INF 473V

INF473V - Modal d'Informatique - Deep Learning in Computer Vision

Announcements

Final Projects

Report:

Due on Friday May 27th at midnight

Penalty of 1 point (out of 20) for every 1 hour delay

Approximate length: 4-6 pages

Suggested content:

- 1 page problem description + method main ideas
- 2 pages your implementation
- 2 pages results (go in detail here)
- 0.5 pages conclusion/extension ideas

Announcements

Final Projects

Presentations:

Monday May 30th – Friday the 3rd of June

Each slot: 30 minutes

(20 minute presentation, 10 minutes Q&A)

Suggested content:

problem description + key ideas of method

what you have done (implementation)

results (most important part)

Last time

- Transfer Learning

- **Object Localization and Detection Networks**
  - Various Problem Formulations
  - General Strategies for Object Localization
  - Overfeat
  - RCNN
  - YoLo

  - Fast RCNN
  - Faster RCNN

Today

Synthesis-based approaches with CNNs

Towards Unsupervised Learning:

- Supervised vs. Unsupervised Learning

Synthesis-based techniques:

- Auto-encoders
- Variational Auto-Encoders
- Generative Adversarial Networks (GANs)
- Conditional GANs

Supervised Learning

- **Data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y

Supervised Learning

- **Data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y

Examples:
Classification,
regression,
object detection,
segmentation,
captioning…

Input:
Image

Output: Assign Image to one of a fixed set of categories

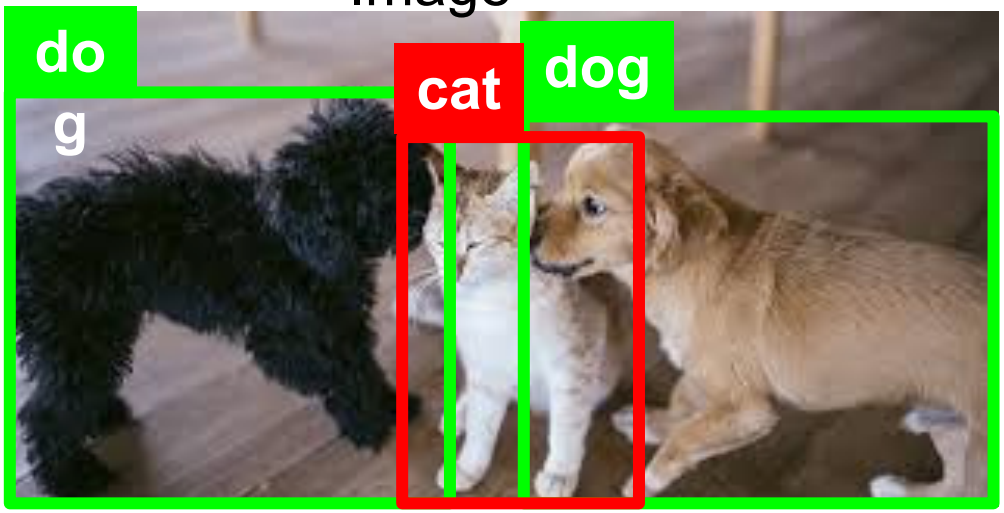
Cat
Dog
Deer
Bird
Car

Supervised Learning

- **Data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y

Examples:
Classification,
regression,
object detection,
segmentation,
captioning…

Input:
Image



Output: Image category (i.e.. class) and location of the object with a rectangle, i.e. bounding box

Supervised Learning

- **Data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y
- **Training:** We are given ground truth (x,y) pairs.

Input: Image

Output: Assign Image to one of a fixed set of categories

Cat
Dog
Deer
Bird
Car

Supervised Learning

- **Training data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y

Unsupervised Learning

- **Training data**: x, where x is data, NO label!
- **Goal**: Learn some underlying hidden structure of the data

Supervised Learning

- **Training data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y

Unsupervised Learning

- **Training data**: x, where x is data, NO label!
- **Goal**: Learn some underlying hidden structure of the data
- **Advantage**: No labelling required

Supervised Learning

- **Data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y

Unsupervised Learning

Examples ??

- **Data**: x, where x is data, NO label!
- **Goal**: Learn some underlying hidden structure of the data
- **Advantage**: No labelling

Supervised Learning

- **Data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y

Unsupervised Learning

- **Data**: x, where x is data, NO label!
- **Goal**: Learn some underlying hidden structure of the data
- **Advantage**: No labelling

Examples:

Clustering

Dim reduction

Feature learning

Density estimation...

Unsupervised Learning

- **Data**: x, where x is data, NO label!
- **Goal**: Learn some underlying hidden structure of the data

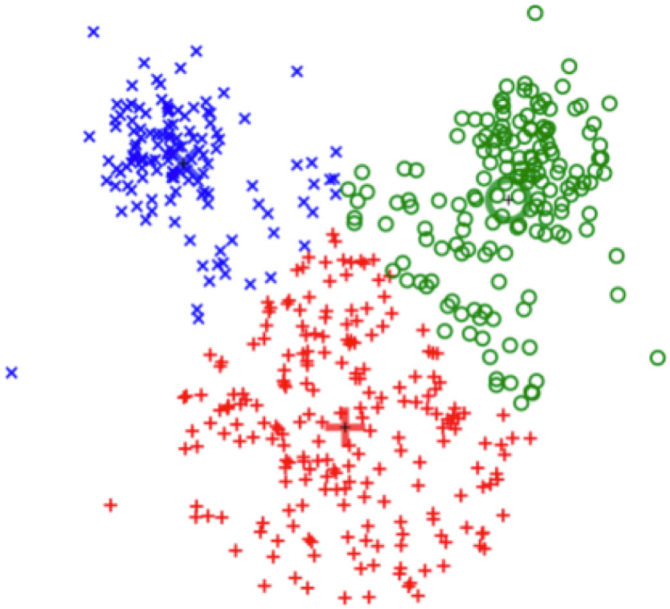
Examples:

Clustering

Dim reduction

Feature learning

Density estimation...



K-Means Clustering

Unsupervised Learning

- **Data**: x, where x is data, NO label!
- **Goal**: Learn some underlying hidden structure of the data

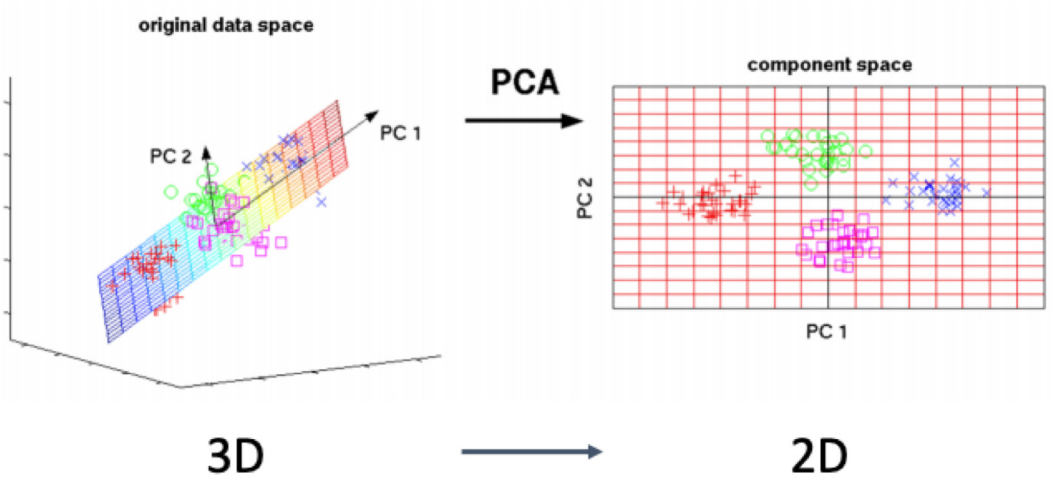
Examples:

Clustering

Dim reduction

Feature learning

Density estimation...



