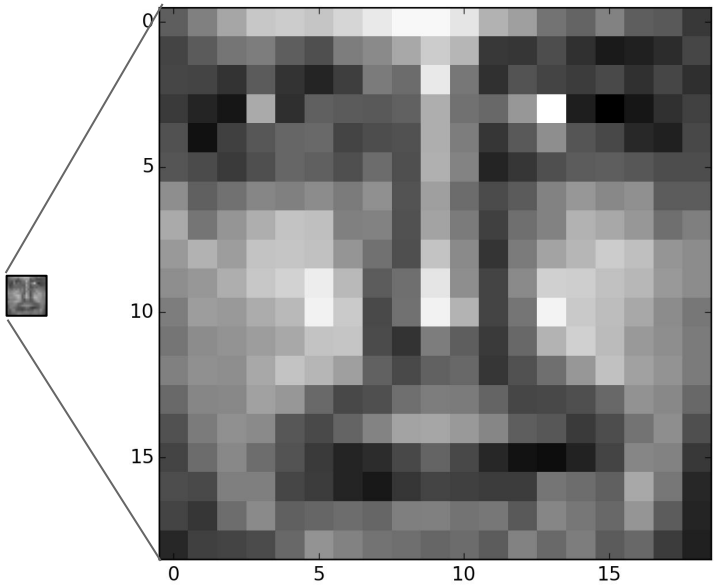
- As the number of dimensions grows, the number of possibilities grows exponentially!

We have a single cube (i.e. 1D), we have 6 possibilities (and thus probability) for an outcome.



Curse of Dimensionality

Keep in mind, our images are high dimension:

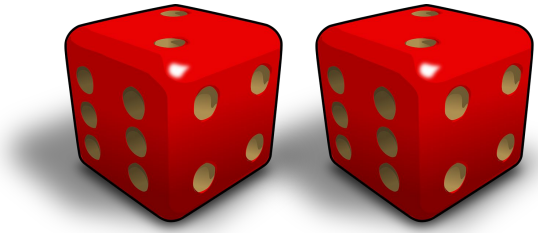


19x19 pixels = 361D vector (flat)

From a combinatorics viewpoint:

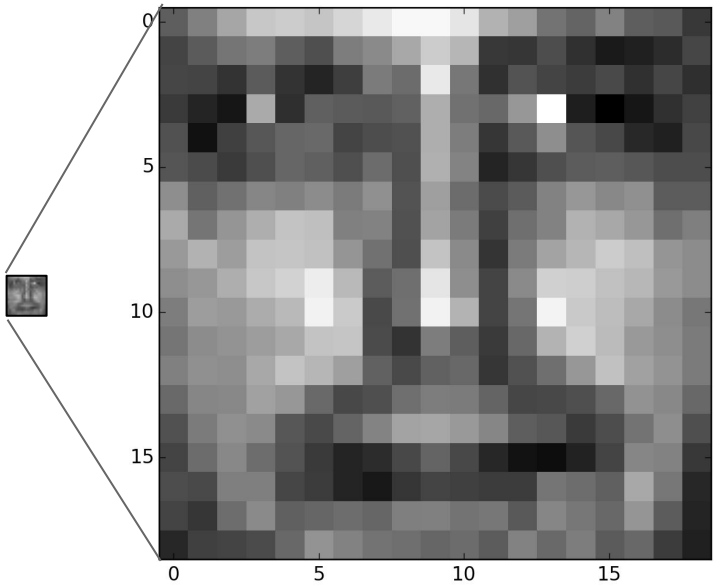
- As the number of dimensions grows, the number of possibilities grows exponentially!

If we have 2 cubes (i.e. 2D), we have **36 (6^2)** possibilities for an outcome.



Curse of Dimensionality

Keep in mind, our images are high dimension:

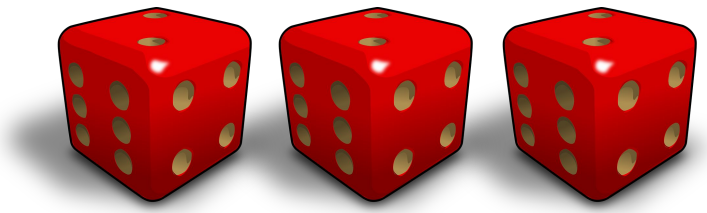


19x19 pixels = 361D vector (flat)

From a combinatorics viewpoint:

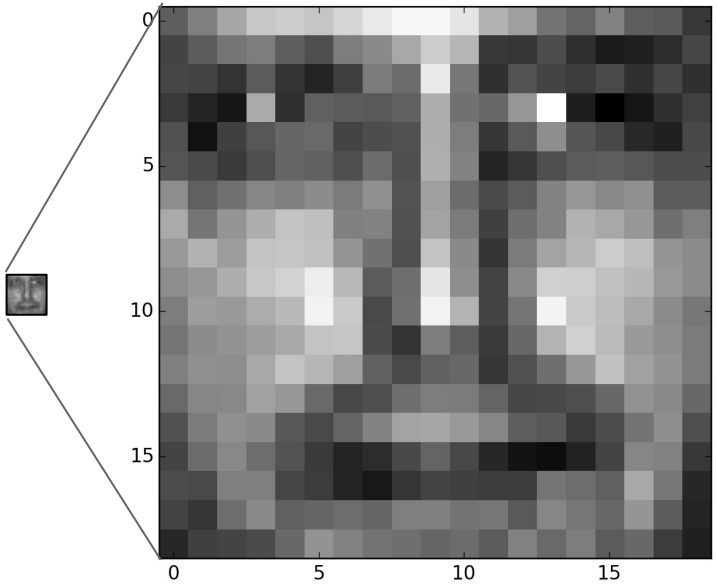
- As the number of dimensions grows, the number of possibilities grows exponentially!

If we have 3 cubes (i.e. 2D), we have **216 (6^3)** possibilities for an outcome.



Curse of Dimensionality

Keep in mind, our images are high dimension:



19x19 pixels = 361D vector (flat)

From a combinatorics viewpoint:

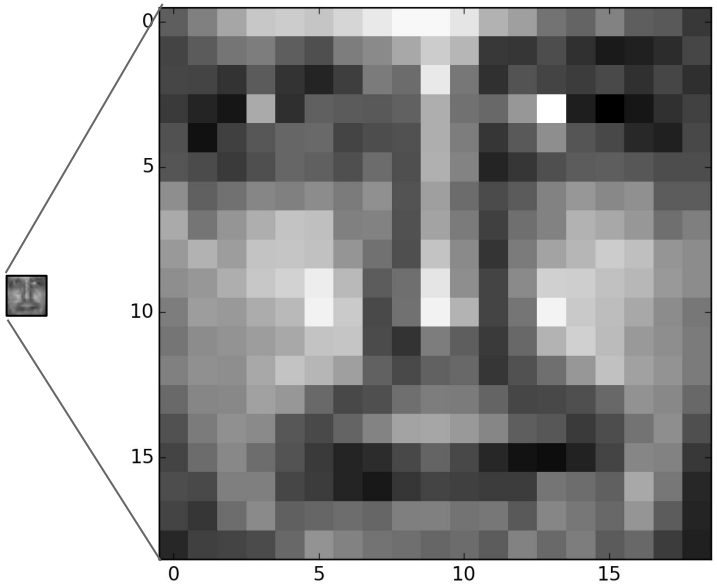
- As the number of dimensions grows, the number of possibilities grows exponentially!

And so on...

Questions

Dimensionality Reduction

Keep in mind, our images are high dimension:



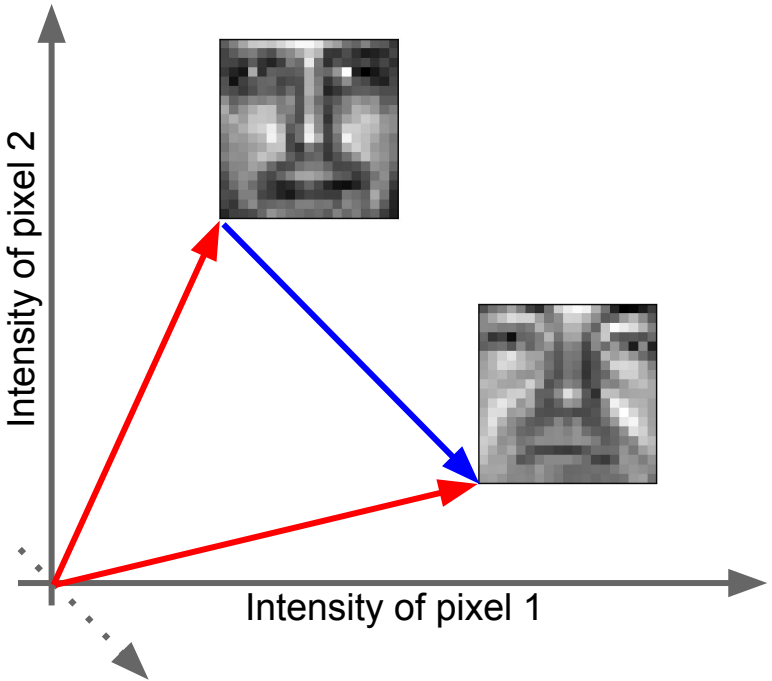
19x19 pixels = 361 vector (flat)

So we should strive to work in lower dimensions as possible.

But how can we do that?

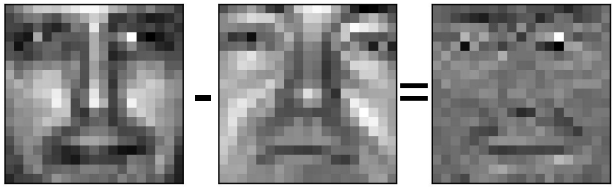
- Leverage on correlation between samples or dimensions
  - Clustering
  - PCA, LDA, ICA, etc.
  - Feature selection
  - Machine learning
- Throw away things at random (works surprisingly well)

Dimensionality Reduction



Images can be treated as high-dimension vectors.

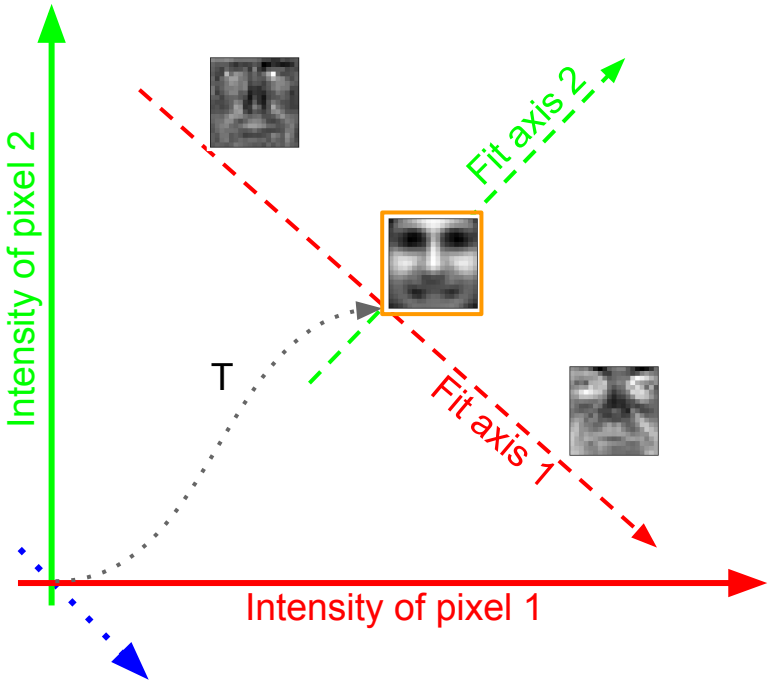
We can apply arithmetics to vectors:



Or take an average: (over all the samples)

Insight: Faces all roughly look the same.
I.e. their vectors have high correlation.

Dimensionality Reduction



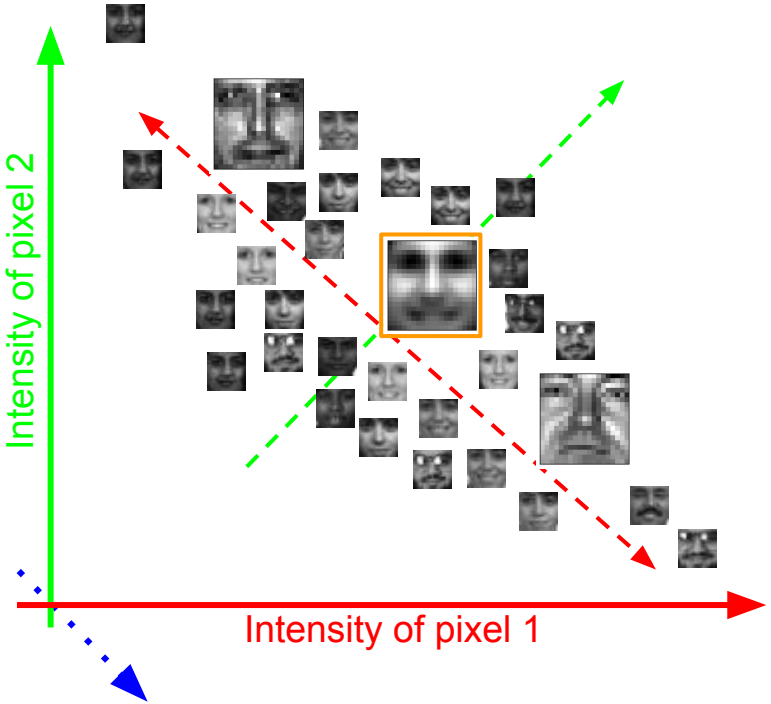
We can translate the origin to the mean and rotate the “intensity” axes to a new linear basis. (what do we call such a rigid transformation?)

We can find new axis so the vectors will lie close to an axis that better represents the data than the original “intensity” axes.

In the new basis:
Axis 2 carries much less information than Axis 1.
So we can simply drop Axis 2.

How to find a new basis that maximizes the information on certain dimensions?

Dimensionality Reduction



Look at the (co-)variance across all the samples:

$$\mathrm{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2,$$

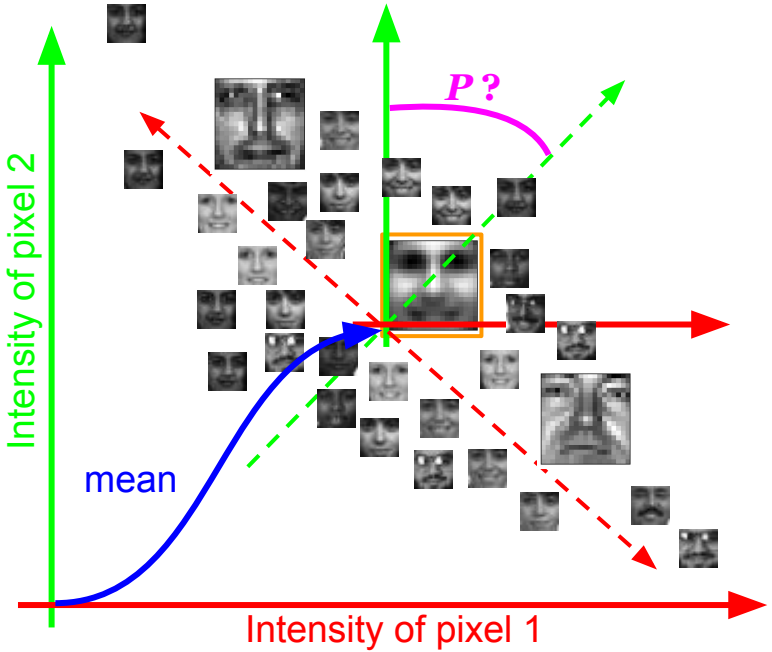
$$\operatorname{var}(\mathbf{X}) = \operatorname{cov}(\mathbf{X}) = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}])(\mathbf{X} - \mathrm{E}[\mathbf{X}])^{\mathrm{T}}\right].$$

$$\Sigma = \mathrm{E}\left[(\mathbf{X} - \mathrm{E}[\mathbf{X}])\,(\mathbf{X} - \mathrm{E}[\mathbf{X}])^{\mathrm{T}}\right] \qquad \boldsymbol{\mu} = \mathrm{E}(\mathbf{X})$$

To maintain the most information we'd like to maintain the most (i.e. maximize) variance.

Find: The vector that maximizes the variance.