Dimensionality Reduction (e.g. PCA: Principal Components Analysis)

Unsupervised Learning

- **Data**: x, where x is data, NO label!
- **Goal**: Learn some underlying hidden structure of the data

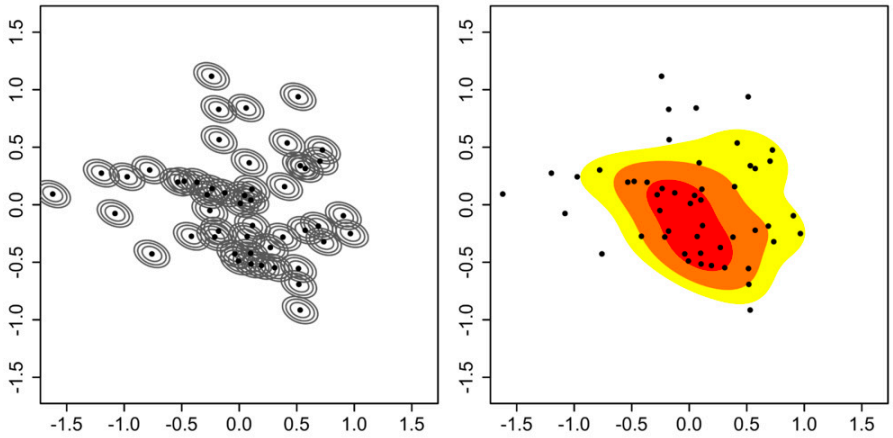
Examples:

Clustering

Dim reduction

Feature learning

Density estimation...



Density Estimation

https://en.wikipedia.org/wiki/Multivariate_kernel_density_estimation

Supervised Learning

- **Data**: (x,y), where x is data, y is label
- **Goal**: Learn a function to map x → y

Examples:
Classification,
regression,
object detection,
segmentation,
captioning…

Unsupervised Learning

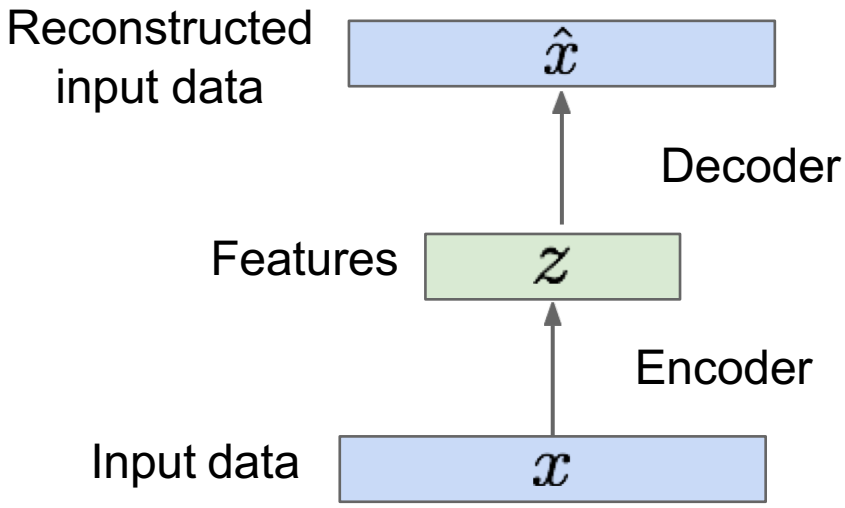
- **Data**: x, where x is data, NO label!
- **Goal**: *Learn some underlying hidden structure of the data*
- **General Questions:** *How can we capture the structure underlying a collection of images? What is the space of "natural" images?*

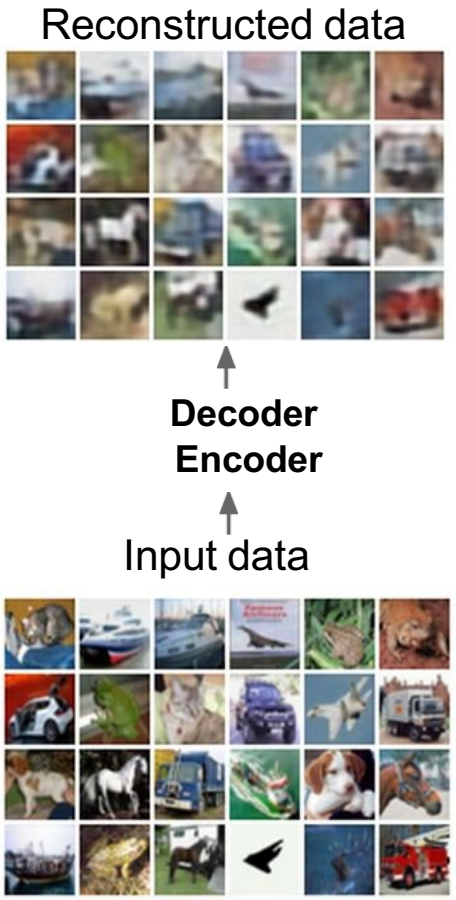
Examples:
Clustering
Dim reduction
Feature learning
Density estimation…

Basic generative model: Autoencoders

How to learn a reduced feature representation?

Train features that can be used to reconstruct the original data "Autoencoding" - encoding itself



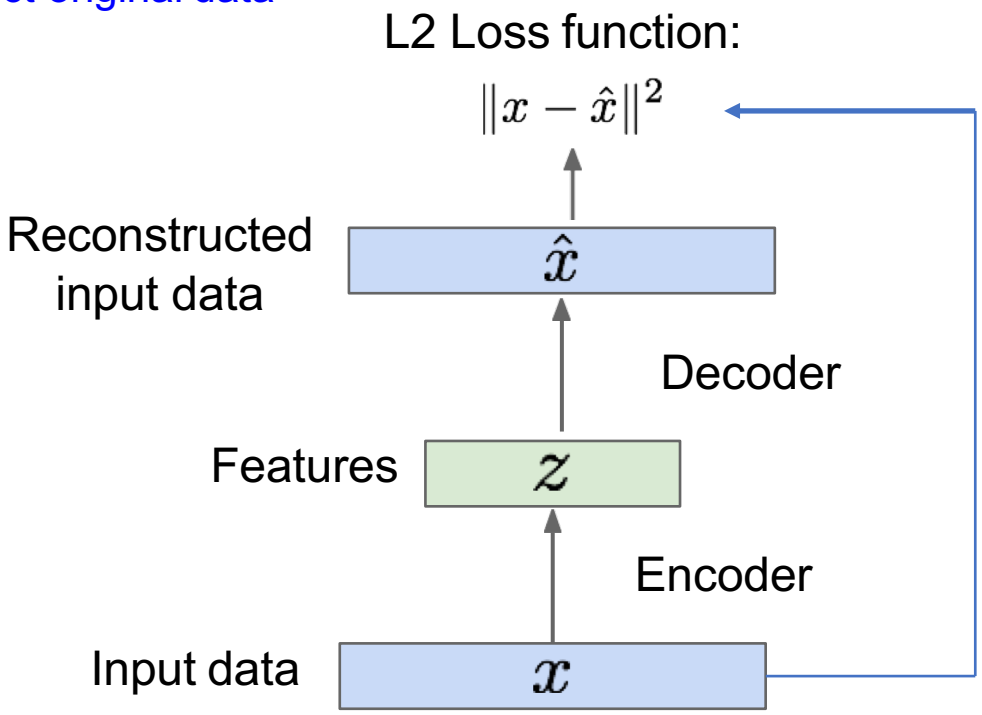


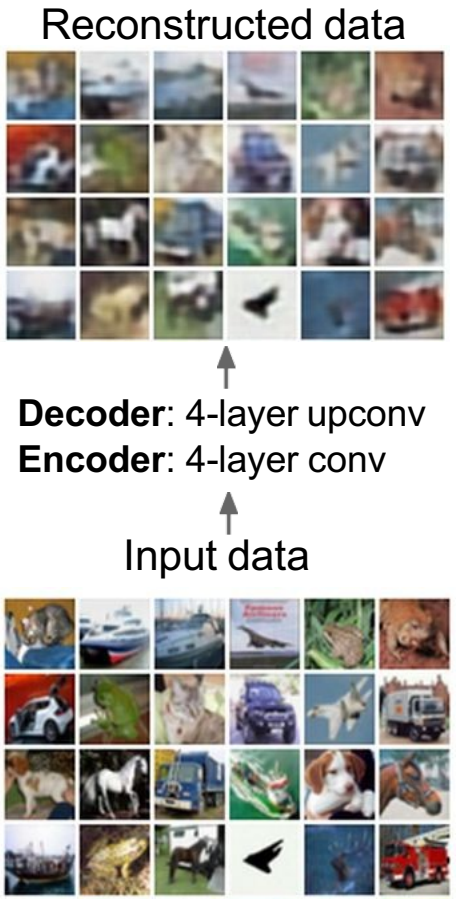
Features are typically (much) lower dimensional than the original data.

Basic generative model: Autoencoders

Train such that features can be used to reconstruct original data

Doesn't use labels!

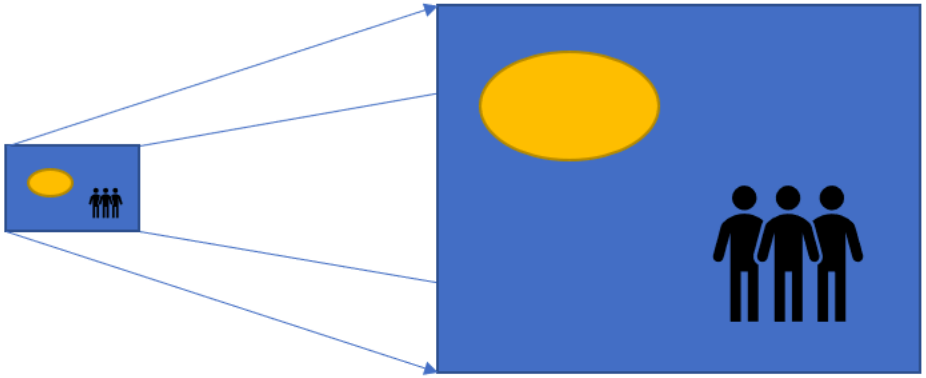




Typical building block for the encoder: a standard **CNN** (think of AlexNet).

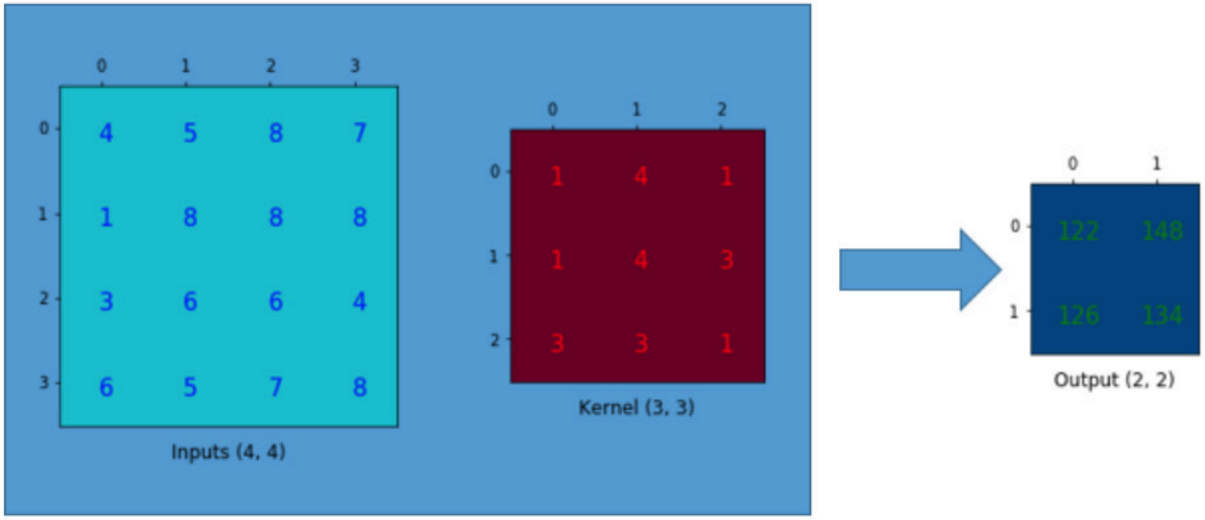
Basic building block: upconvolution

- In a convolutional **decoder** want to upsample from a lower to higher resolution



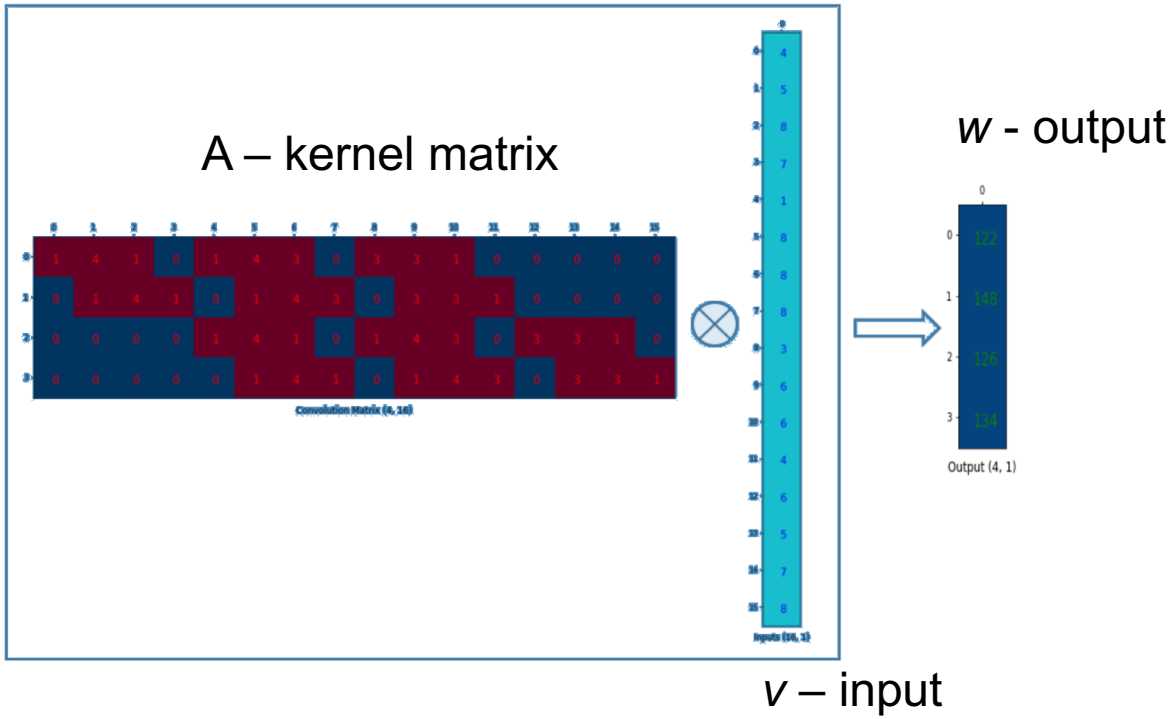
Recall Convolution

- Given a 4x4 input convolution with a 3x3 kernel (stride 1), results in a 2x2 output. Typically, this leads to *down-scaling.*



Recall Convolution

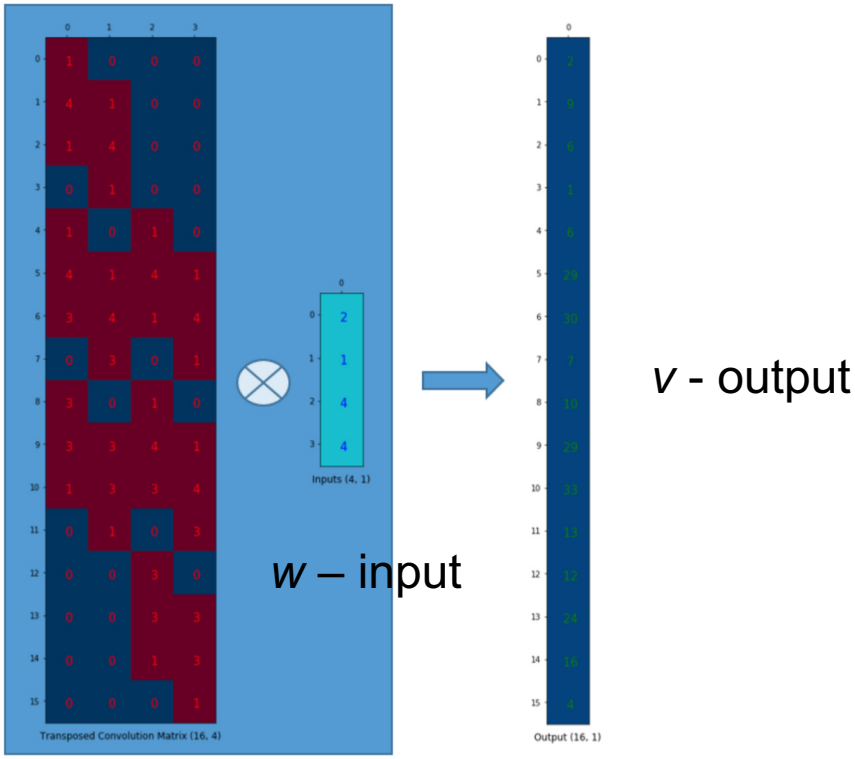
- If we represent the input and output as vectors *v, w* (of size 16 and 4 respectively) then the same operation can be written as matrix-vector product, so that A*v* = *w*



Transposed convolution

- Simply use $v = A^T w$ for upscaling.