Dimensionality Reduction



$$\Sigma = \mathbf{E}\left[(\mathbf{X} - \mathbf{E}[\mathbf{X}])\,(\mathbf{X} - \mathbf{E}[\mathbf{X}])^{\mathrm{T}}\right] \qquad \boldsymbol{\mu} = \mathbf{E}(\mathbf{X})$$

Let's take the mean off of all the samples:

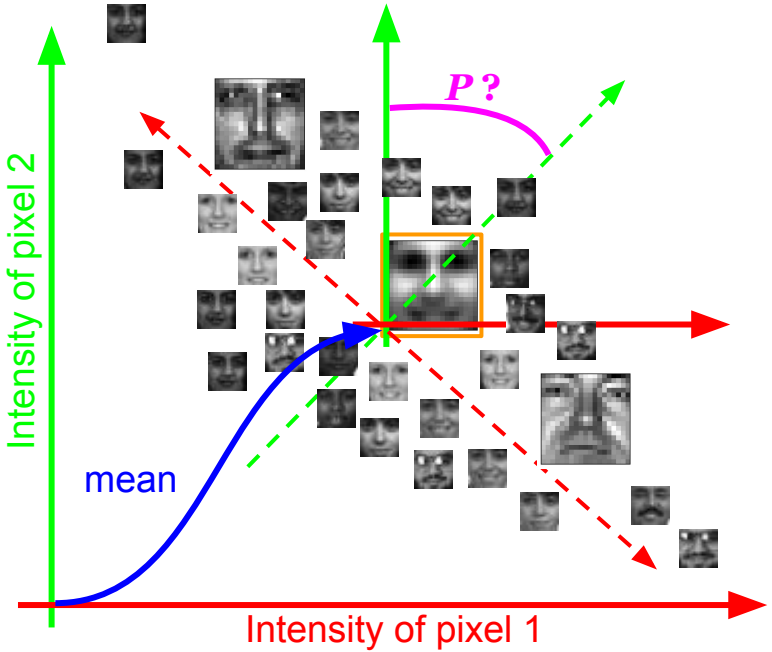
$$\Sigma = E[\hat{X}\hat{X}^T] \qquad \hat{X} = X - \mu$$

Now they have zero mean, we want to find an (orthonormal) transformation P so that $P\hat{X}$ has diagonal covariance matrix $\mathrm{cov}(PX)$ - i.e. each dimension is uncorrelated with the others.

$$\begin{aligned}\mathrm{cov}(PX) &= \mathbb{E}[PX\,(PX)^*] \\ &= \mathbb{E}[PX\,X^*P^*] \\ &= P\,\mathbb{E}[XX^*]P^* \\ &= P\ \mathrm{cov}(X)P^{-1}\end{aligned}$$

P diagonalizes $\mathrm{cov}(X)$

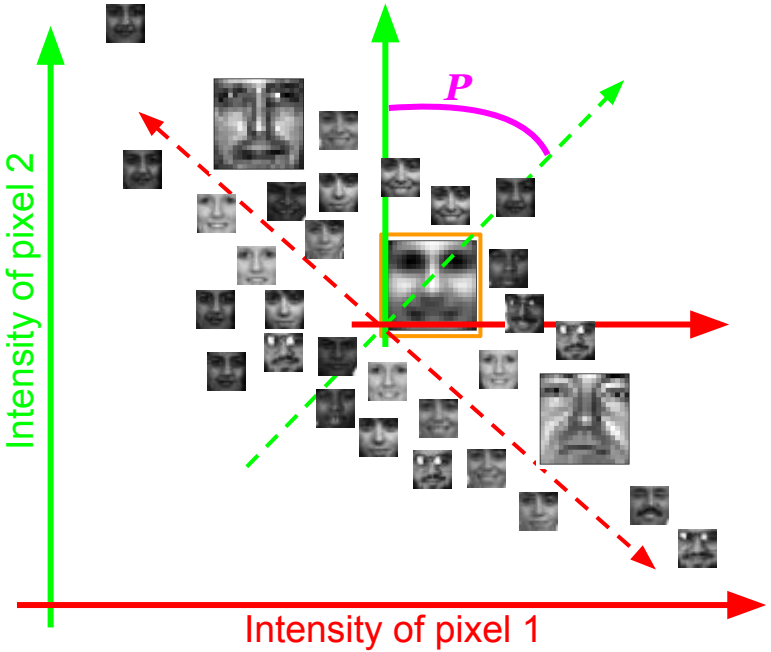
Dimensionality Reduction



We're looking for $\boldsymbol{P}$ that diagonalizes $\text{cov}(X)$.

How do we find a diagonalization of matrices?

Dimensionality Reduction



We're looking for ***P*** that diagonalizes $\mathrm{cov}(X)$.

How do we find a diagonalization of matrices?

Eigen decomposition!

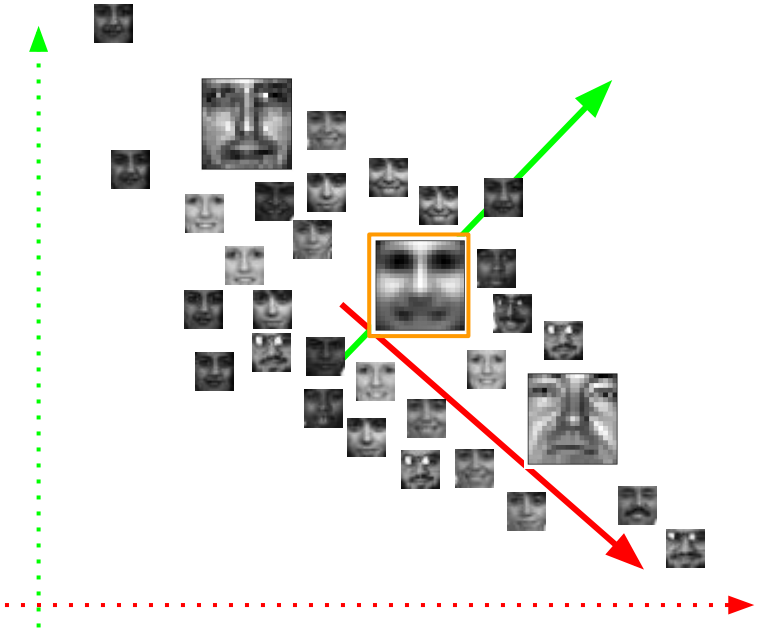
$$cov(X) = \Sigma = PDP^{-1}$$

Verify **Σ** is rectangular and positive definite

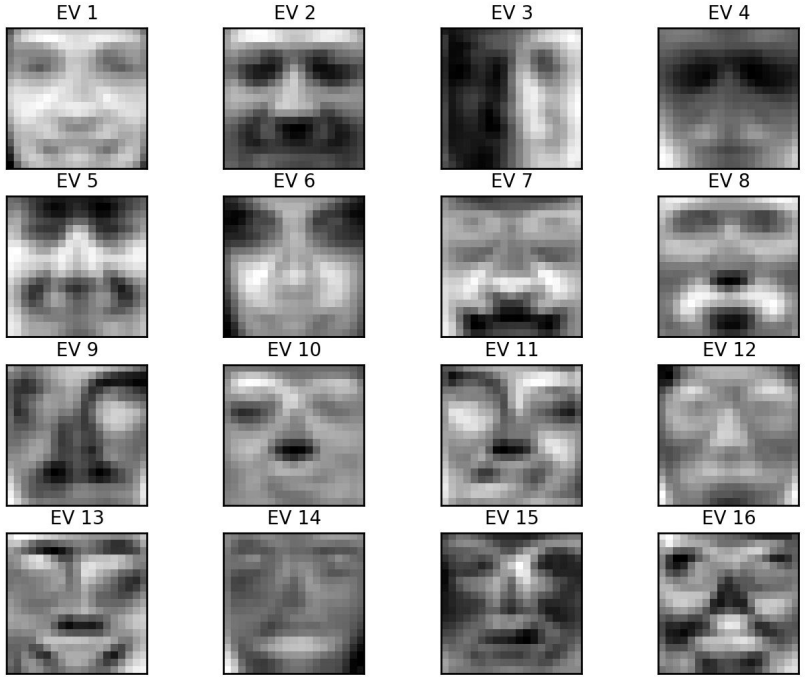
Where the columns of P are the eigenvectors and eigenvalues are on the diagonal of D.

The EVecs are called the Principal Components, and this process is called Principal Component Analysis (PCA).

Dimensionality Reduction



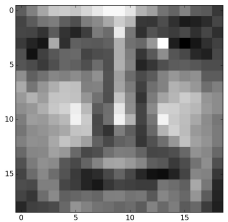
Here's how the EVecs may look like:



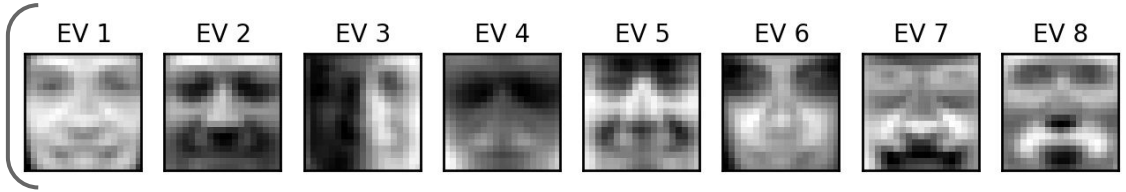
Dimensionality Reduction

Since the EVecs are a new basis for the data, each sample can be represented as a linear combination:

Use the entire new basis to reconstruct the image precisely to the original - but that defeats the purpose of reducing dimension… What can we do?



=

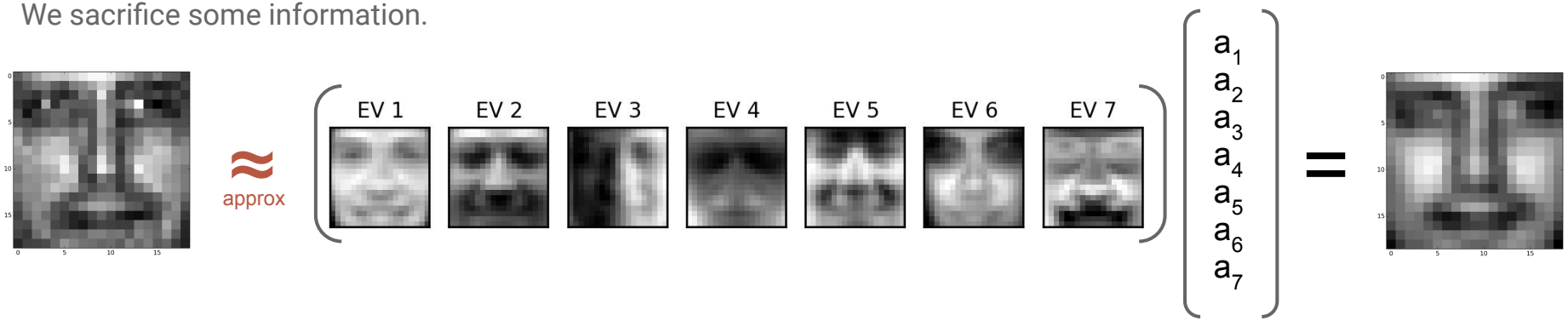


$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \\ \dots \\ \dots \\ \dots \end{pmatrix}$$

Dimensionality Reduction

We can take just part of the EVecs, the heaviest ones (according to their EVals) - reduce the dimensionality, but --

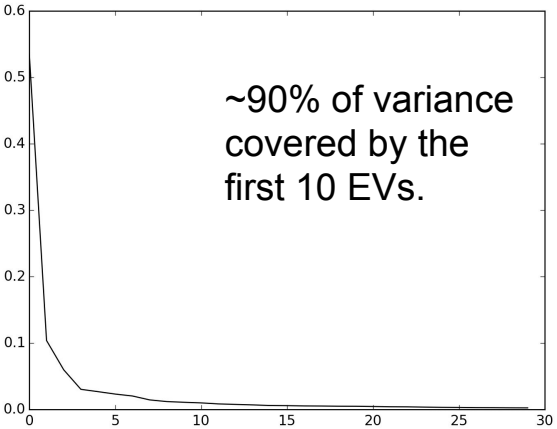
We sacrifice some information.



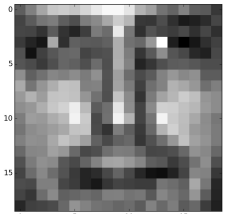
But - how many EVecs should we take for our new base?

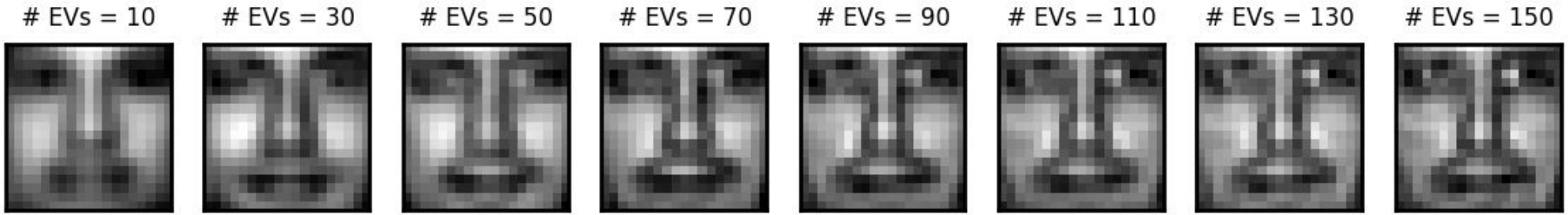
Dimensionality Reduction

How many EVecs should we take for our basis?



Original





Questions

Compact Representation: JPEG

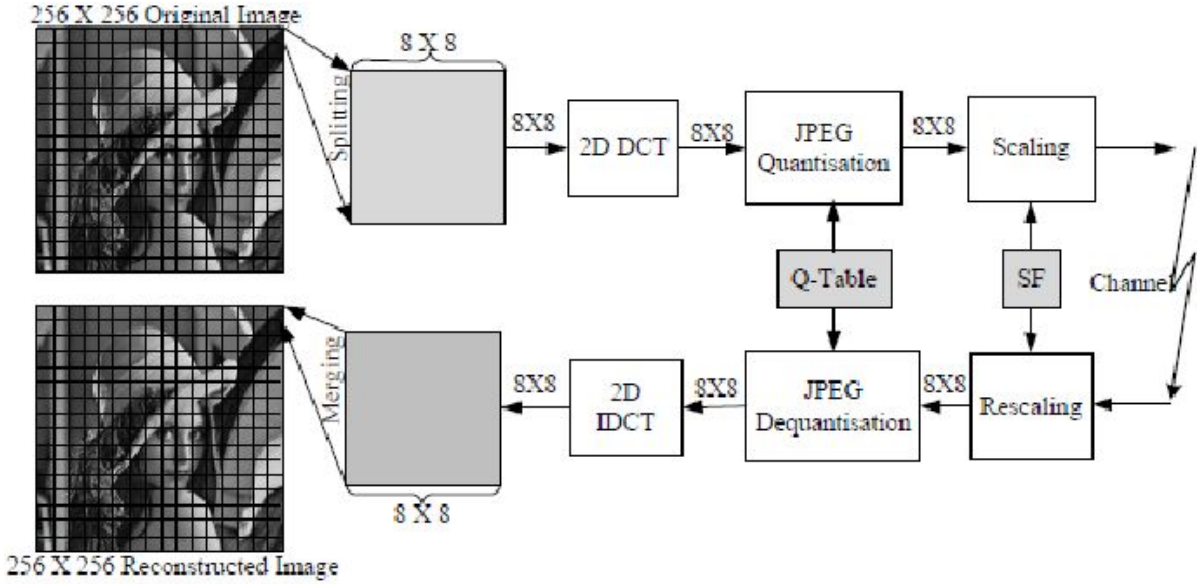
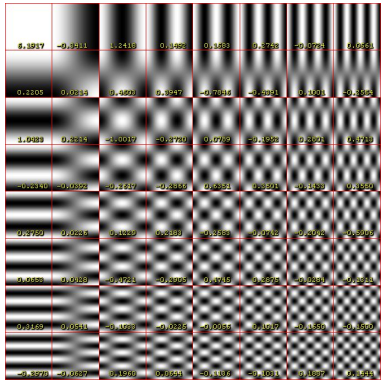


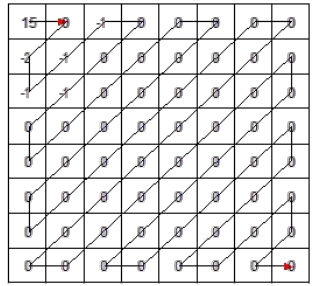
Figure 3: Block diagram of the JPEG based DCT