A – kernel matrix

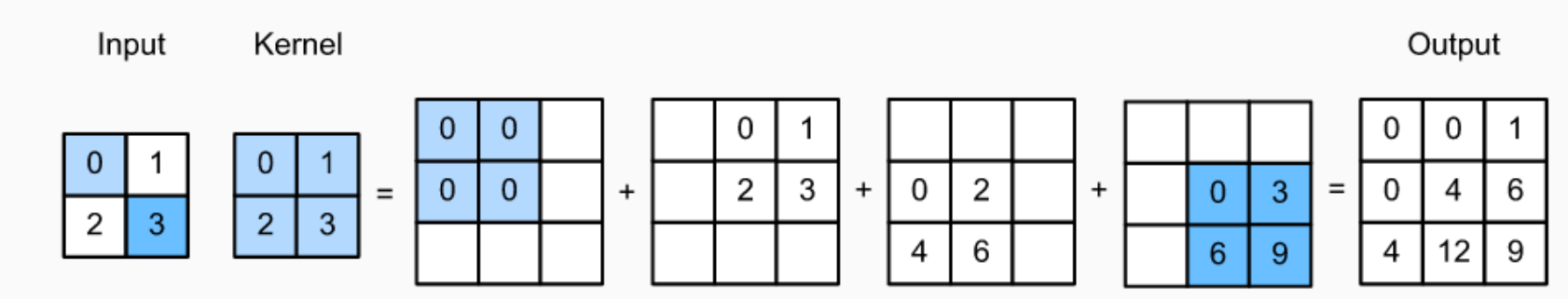


w – input

v - output

Transposed convolution

- Another (equivalent) perspective: instead of multiply, sum and place the result (convolution), multiply and place, then sum (transpose convolution).



Applications of transposed convolution

- Image super-resolution

Ground Truth

¼ Sized Input

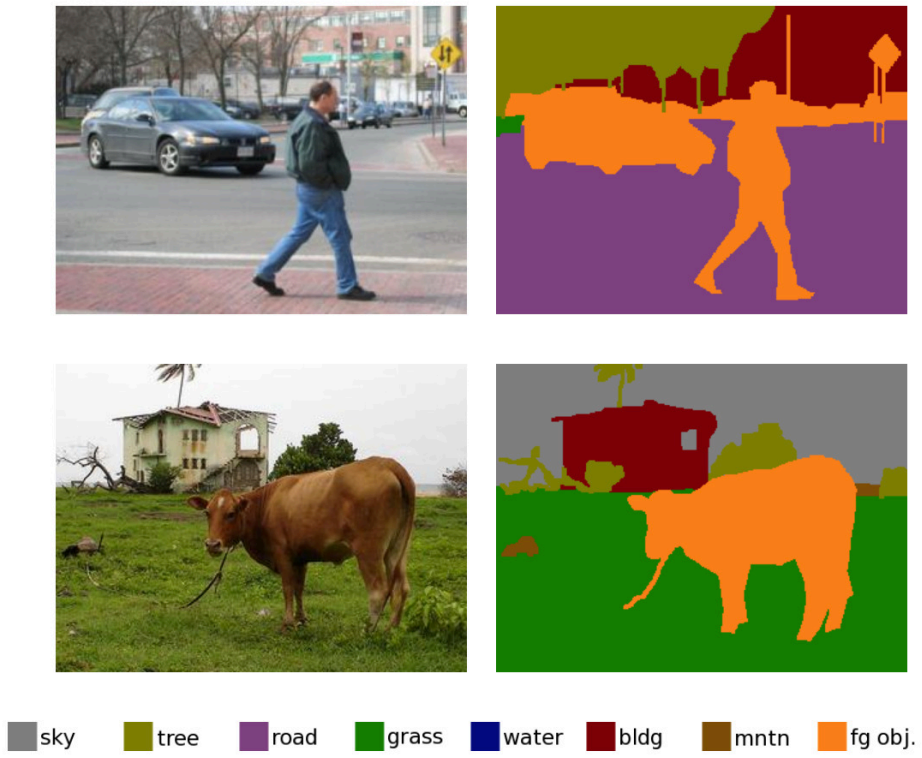
Bicubic

Super Resolution Network

SRFeat: Single Image Super-Resolution with Feature Discrimination, Park et al., ECCV 2018

Applications of transposed convolution

- Semantic Segmentation, Pixel-level labeling, etc.

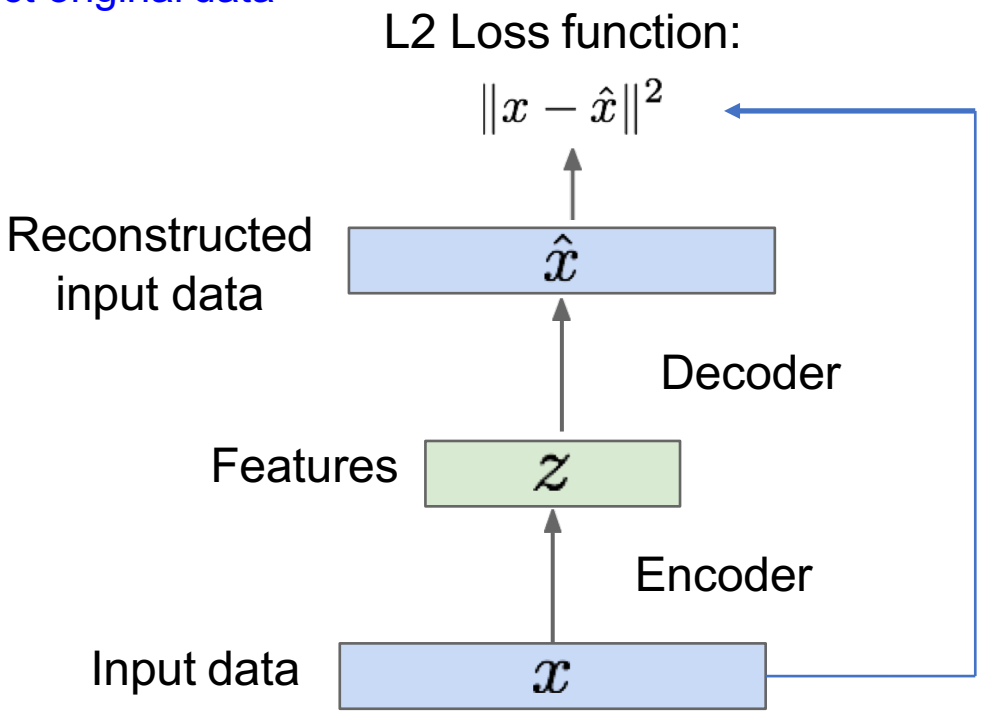


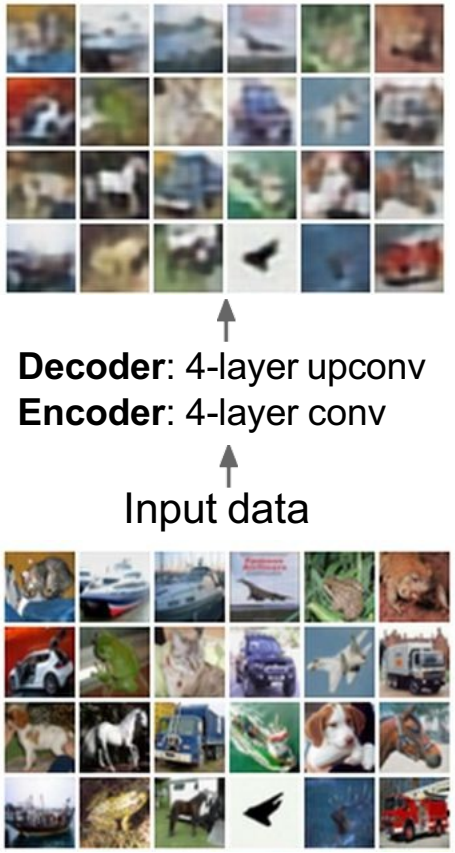
Going beyond the bounding box with semantic segmentation, Chen and Asawa, 2018.

Back to autoencoders

Train such that features can be used to reconstruct original data

Doesn't use labels!

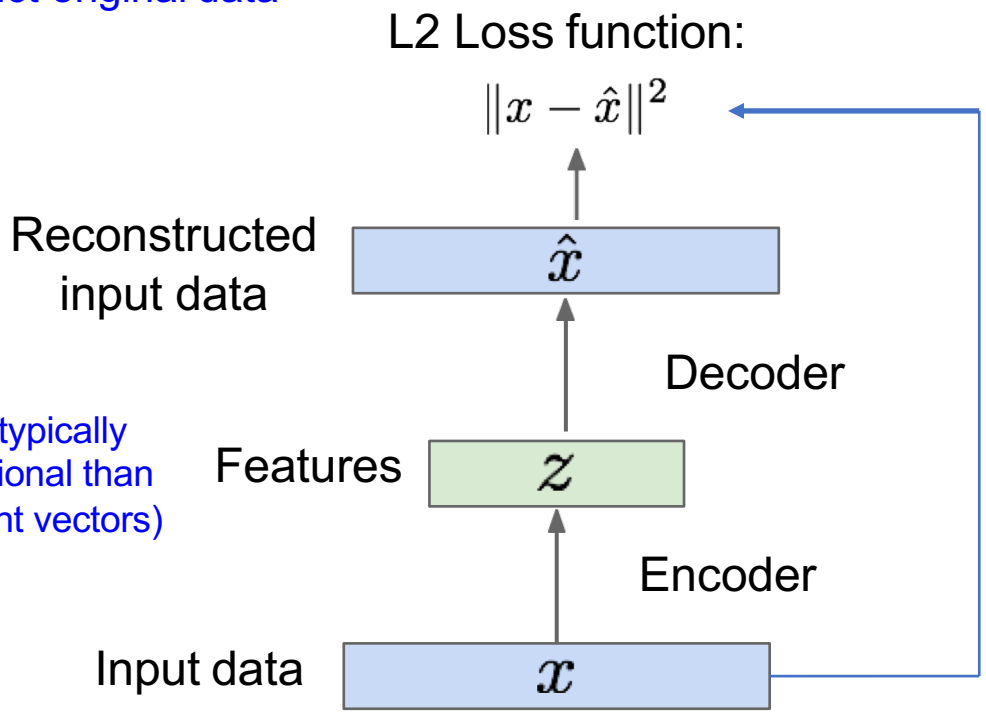




Back to autoencoders

Train such that features can be used to reconstruct original data

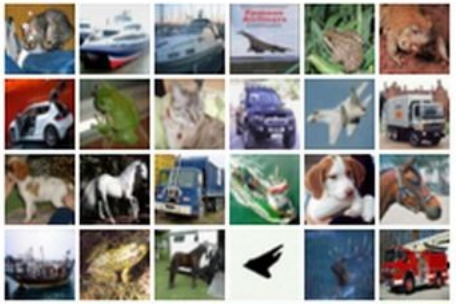
Doesn't use labels!



Features are typically lower dimensional than the data (latent vectors)

Decoder: 4-layer upconv
Encoder: 4-layer conv

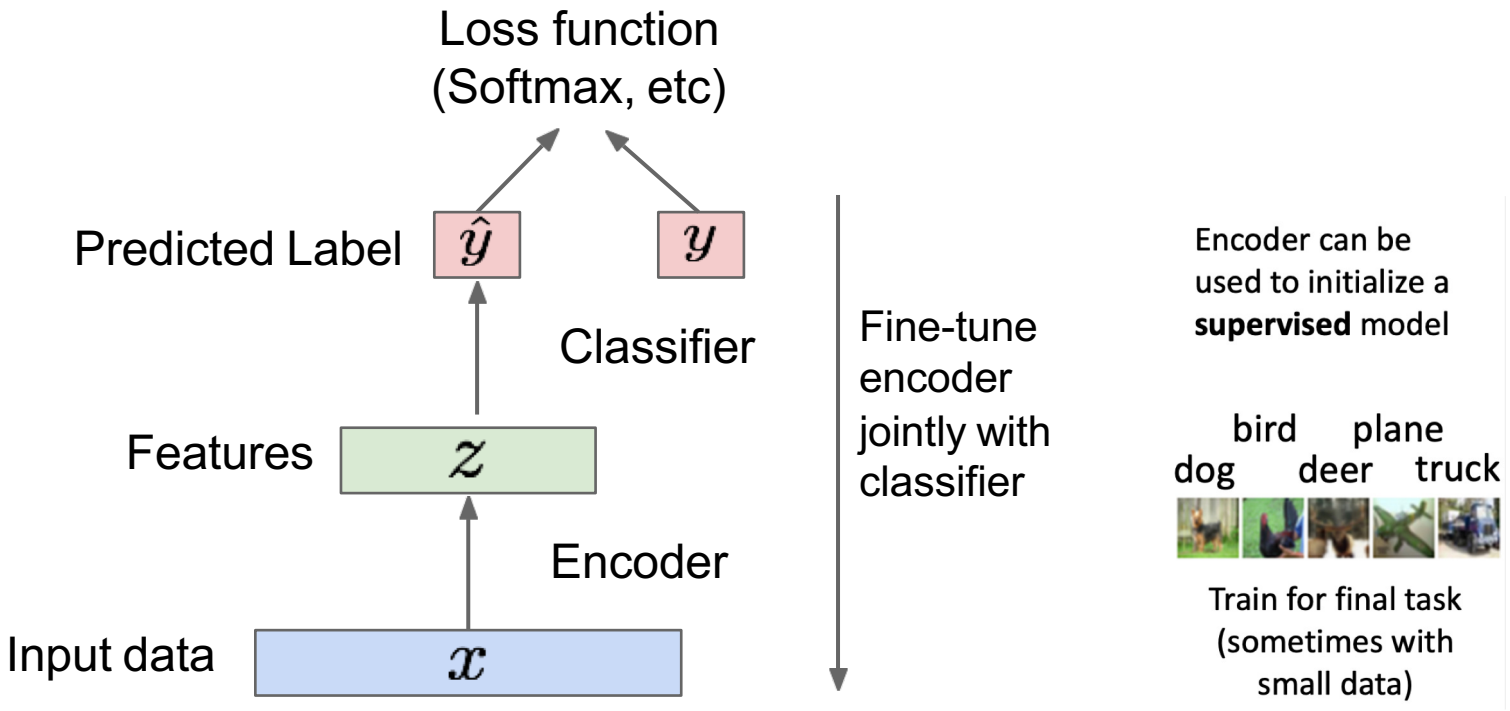
Input data



Applications of autoencoders

After training:

Ignore the decoder and use the *encoder* for a downstream task

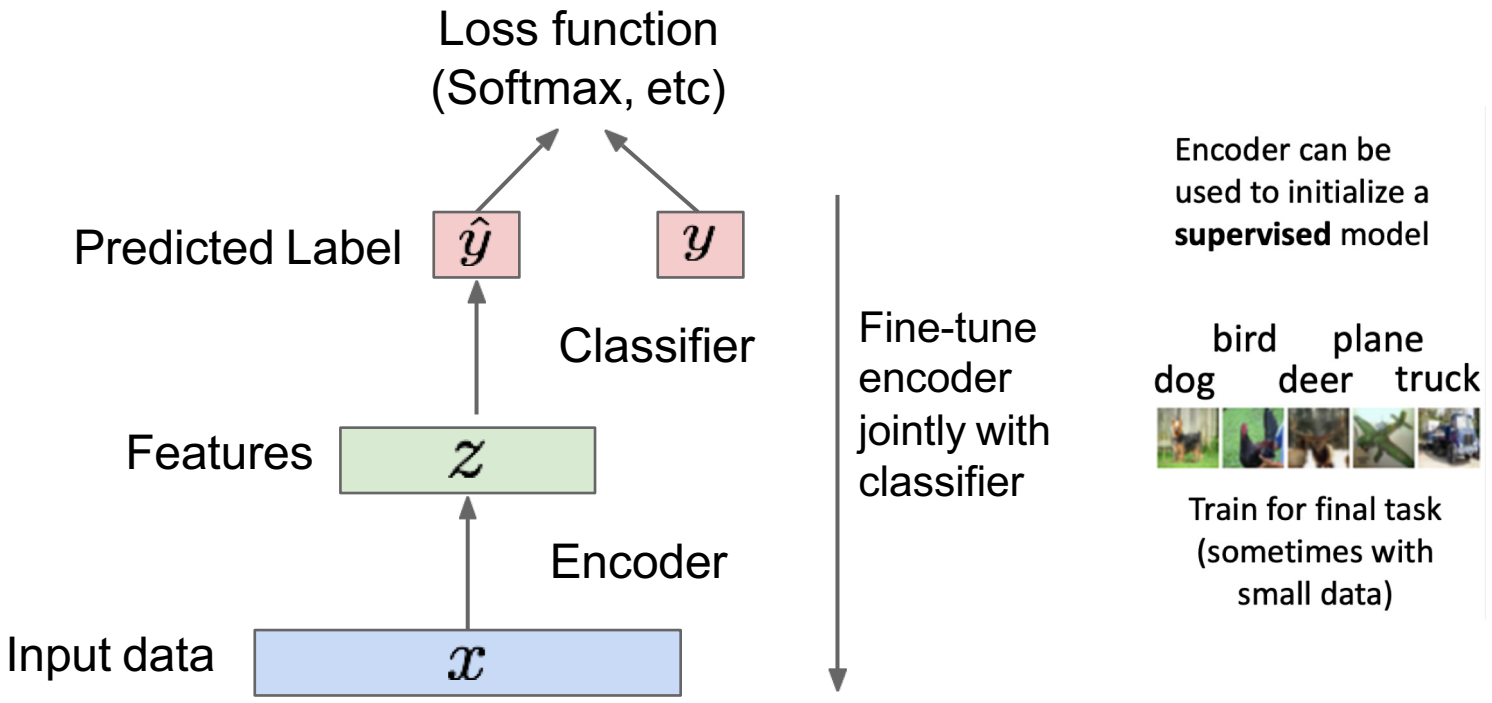


Applications of autoencoders

Encoder can learn a good latent (feature) representation, which can then be used:

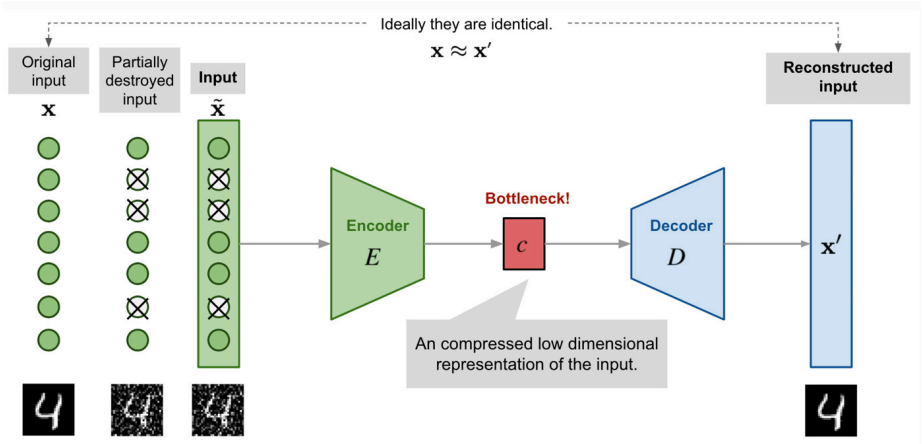
- Pre-training a classifier (with limited labeled data)
- Dimensionality reduction

…

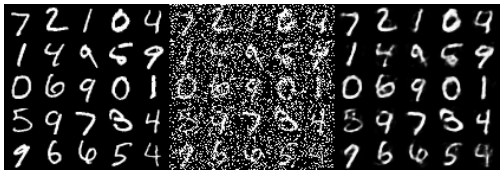


Other applications of autoencoders

- Unsupervised feature learning (e.g., for pre-training a classifier)
- Denoising, image auto-completion
- Compression?



Denoising auto-encoders



Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." Proc. ICML 2008.

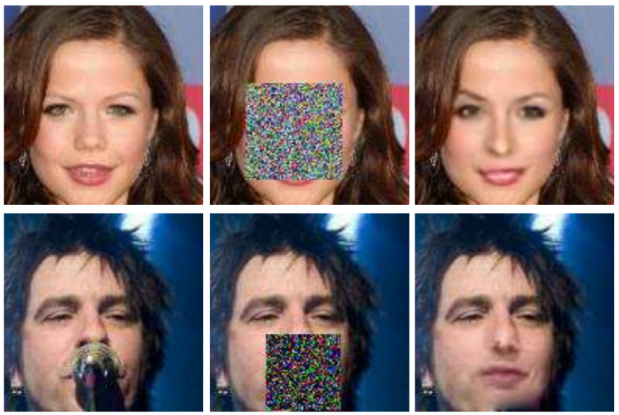
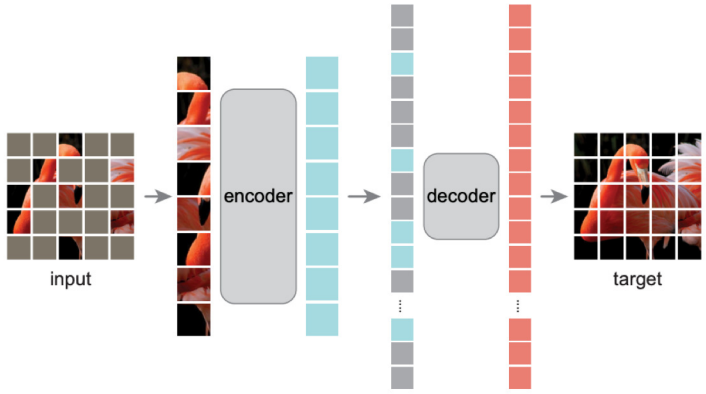


Image auto-completion

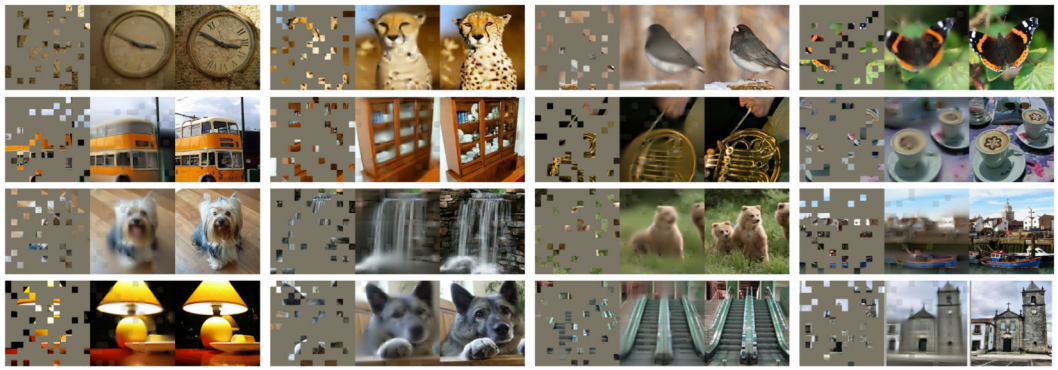
Li, Yijun, et al. "Generative face completion." Proc. CVPR, 2017.

Other applications of autoencoders

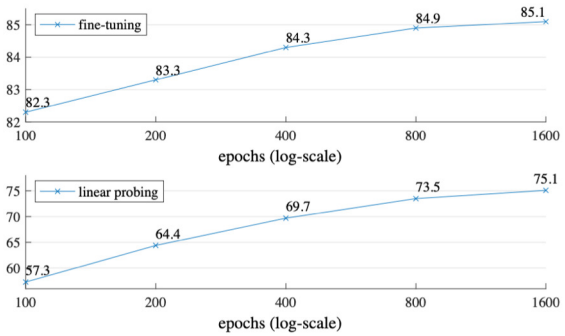
- Masked Autoencoders Are Scalable Vision Learners



Learn image features by masking large portions (75%) of the input and trying to reconstruct the clean image.



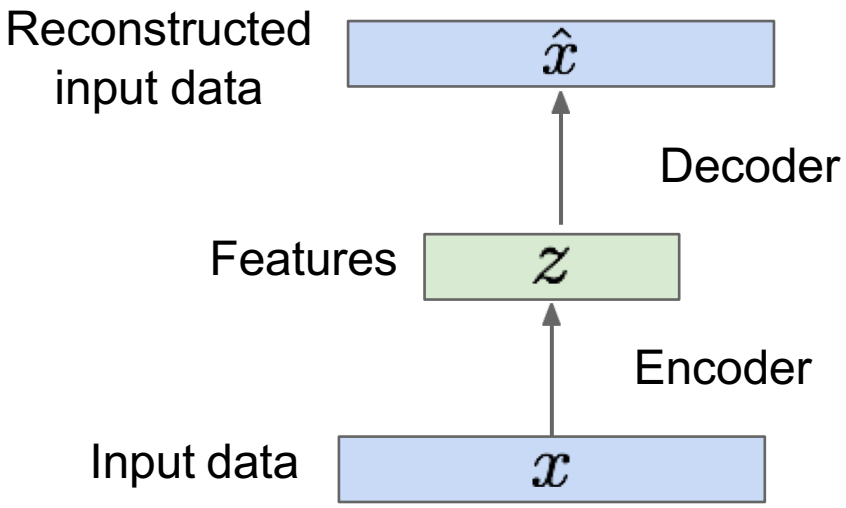
Leads to state-of-the-art unsupervised feature extractor



Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. “Masked autoencoders are scalable vision learners.” arXiv preprint arXiv:2111.06377, 2021.