DCT basis

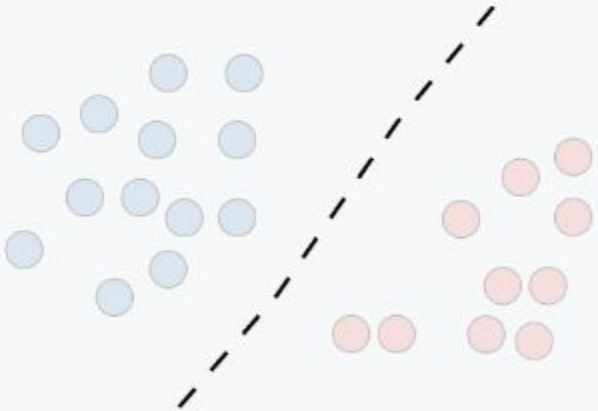
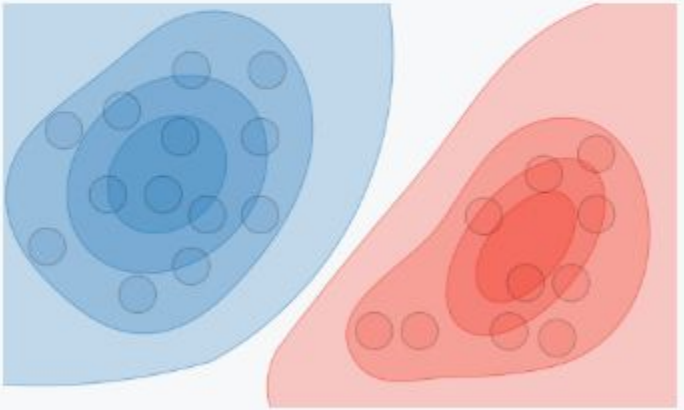


Huffman coding

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99



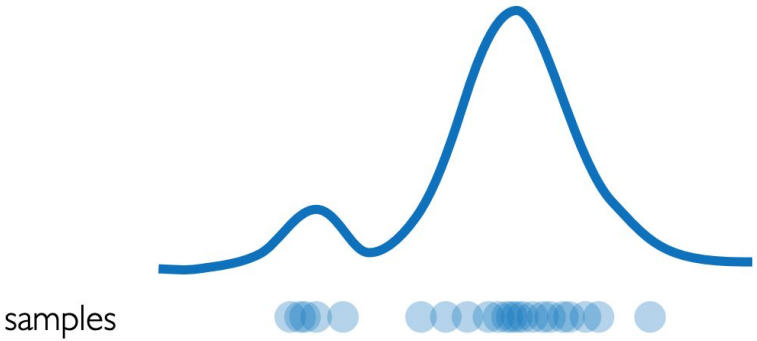
Generative vs. Discriminative Models

	Discriminative model	Generative model
Goal	Directly estimate $P(y\|x)$	Estimate $P(x\|y)$ to then deduce $P(y\|x)$
What's learned	Decision boundary	Probability distributions of the data
Illustration		
Examples	Regressions, SVMs	GDA, Naive Bayes

Generative Modeling

Goal: Take as input training samples from some distribution and learn a model that represents that distribution

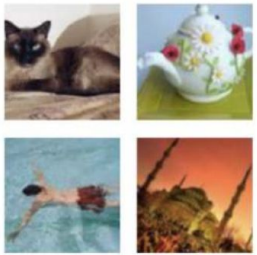
Density Estimation



Sample Generation

Input samples

Training data ~ $P_{data}(x)$



Generated samples

Generated ~ $P_{model}(x)$

Variational Autoencoders

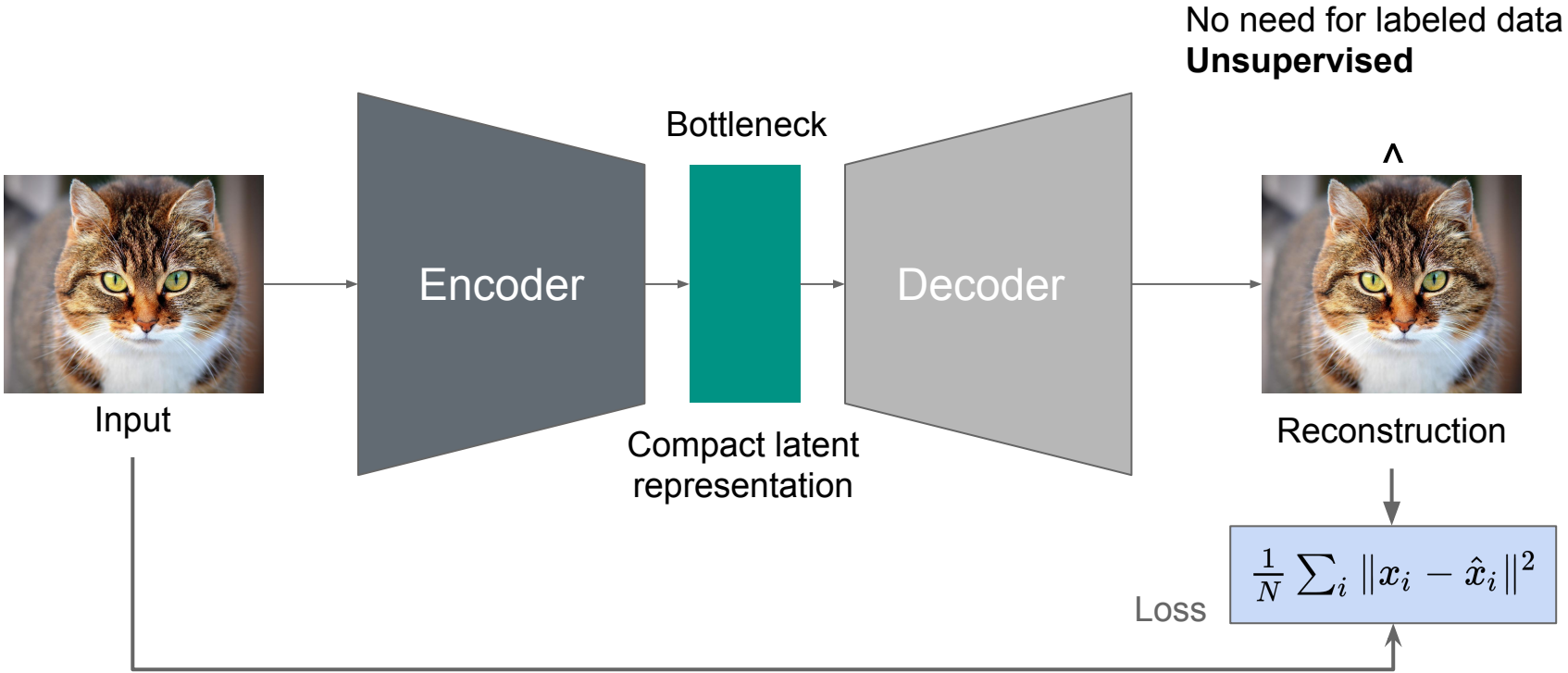
Autoencoders

Variational Autoencoders

~~Reparameterization Trick (Stochastic Backprop)~~

~~Loss and the ELBO~~

Autoencoders



Autoencoders

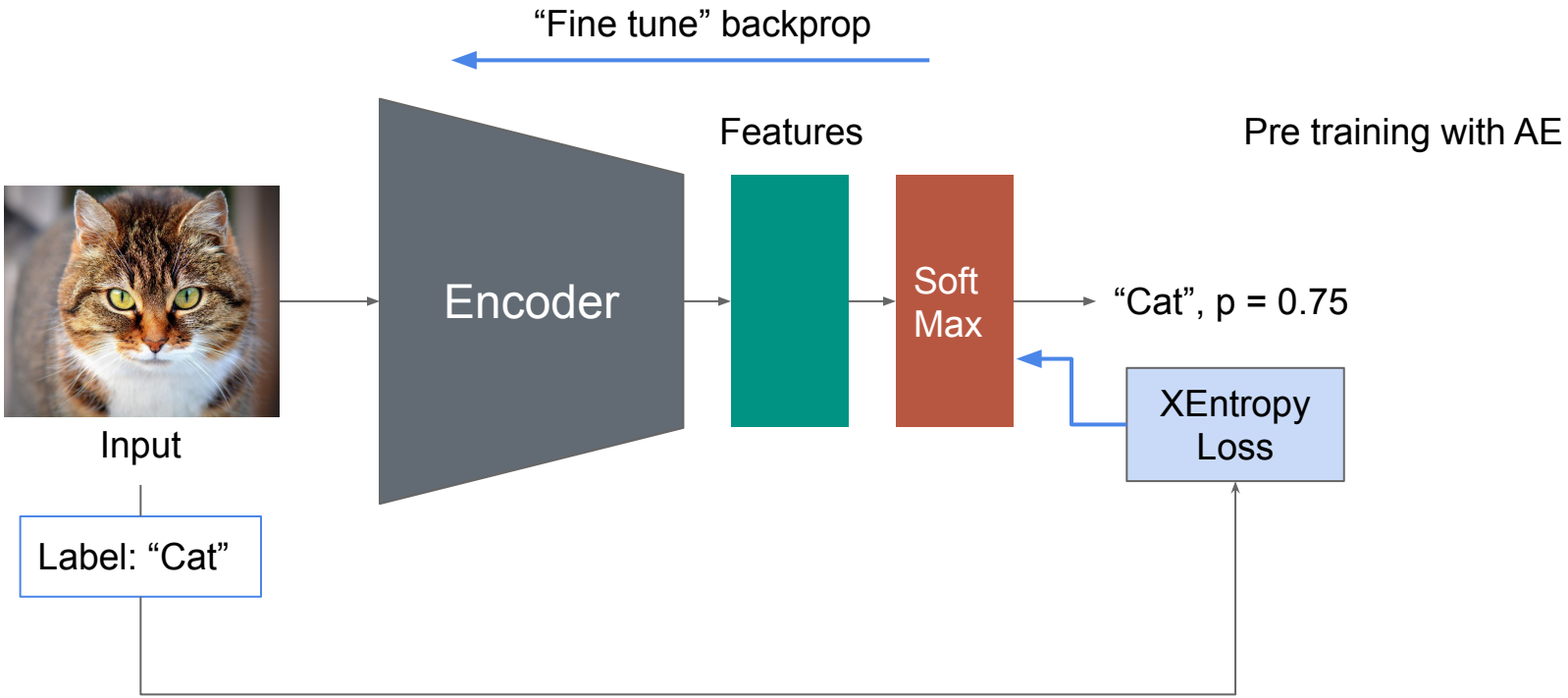
Autoencoding is a form of compression!
Smaller latent space will force a larger training bottleneck

2D latent space

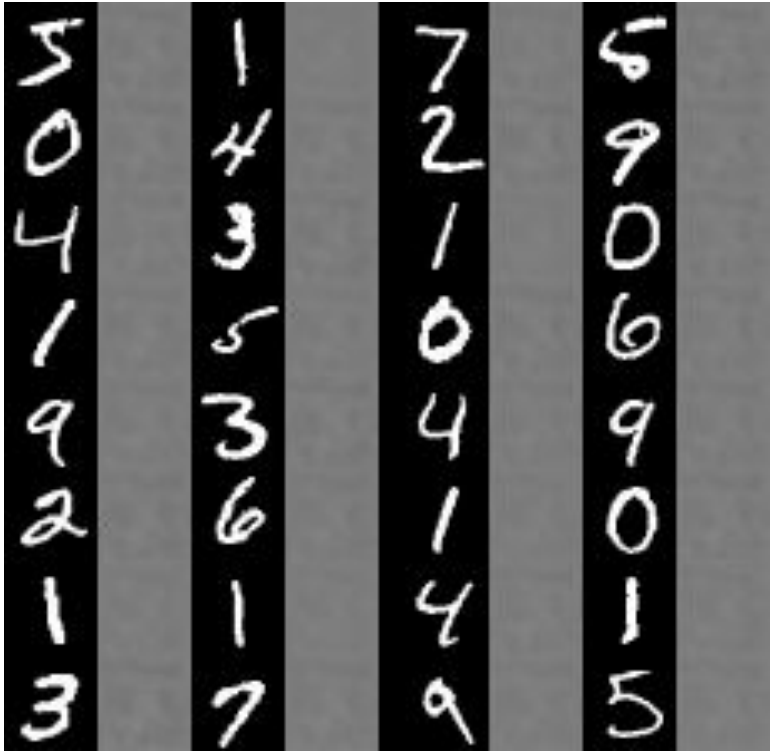
5D latent space

Ground Truth

Autoencoders | Warm-up for Discriminators



Autoencoders



Autoencoders Application: Denoising

AEs are widely used for denoising.

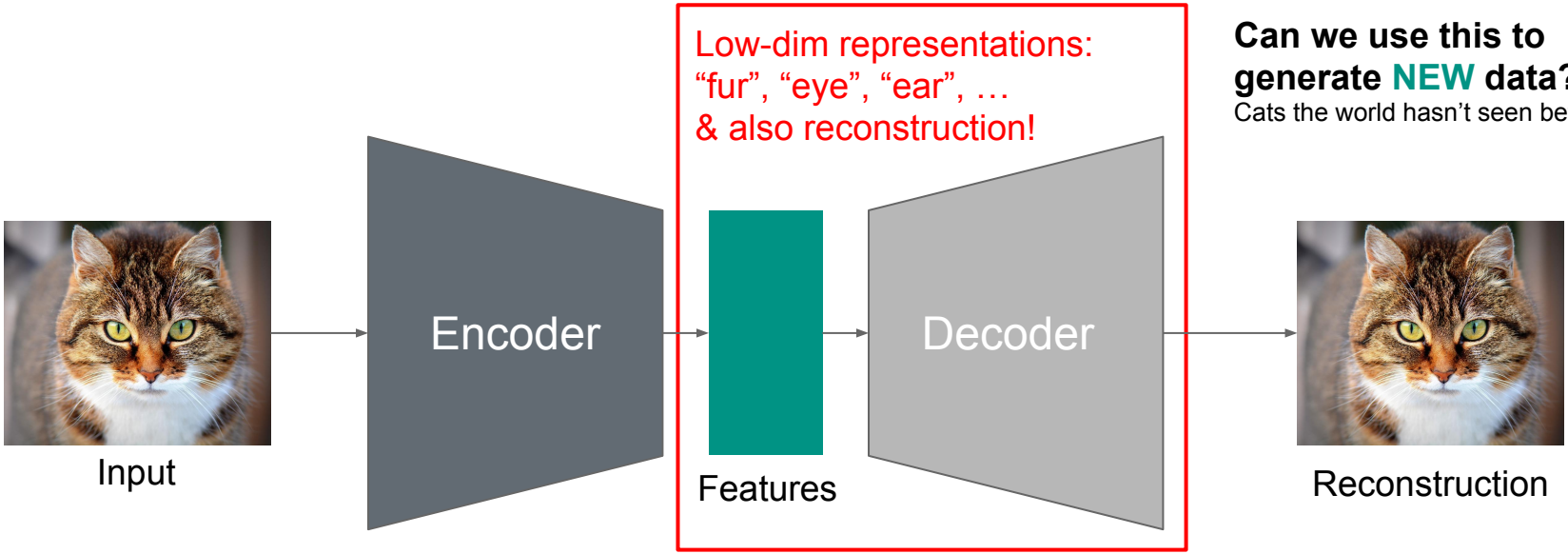
Since they are able to capture the essence of the shape.

AEs can be used on any type of data...

[Chollet]

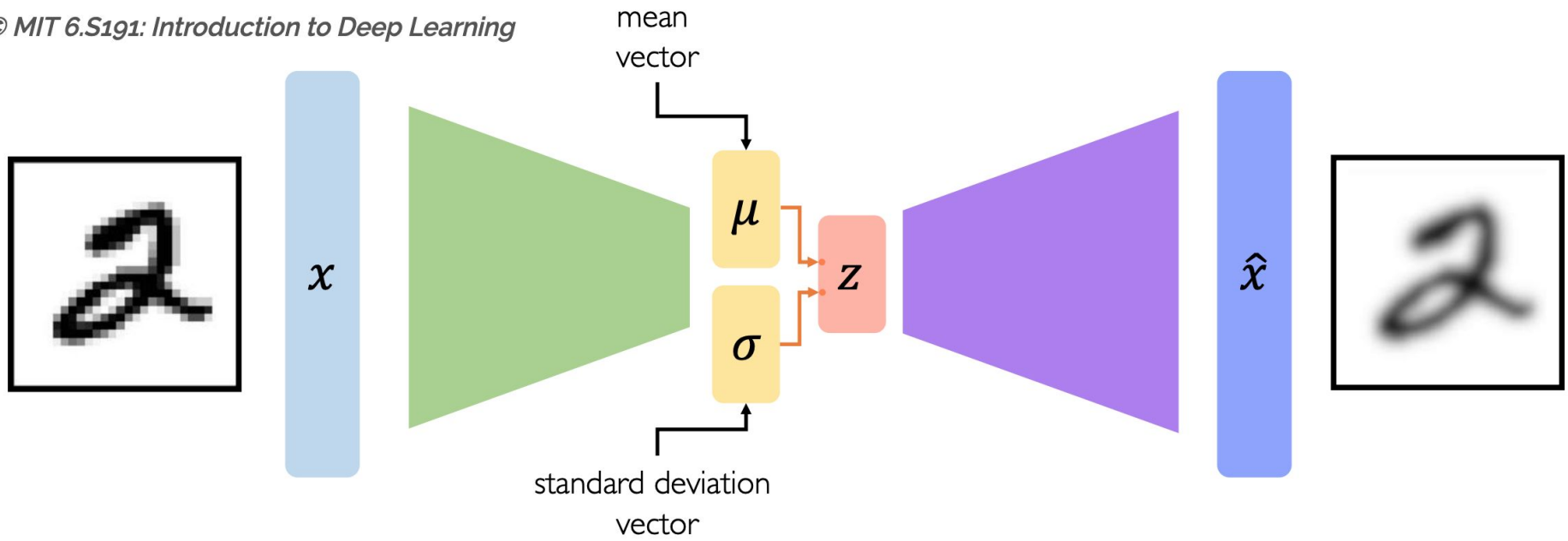
[Link]

Autoencoders



Major problem: Latent space is unbounded and doesn’t have any semantic value.