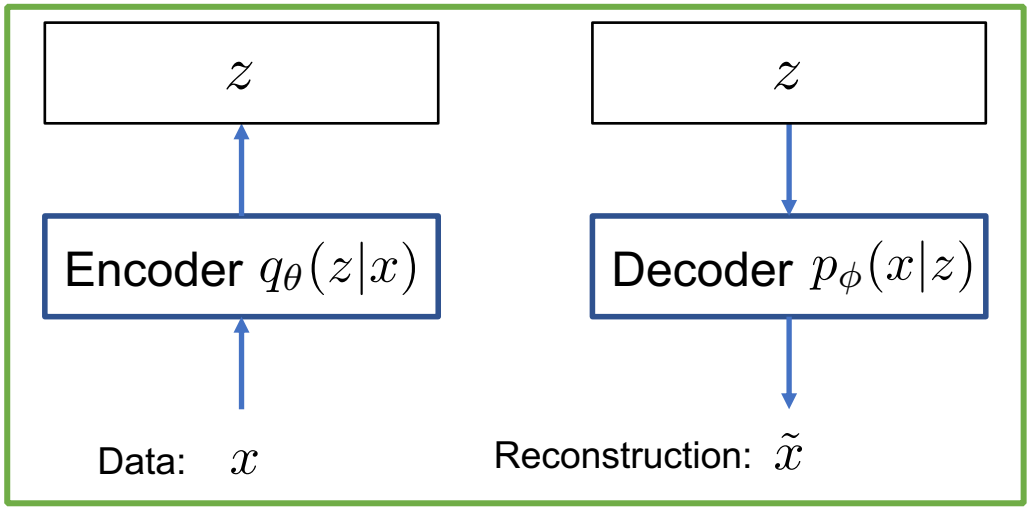
Autoencoders summary

- Autoencoders learn **latent features** for data without any labels!
- Can use features to initialize a **supervised** model
- Not probabilistic: No way to sample new data from learned model (no way to generate realistic new images).



Variational Autoencoder (VAE)

- **Key idea:** make both the encoder and the decoder probabilistic.
- I.e., the latent variables, z, are drawn from a probability distribution depending on the input, X, and the reconstruction is chosen probabilistically from z.

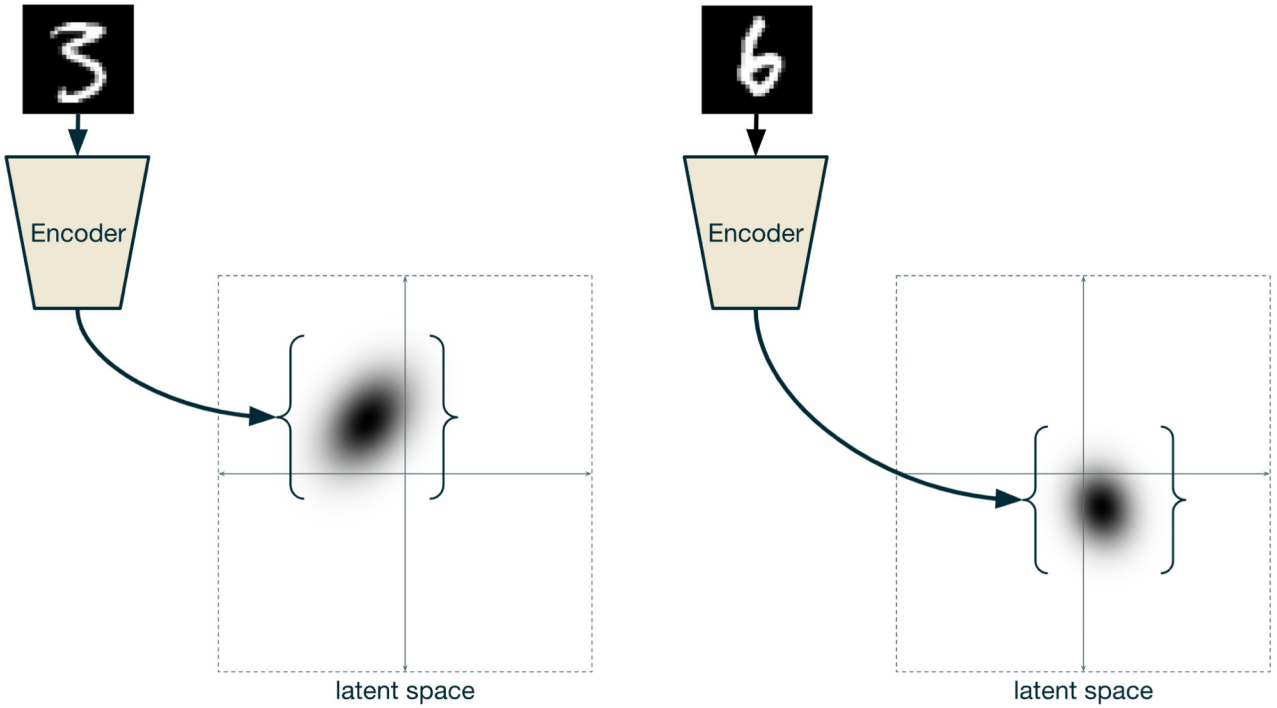


https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Diederik P. Kingma, Max Welling: *Auto-Encoding Variational Bayes*. ICLR 2014

Variational Autoencoder (VAE)

Encoder structure



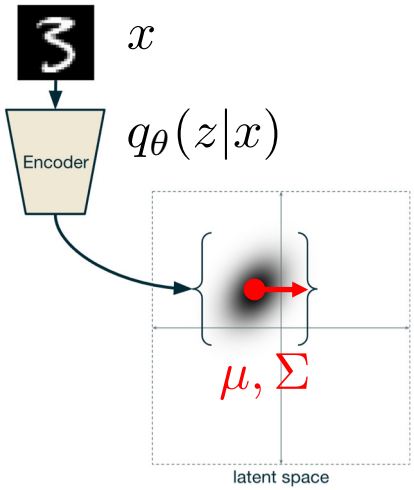
https://ijdykeman.github.io/ml/2016/12/21/cvae.html

VAE Encoder

- The encoder takes input and returns parameters for a probability density (e.g., Gaussian): I.e., gives the mean and co-variance matrix.

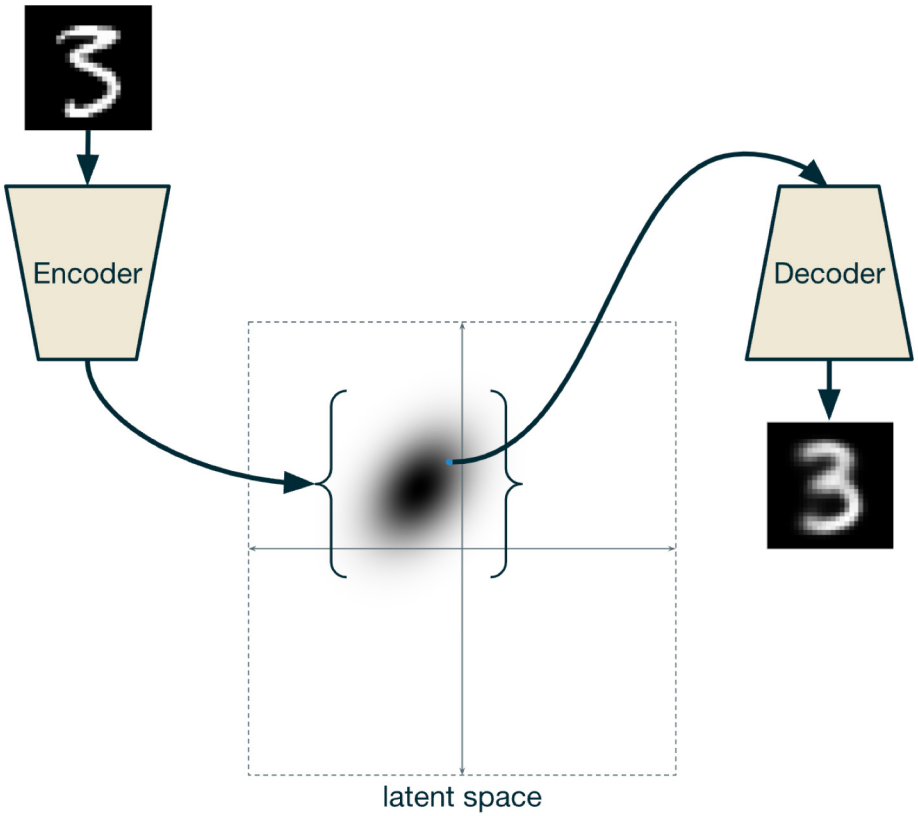
$$q_\theta(z|x) = \mu, \Sigma$$

- Implemented via a neural network: each input *x* gives a vector mean and a *diagonal covariance matrix* that determine the Gaussian density $\mathcal{N}(\mu, \Sigma)$.
- We can then *sample from this distribution* to get random values of the lower-dimensional representation *z*.
- Parameters θ for the NN need to be learned: need to set up a loss function.