Variational Autoencoder



Variational autoencoders are a probabilistic twist on autoencoders!

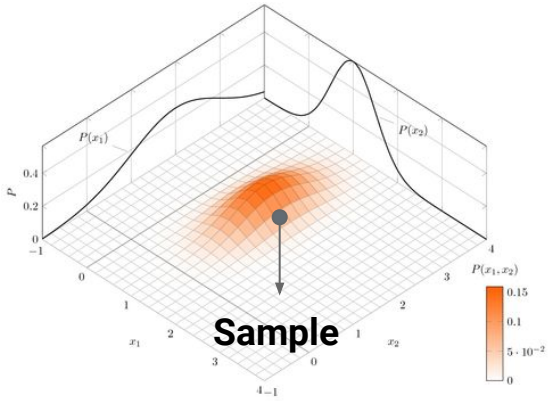
Sample from the mean and standard dev. to compute latent sample

Variational Autoencoder

x

Encoder

μ, σ

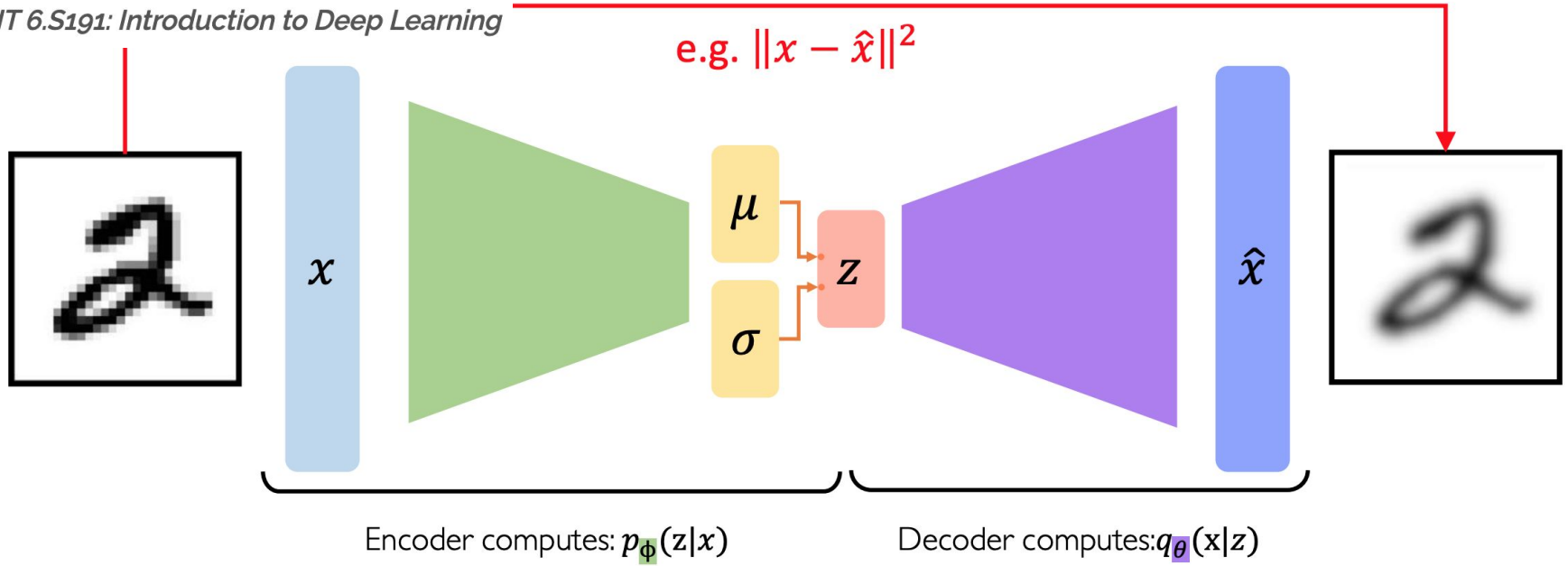


Decoder

$\hat{x}$

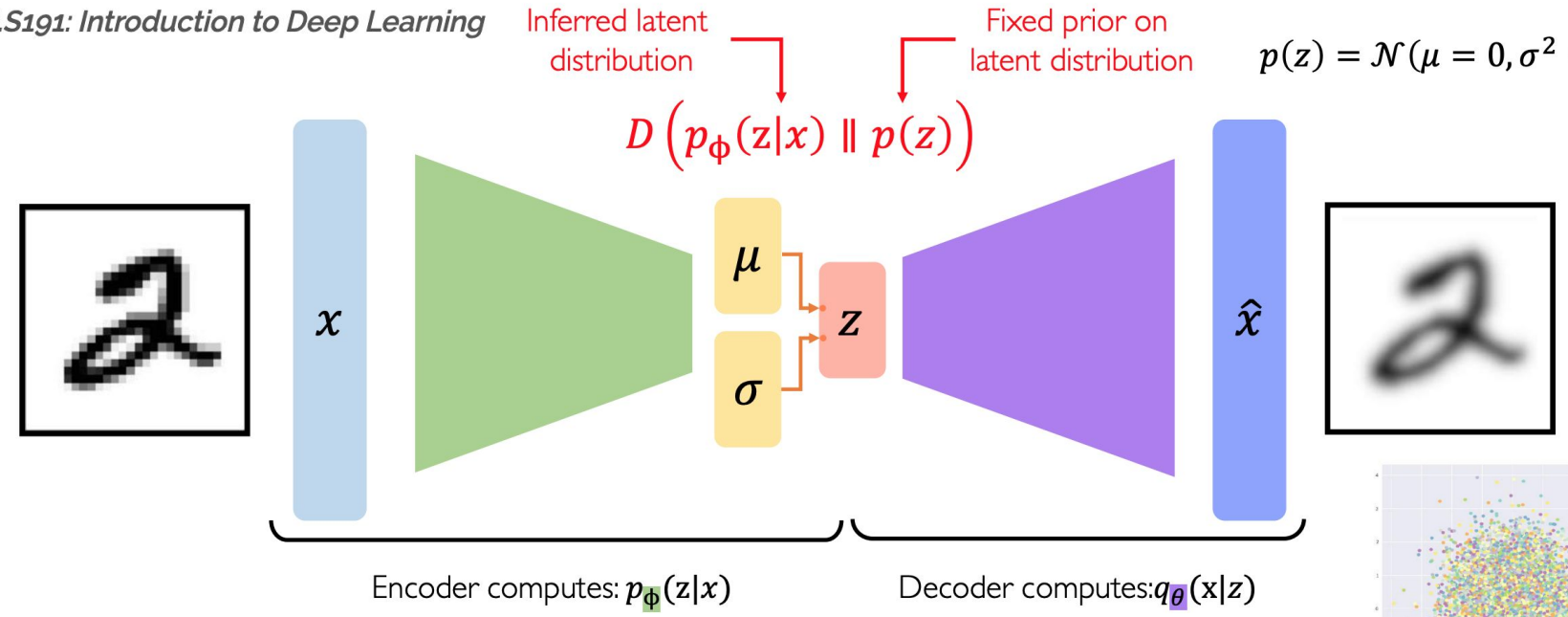
Sampling is non-deterministic
Involves randomness

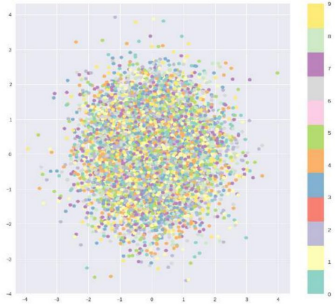
Training VAEs



$$\mathcal{L}(\phi, \theta, x) = \boxed{\text{(reconstruction loss)}} + \text{(regularization term)}$$

Training VAEs





* Reparameterization Trick

$$\mathcal{L}(\phi, \theta, x) = \text{(reconstruction loss)} + \boxed{\text{(regularization term)}}$$

VAEs | Generate Data

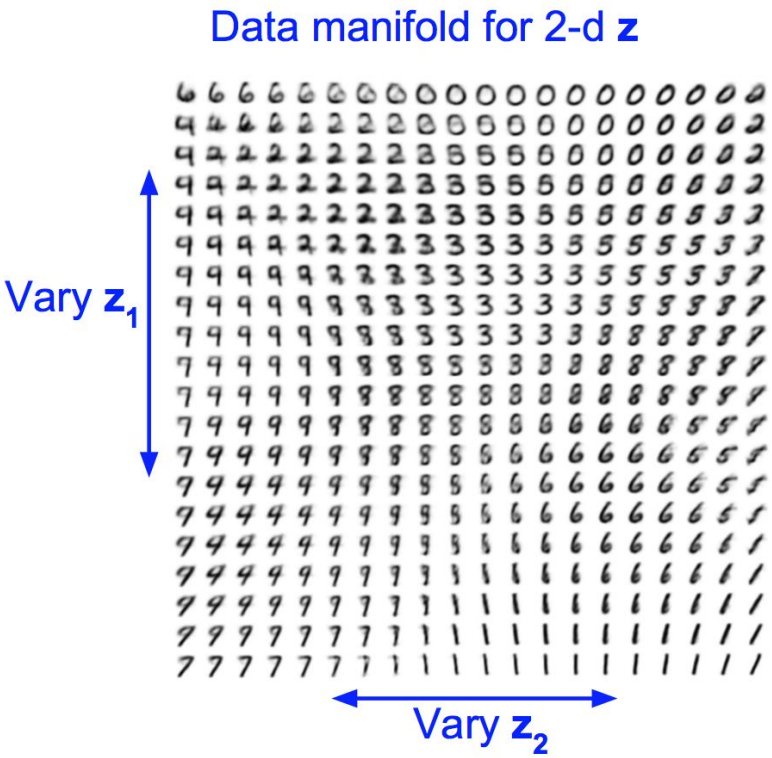
Simply sample z from $\sim \mathcal{N}(0, I)$ and feedforward:

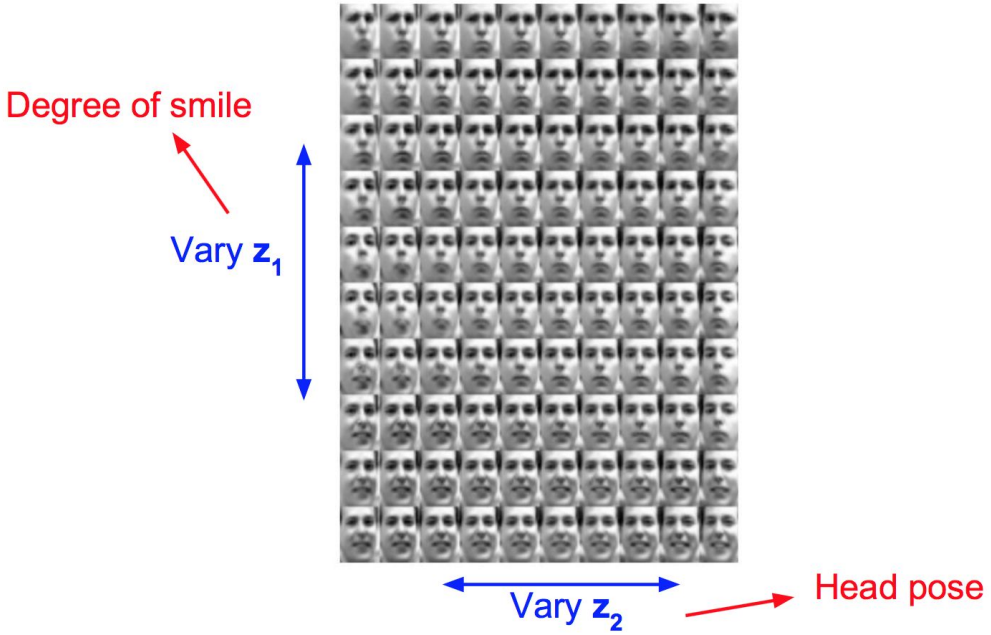
```
codings_rnd = np.random.normal(size=[n_digits, n_hidden3])
outputs_val = outputs.eval(feed_dict={hidden3: codings_rnd})
```

[Géron]

VAEs | Latent Vector Manipulation

[Fei Fei Li]





VAEs | Latent Vector Manipulation

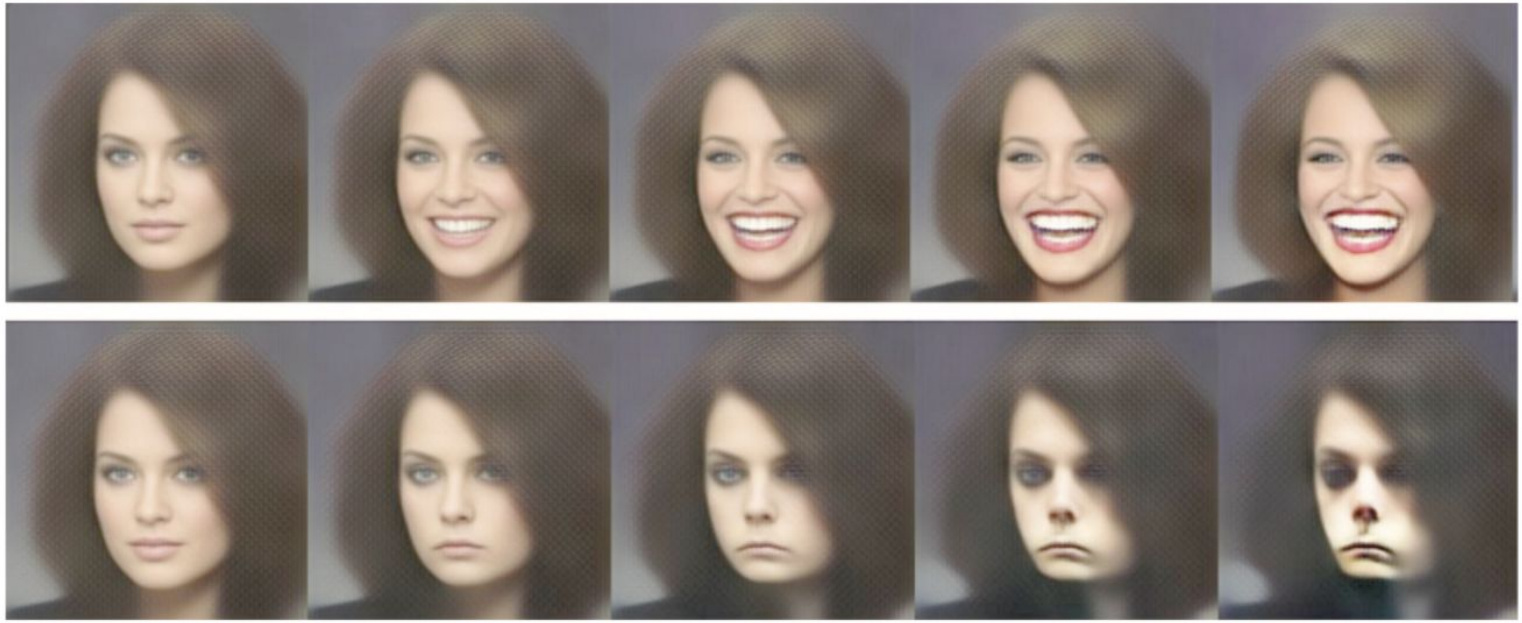


Figure 8.11 The smile vector

VAEs | Interpolation

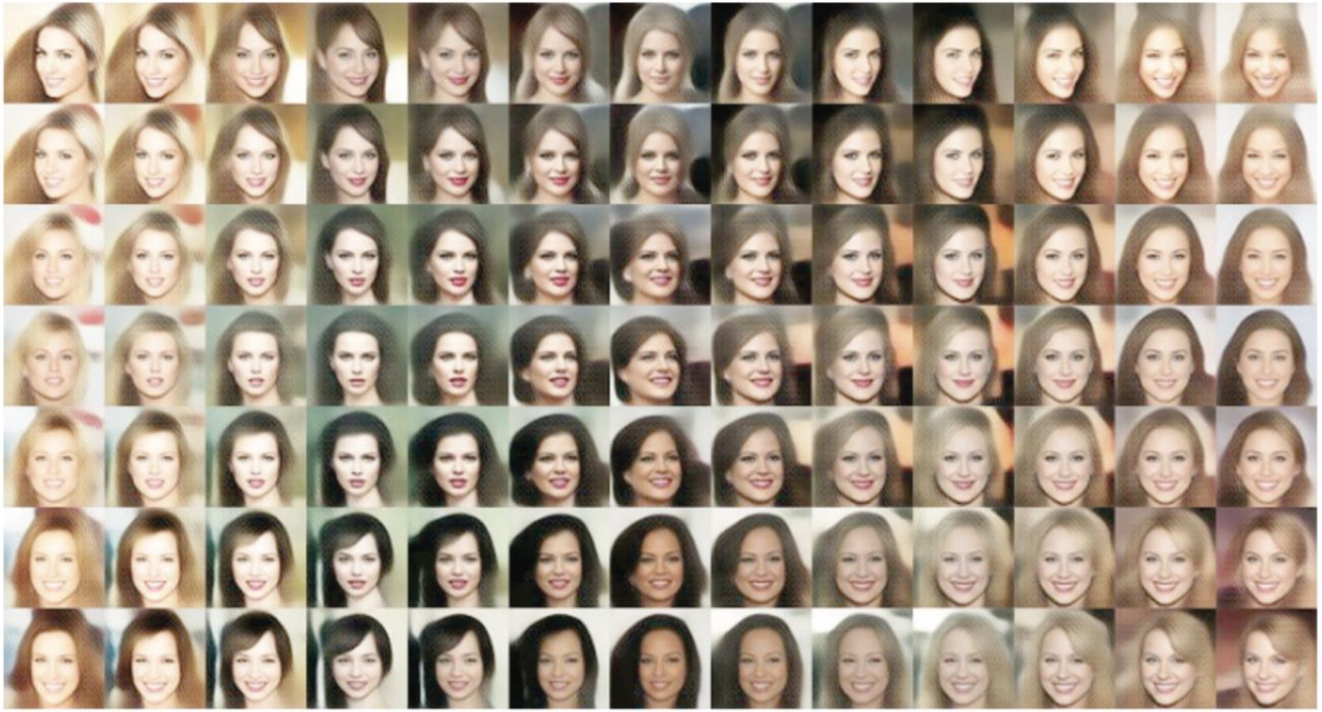


Figure 8.10 A continuous space of faces generated by Tom White using VAEs

[Chollet]

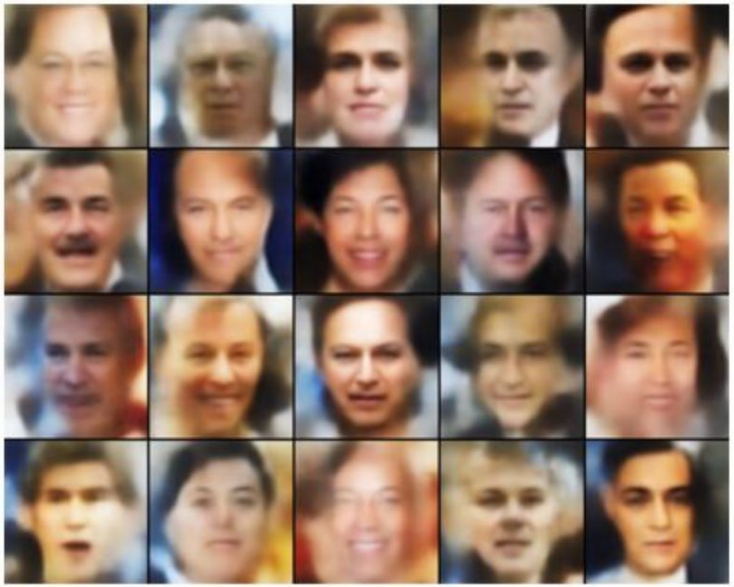
VAEs | Interpolation

https://experiments.withgoogle.com/ai/beat-blender/view/

VAEs | Generate Data

[Fei Fei Li]

32x32 CIFAR-10



Labeled Faces in the Wild

Generative Adversarial Neworks (GANs)

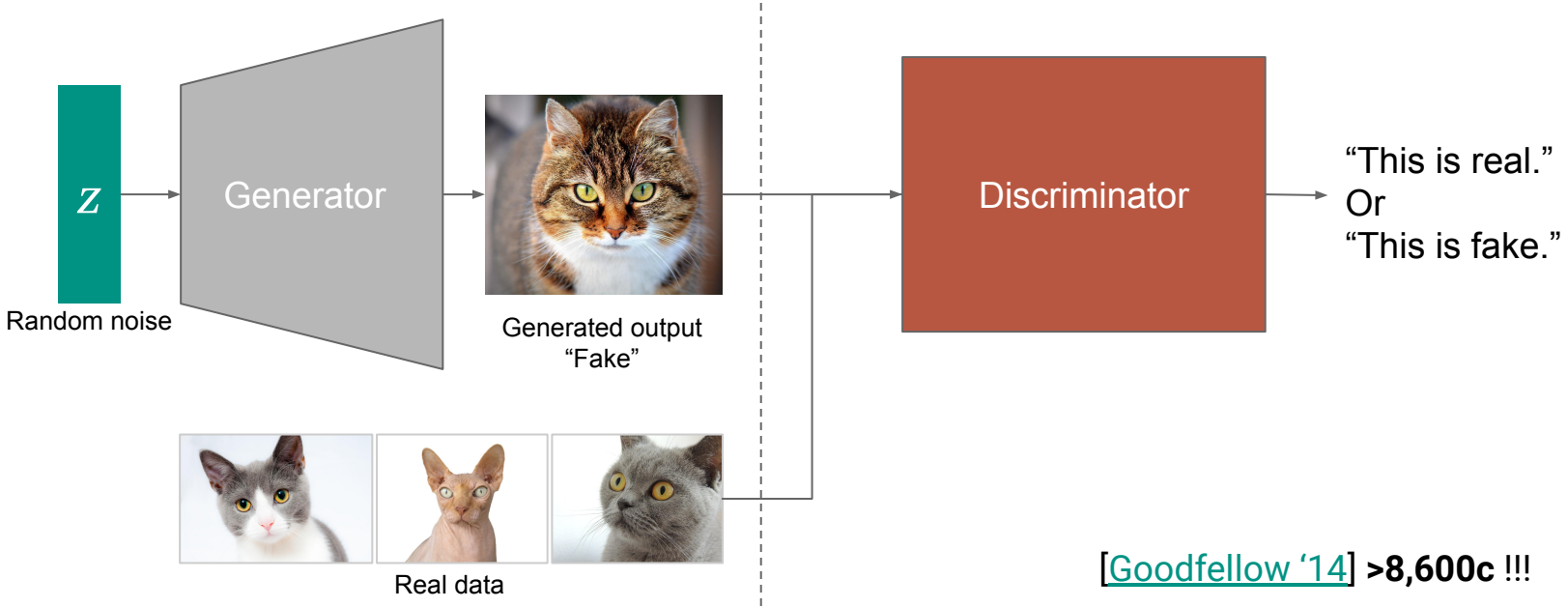
GAN Intro

GAN Supervision (CGAN, InfoGAN)

Generative Adversarial Networks

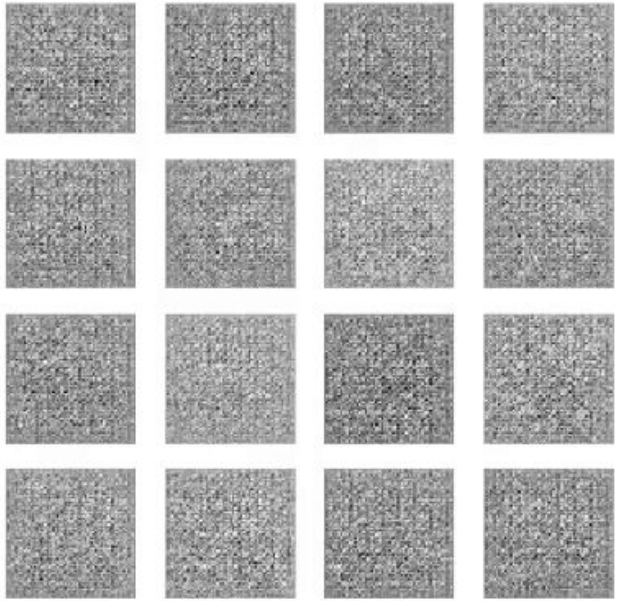
Problem: Distribution of “Cat Images” very complex.
We can’t sample from it directly.

Idea: Sample random noise, learn how to turn noise into image (a sample from distribution).



[Goodfellow ‘14] **>8,600c** !!!

GANs Training



GANs Training

Step 0: **Generator** Discriminator

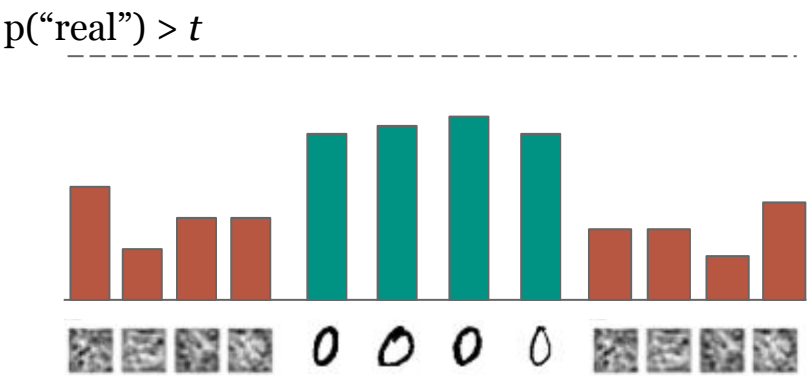
p(“real”) > t

Samples generated Samples seen

GANs Training

Step 1: Generator **Discriminator**

Samples generated

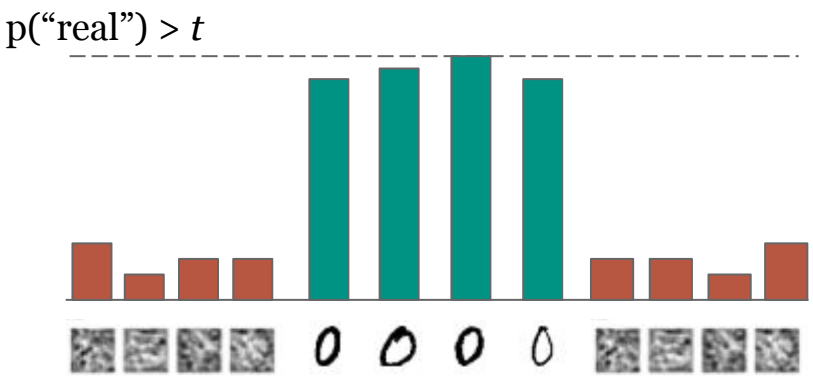


Samples seen

GANs Training

Step 2: Generator **Discriminator**

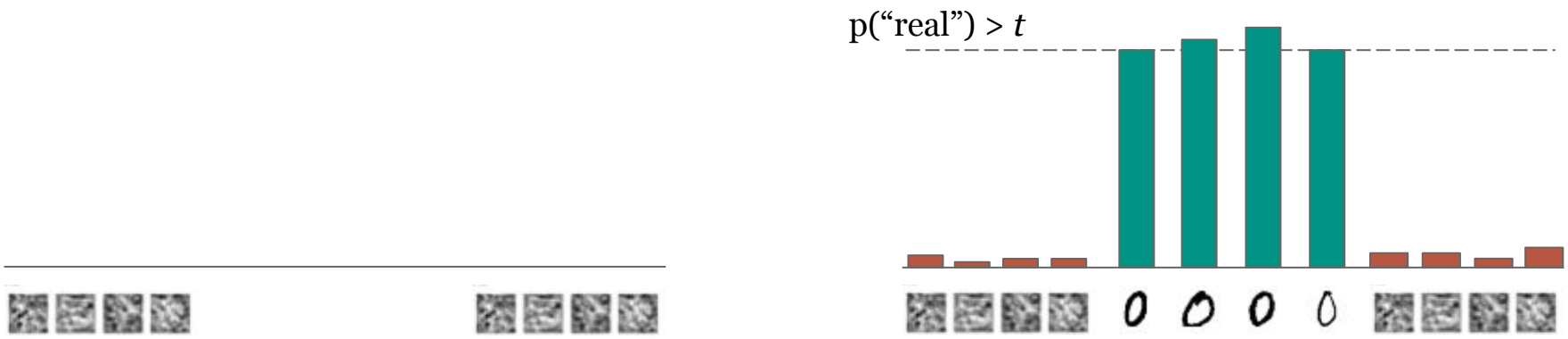
Samples generated



Samples seen

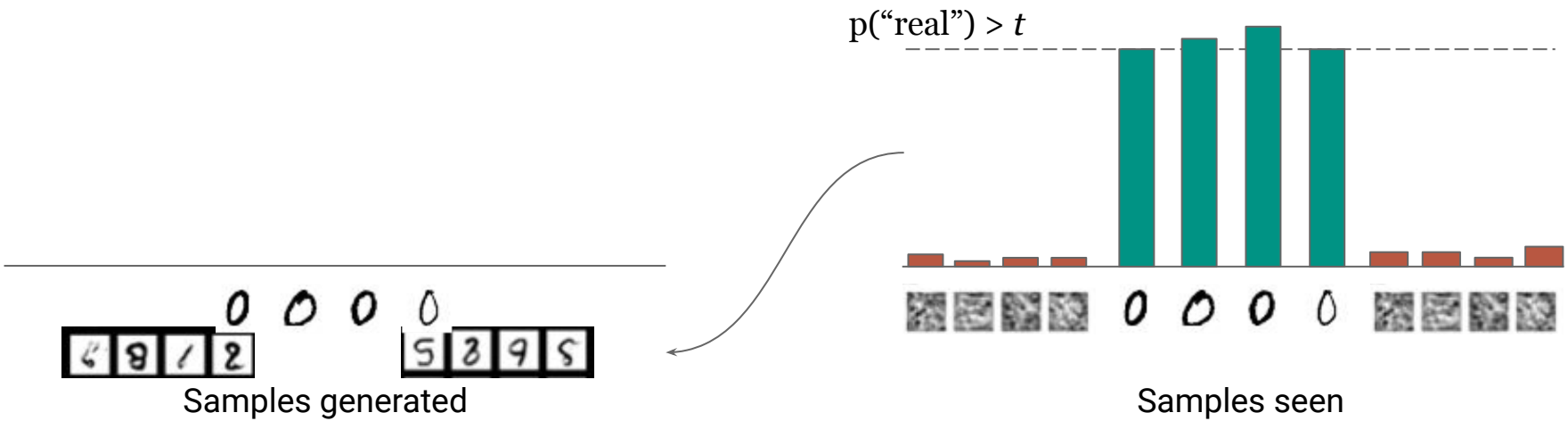
GANs Training

Step 3: Generator **Discriminator**



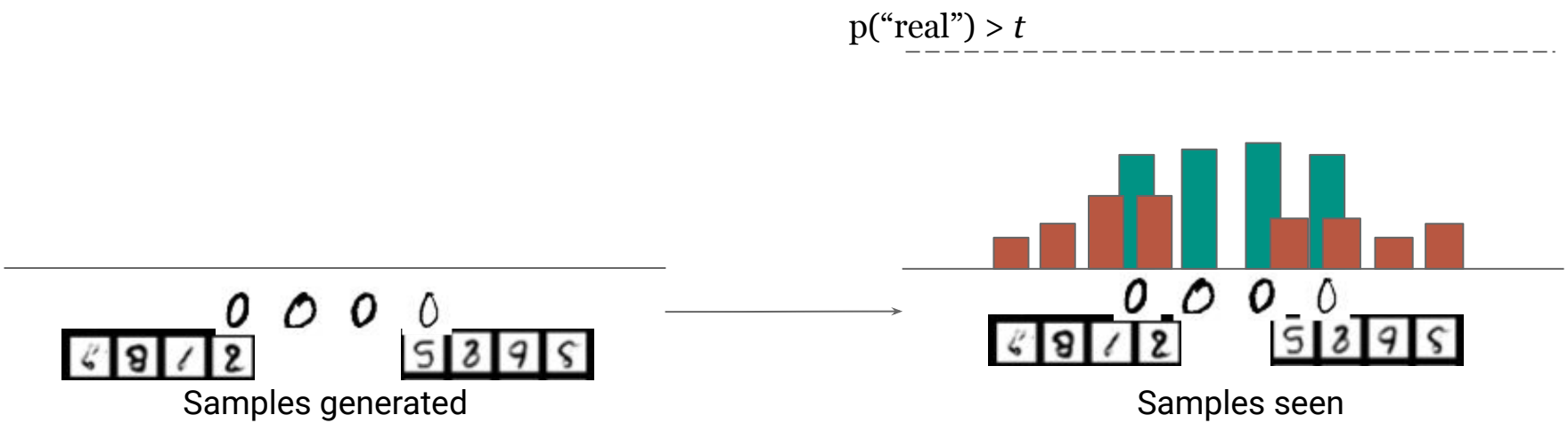
GANs Training

Step 4: **Generator** Discriminator



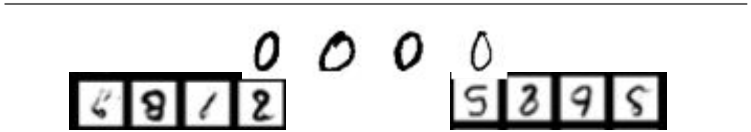
GANs Training

Step 5: Generator **Discriminator**

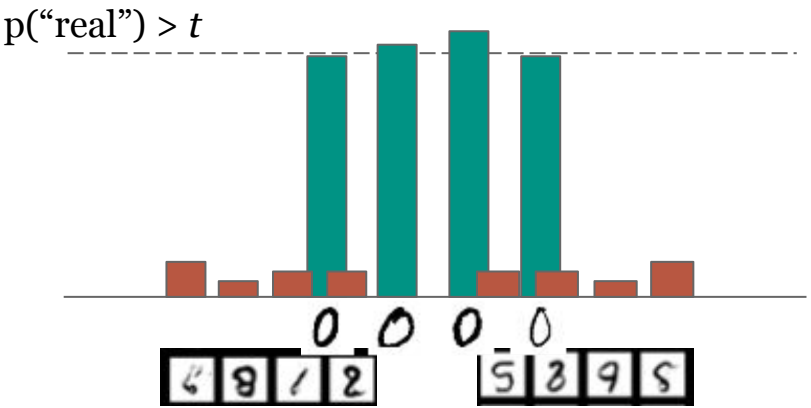


GANs Training

Step 5: Generator **Discriminator**



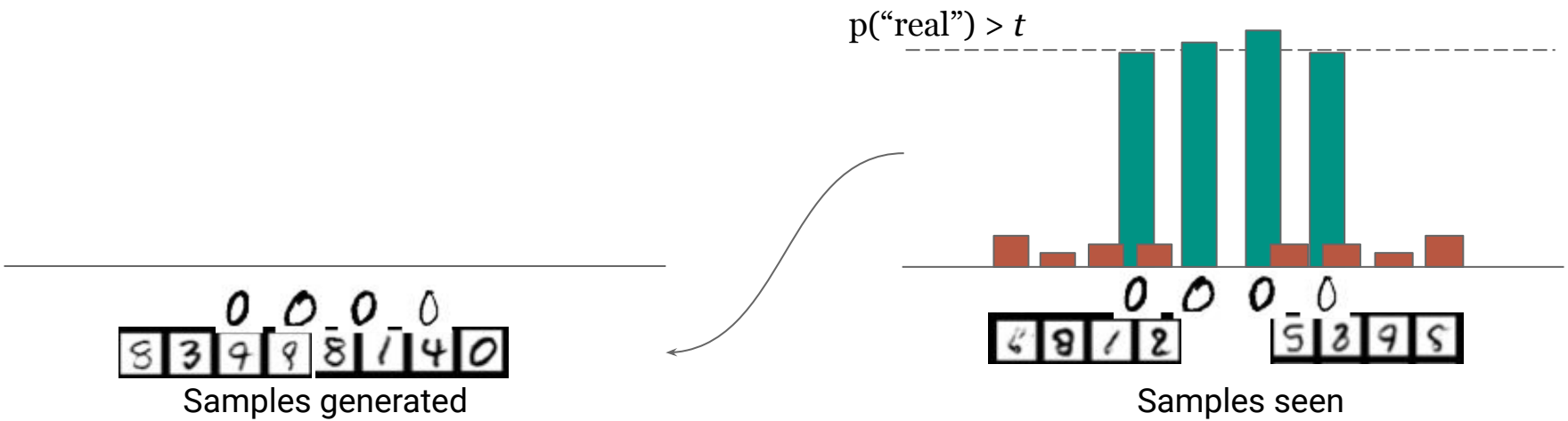
Samples generated



Samples seen

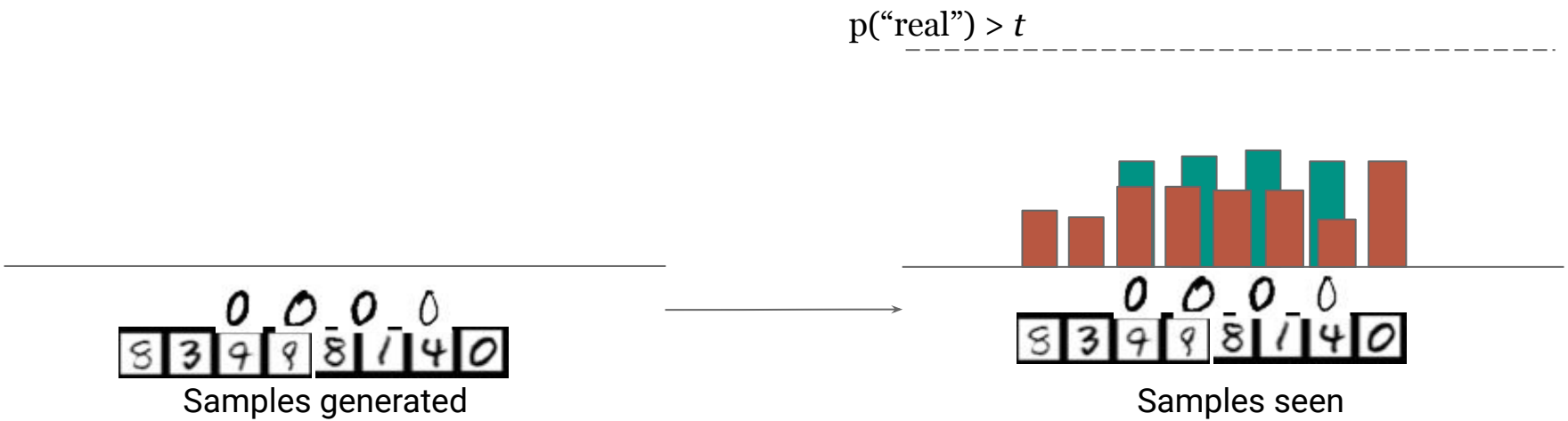
GANs Training

Step 6: **Generator** Discriminator



GANs Training

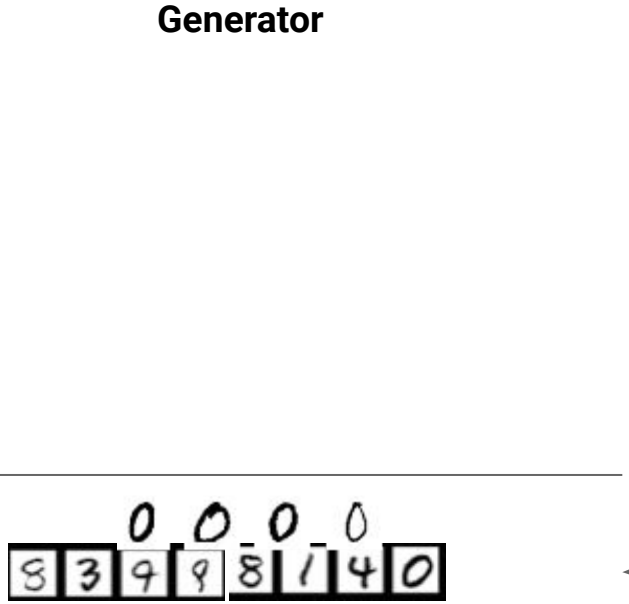
Step 7: Generator **Discriminator**



GANs Training

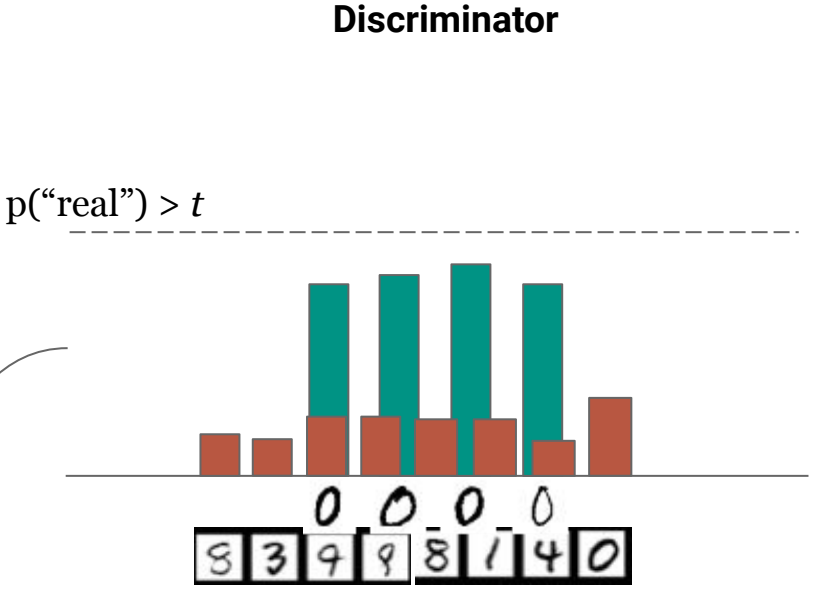
Step 8:

Generator



Samples generated

Discriminator



Samples seen

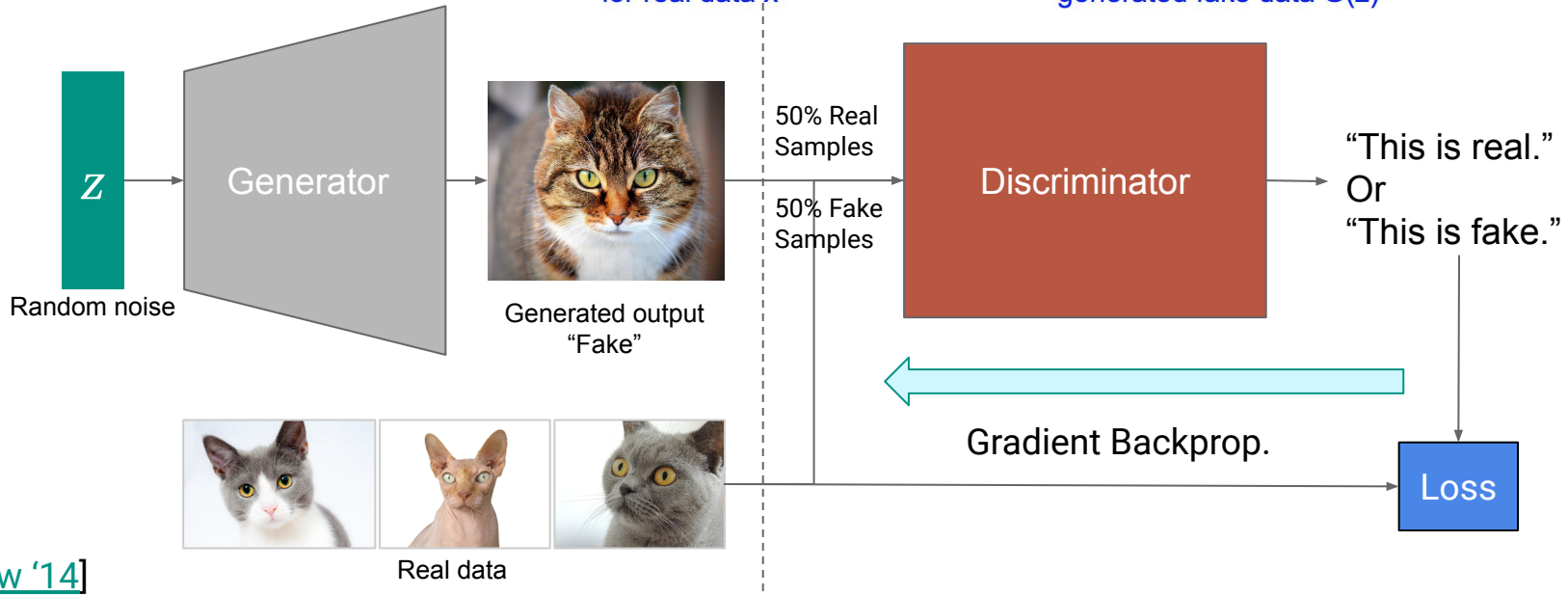
GANs Training

[Fei Fei Li]

Minimax objective: $$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Discriminator output for real data x

Discriminator output for generated fake data G(z)



[Goodfellow '14]

GANs | Results

Deep Convolutional GANs (DCGANs)

[Radford '15] **>3,400c !!**

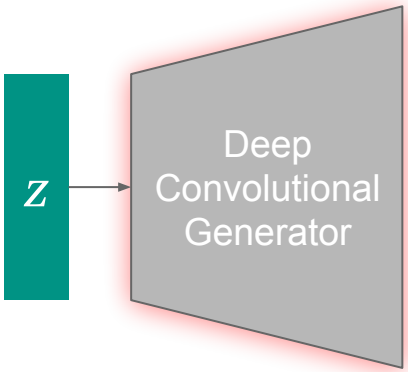
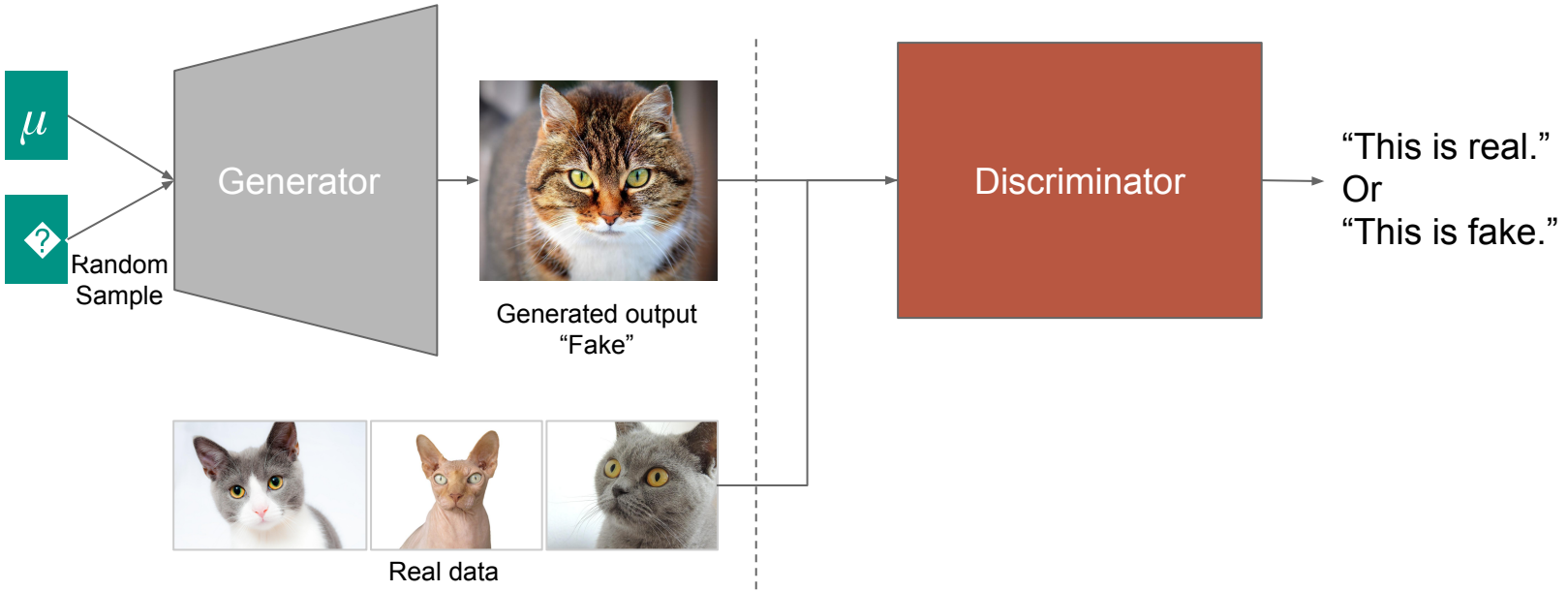


Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

VAE-GAN

Simple: Take PDF modeling idea from VAE and use for GAN G side.

GAN is a good generator. VAE is a “parameterizer”. [Larsen 2015] >500c



VAE–GAN

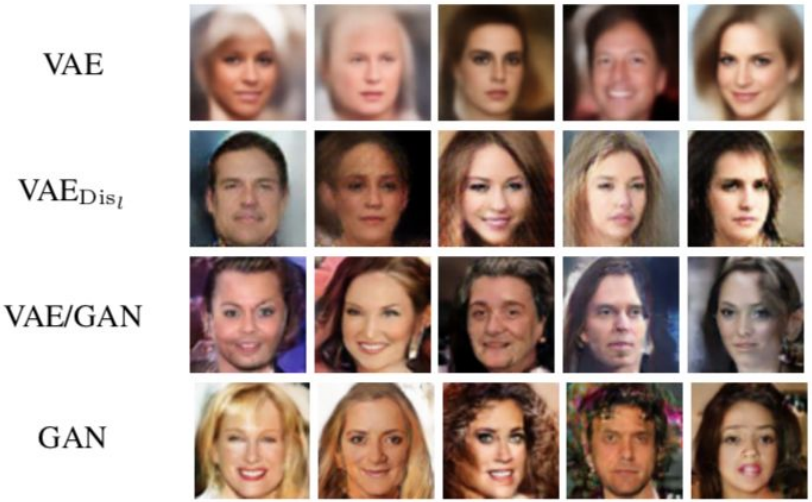


Figure 3. Samples from different generative models.

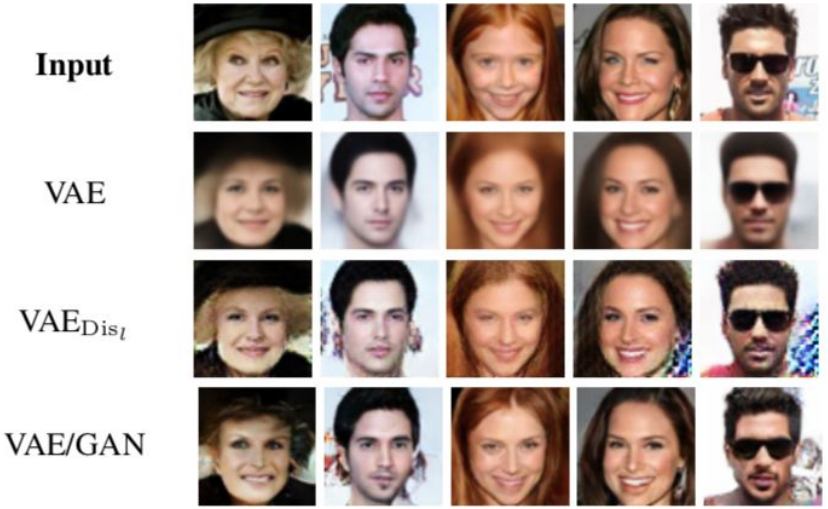


Figure 4. Reconstructions from different autoencoders.

[Larsen 2015]

VAE-GAN

[Larsen 2015]

Figure 5. Using the VAE/GAN model to reconstruct dataset samples with visual attribute vectors added to their latent representations.

Questions?

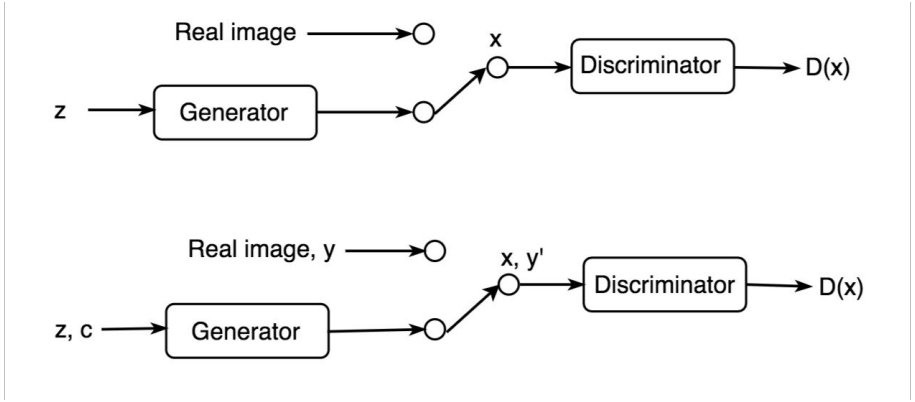
Conditional GAN

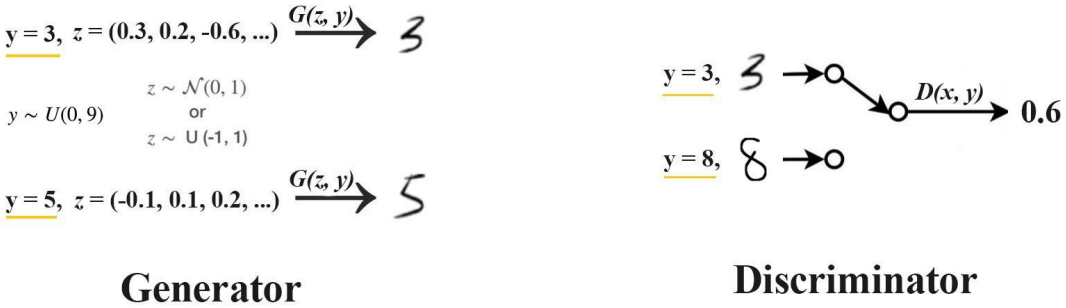
Goal: Better control of generation.

Idea: Add alongside the image a labeling, classification.

[Isola ‘16] >2,100c (!!)

Motivation: Our (human) visual system is biased by “concepts” / “labels”.





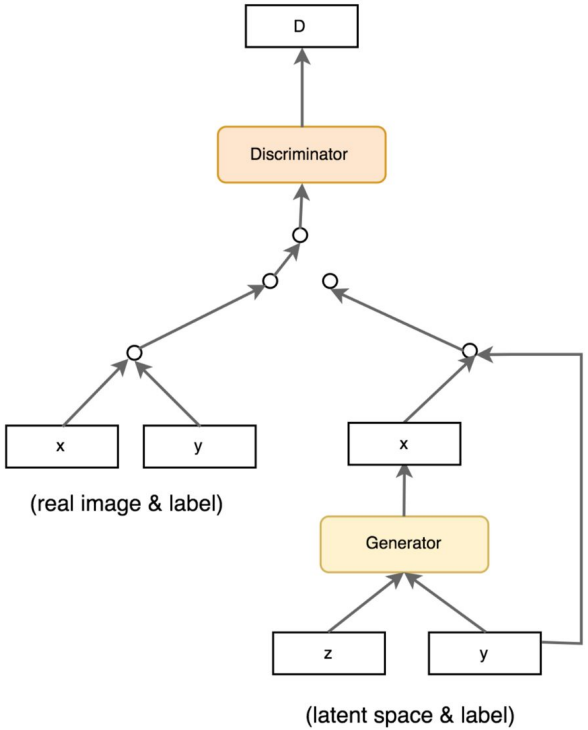
[Link]

Conditional GAN

Training CGAN is mostly similar to regular GAN.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

Novelty: Discriminator gets image & label.

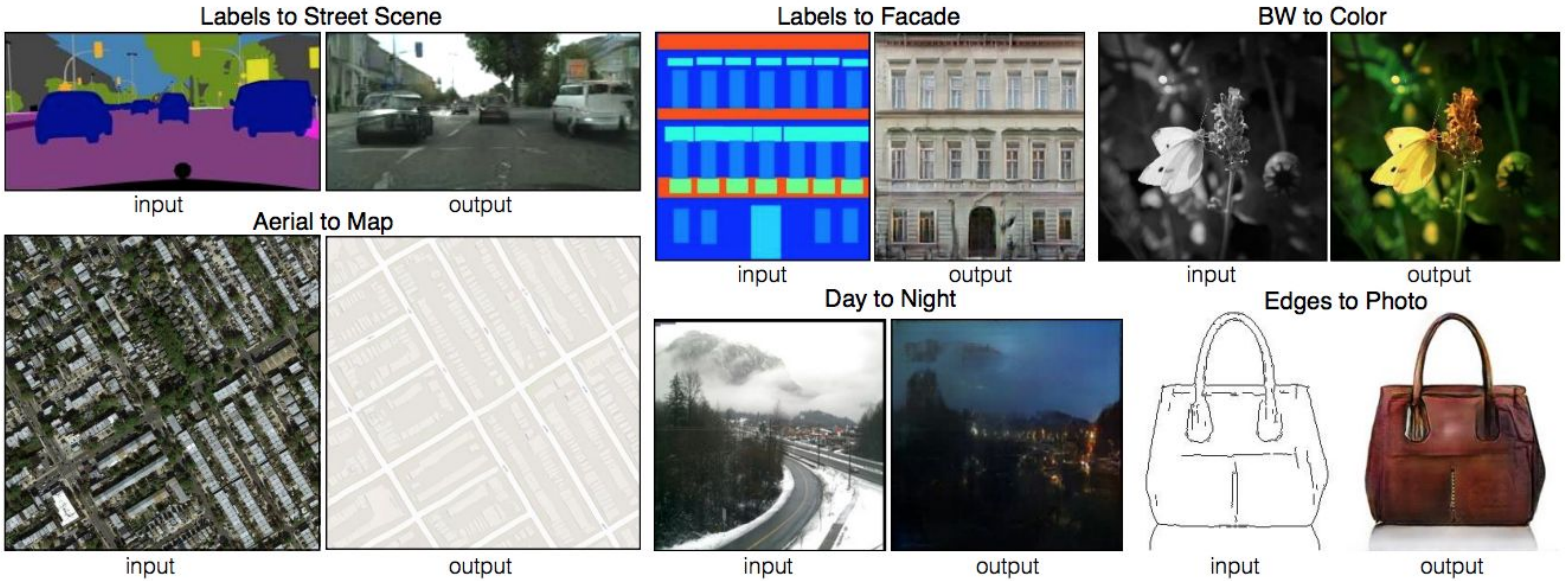


[Link]

CGANs | Results

Image-to-Image

[Isola]



CGANs | Results

Image-to-Image

[Isola]

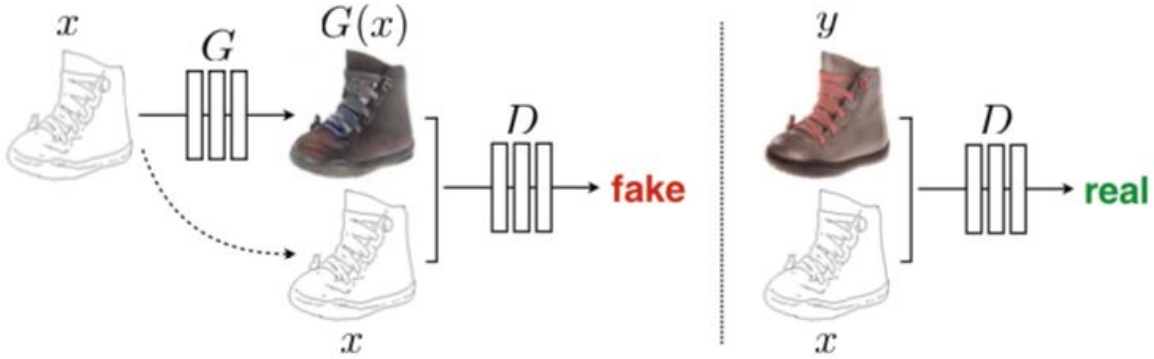


Figure 2: Training a conditional GAN to map edges→photo. The discriminator, D, learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator, G, learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

CGANs | Results

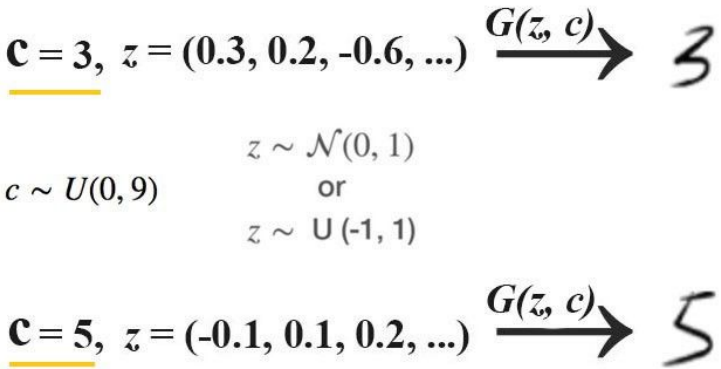
Image-to-Image

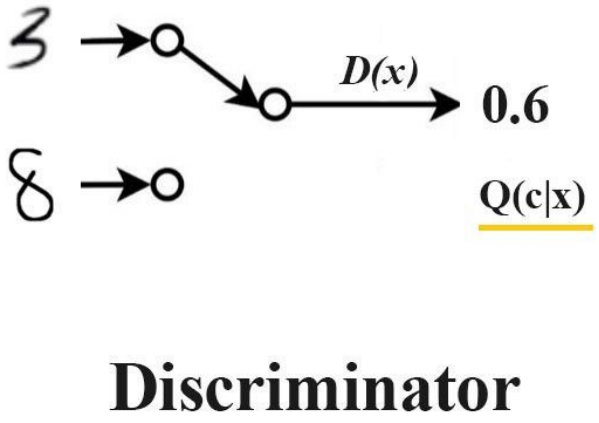
[Isola]

InfoGAN

What if we don't have specific labeling in the data?

The GAN can learn them for you!

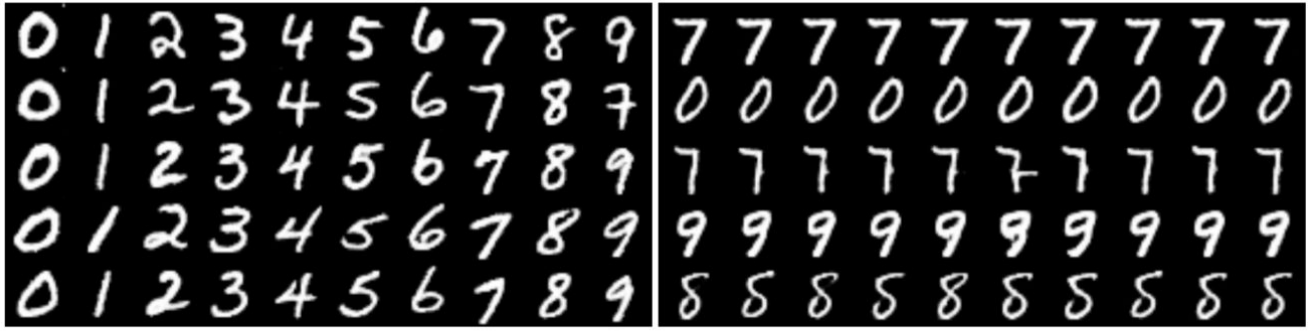




[Link]

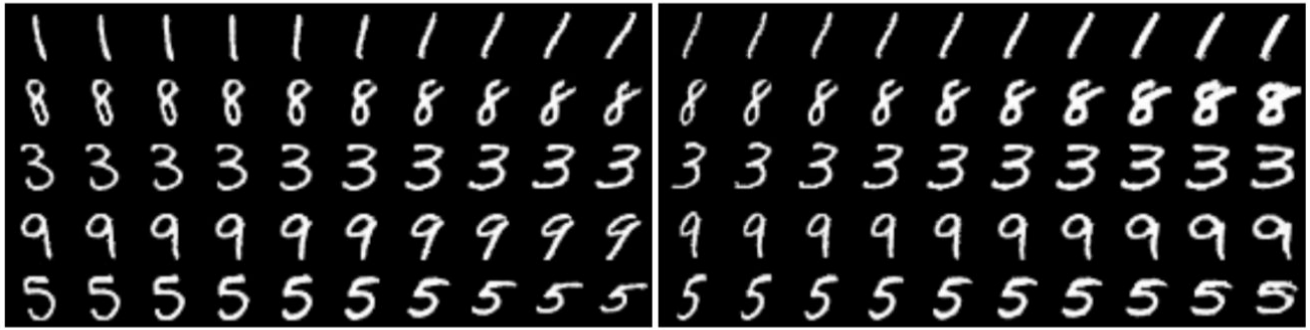
InfoGAN | Results

[Chen ‘16] >900c



(a) Varying c_1 on InfoGAN (Digit type)

(b) Varying c_1 on regular GAN (No clear meaning)



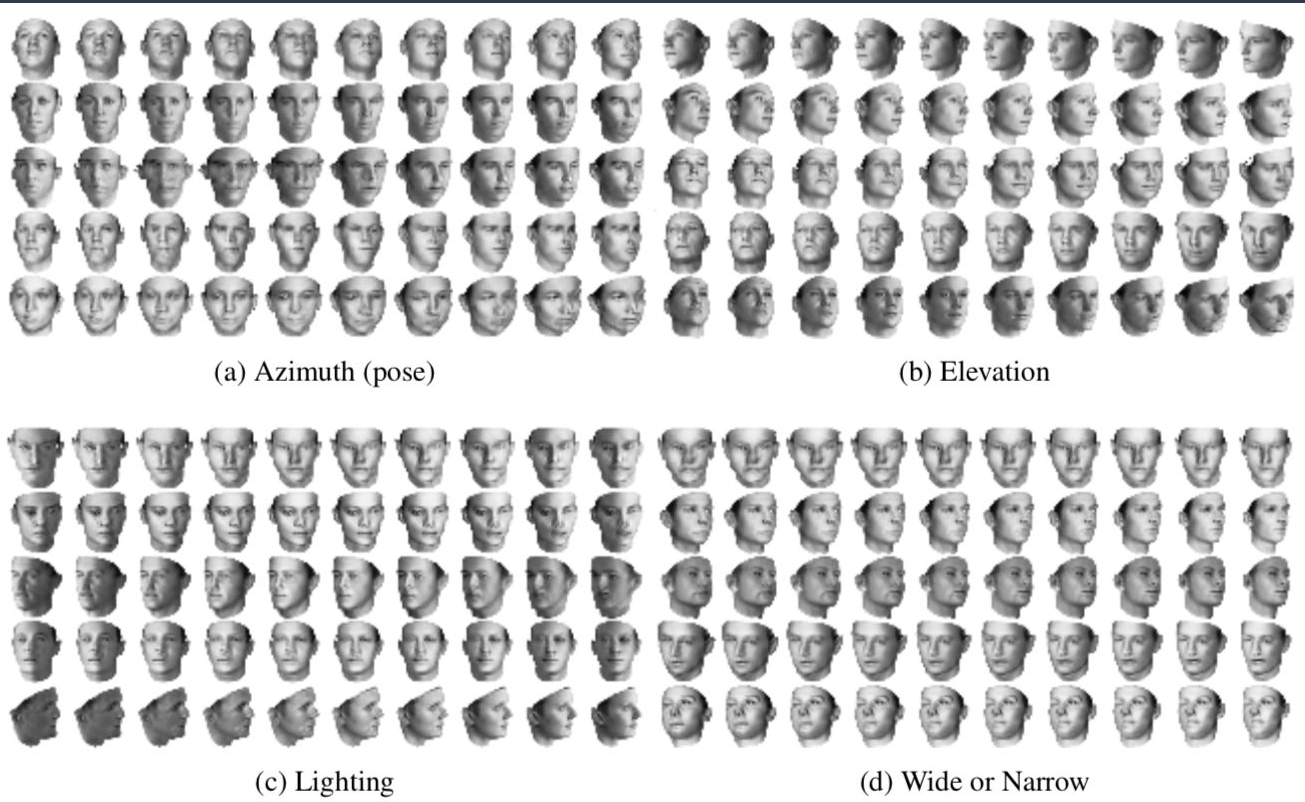
(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation)

(d) Varying c_3 from -2 to 2 on InfoGAN (Width)

[Link]

InfoGAN | Results

[Chen ‘16] >900c



[Link]

GANs | Results

Applications

Here comes the fun part!

StarGAN

CycleGAN

pix2pix(HD)

ProGAN

BigGAN

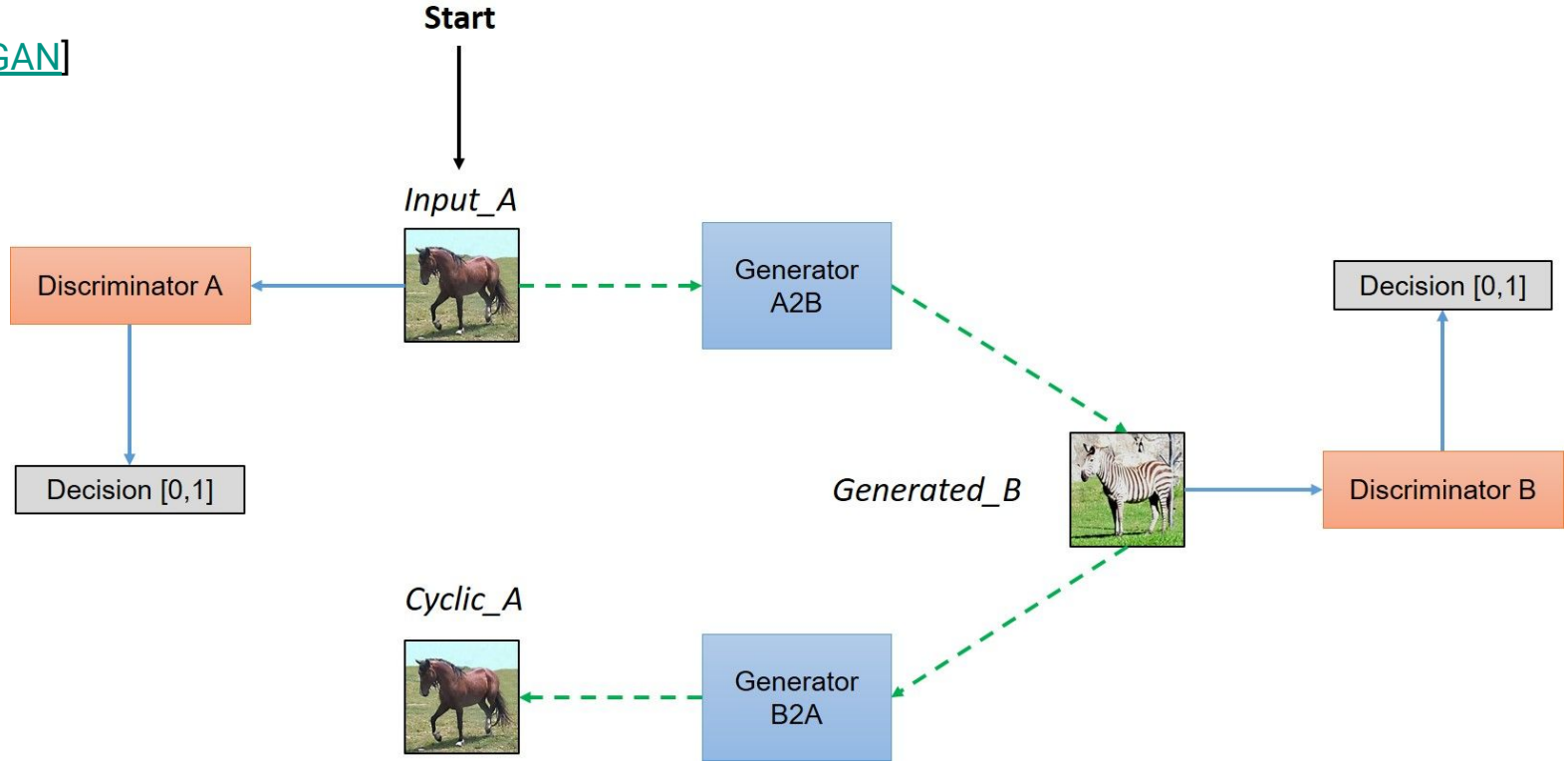
StyleGAN

UNIT / MUNIT / FUNIT / DRIT

StarGAN

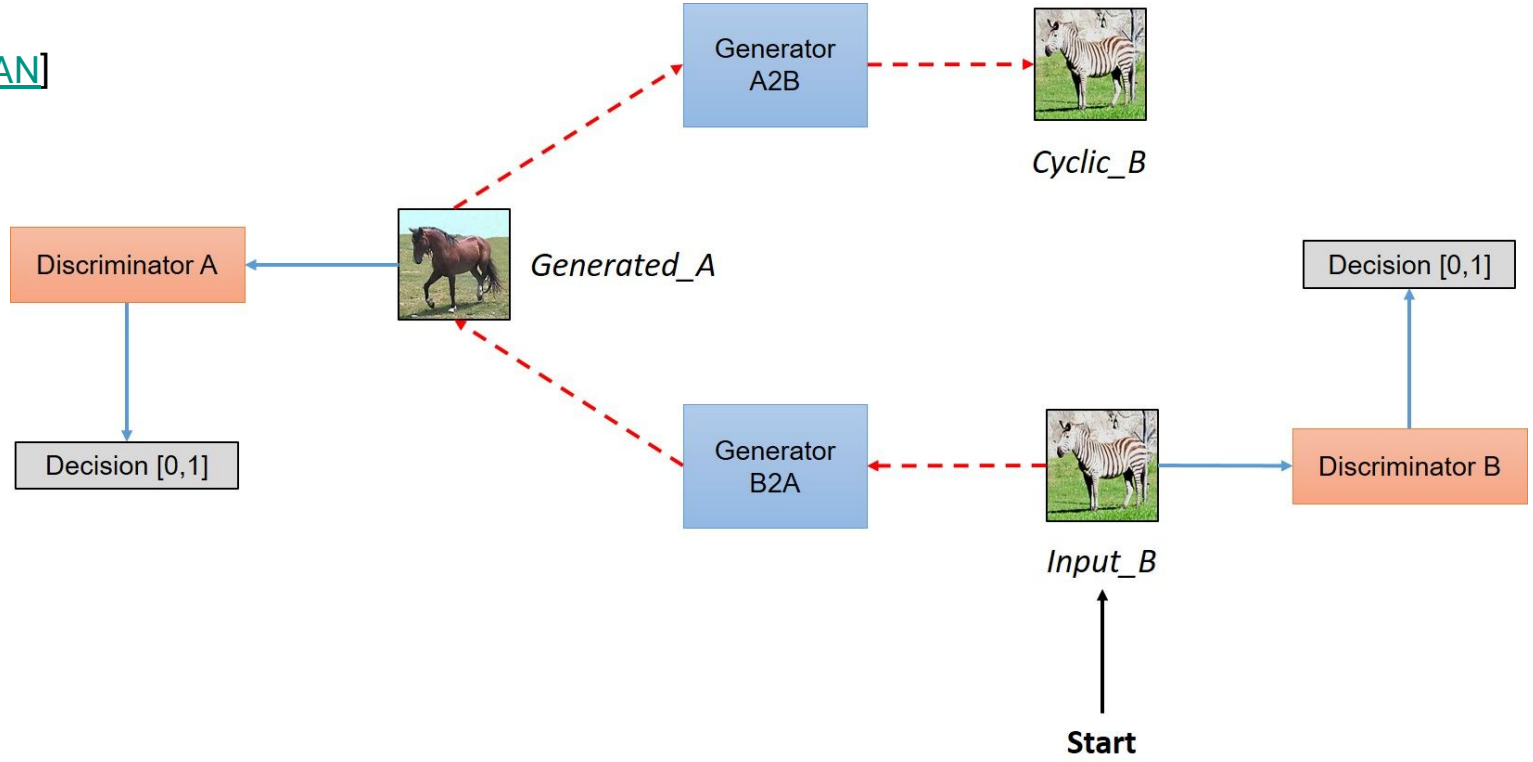
CycleGAN | Zhu ‘17

[CycleGAN]



CycleGAN | Zhu ‘17

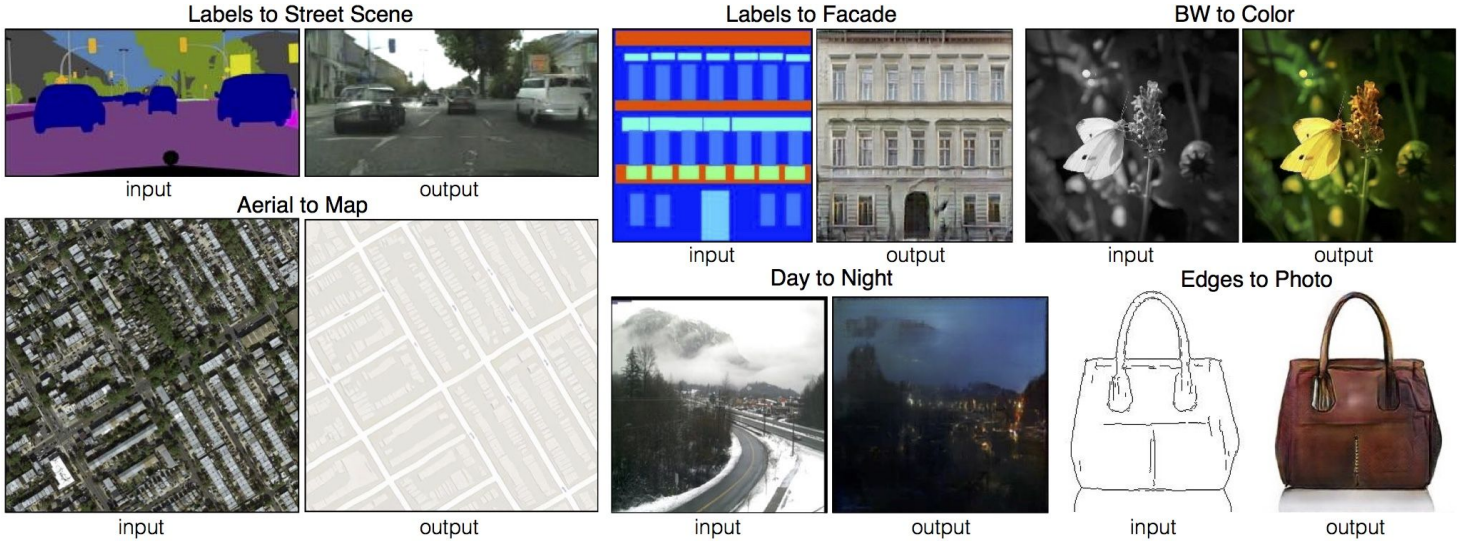
[CycleGAN]



Pix2pix | Isola ‘16

[pix2pix]

Interactive anime



Progressive Growing of GANs

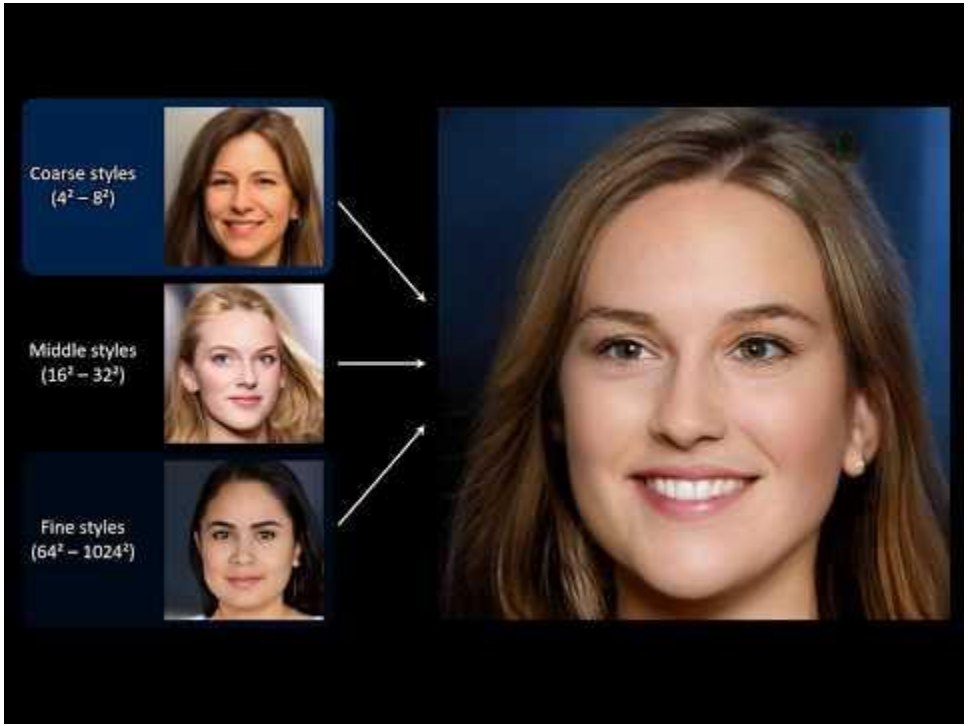
Big GAN | Brock ‘18

[BigGAN]

Dogball?

StyleGAN | Karras ‘19

StyleGAN



FUNIT | Liu ‘19

[FUNIT]

HW1: VAE and GAN

We will be using VAEs and GANs to generate images.

One exciting thing I’m hoping to prepare for you is: **Face Swapping** (e.g. DeepFakes)