https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

Variational Autoencoder (VAE)

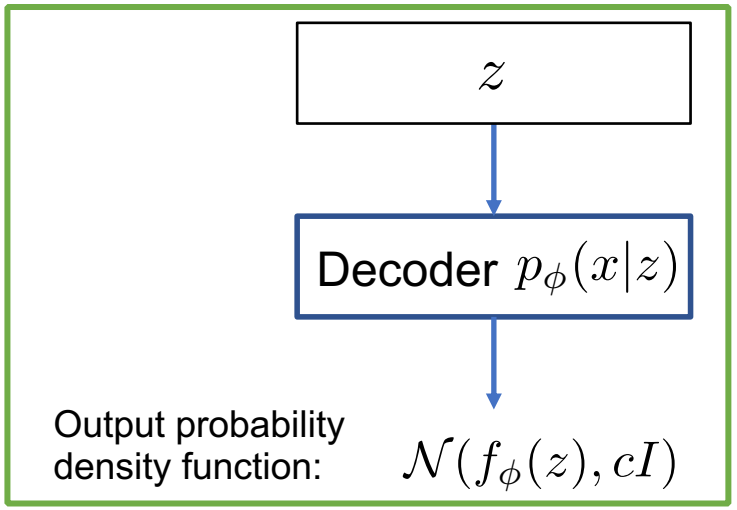


https://jaan.io/what-is-variational-autoencoder-vae-tutorial/

VAE Decoder

- The decoder takes latent variable *z* and returns parameters for a distribution. E.g., $p_\phi(x|z)$ gives the mean (and possibly variance) for each pixel in the output.

- In the simplest case we assume: $p_\phi(x|z) = \mathcal{N}(f_\phi(z), cI)$ where f_ϕ is the (deterministic) decoder network with trainable parameters ϕ.



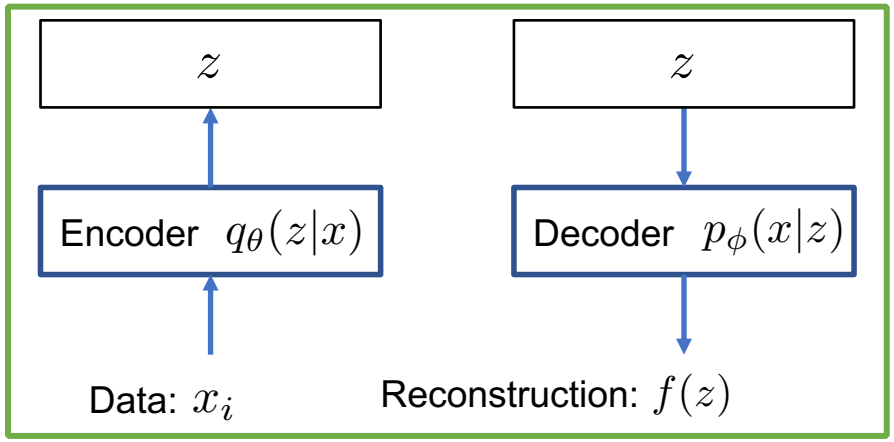
VAE loss function

- **Question:** how to train the networks q_θ, p_ϕ ?

- First, we would like to *maximize the likelihood* of the input, x_i under the predicted density $p_\phi(x|z) = \mathcal{N}(f_\phi(z), cI)$:

- Same as *minimizing* the negative log likelihood:

$$-\log p_\phi(x_i|z) = \frac{\|x_i - f(z)\|^2}{2c}$$



VAE loss function

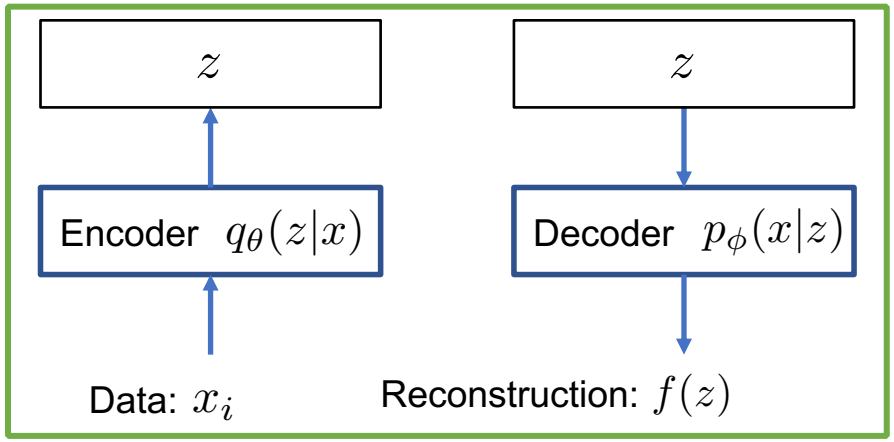
- **Question:** how to train the networks q_θ, p_ϕ ?

- First, we would like to *maximize the likelihood* of the input, x_i under the predicted density $p_\phi(x|z) = \mathcal{N}(f_\phi(z), cI)$.

- Second, we would like predicted distribution in the latent space to be well-structured:

$$\mathbb{KL}(q_\theta(z|x_i) \;||\; \text{prior}(z))$$

typically $\text{prior}(z) = \mathcal{N}(0, I)$. KL-divergence has closed-form expression.



VAE loss function

- Overall, for a single data point x_i we get the loss function

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + \mathbb{KL}(q_\theta(z|x_i) \;||\; \text{prior}(z))$$

- The first term promotes recovery of the input.
- The second term keeps the encoding continuous, by promoting its similarity to the prior, regardless of the input. This helps to structure the latent space and inhibits memorization.
- This loss function is called ELBO (Evidence Lower BOund), and can be derived from the log likelihood: $\log p(x_i)$
- With this loss function the VAE can (almost) be trained using gradient descent on minibatches.

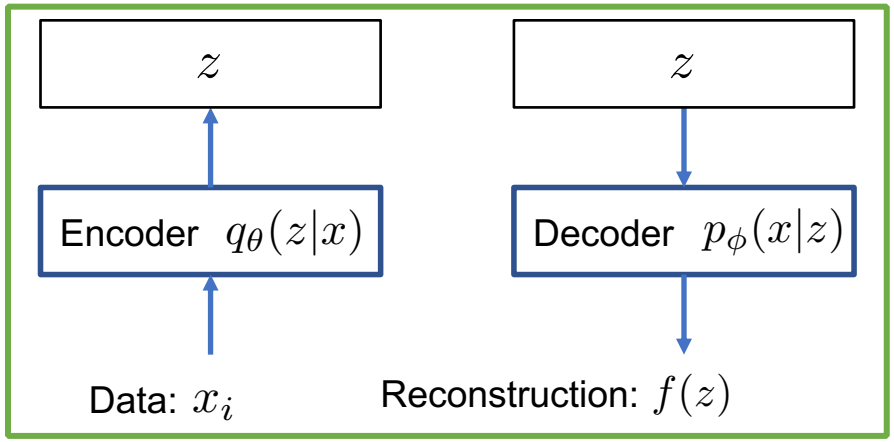
Odaibo, Stephen. "Tutorial: Deriving the standard variational autoencoder (VAE) loss function." arXiv preprint arXiv:1907.08956 (2019).

VAE loss function

- For a single data point x_i we get the loss function

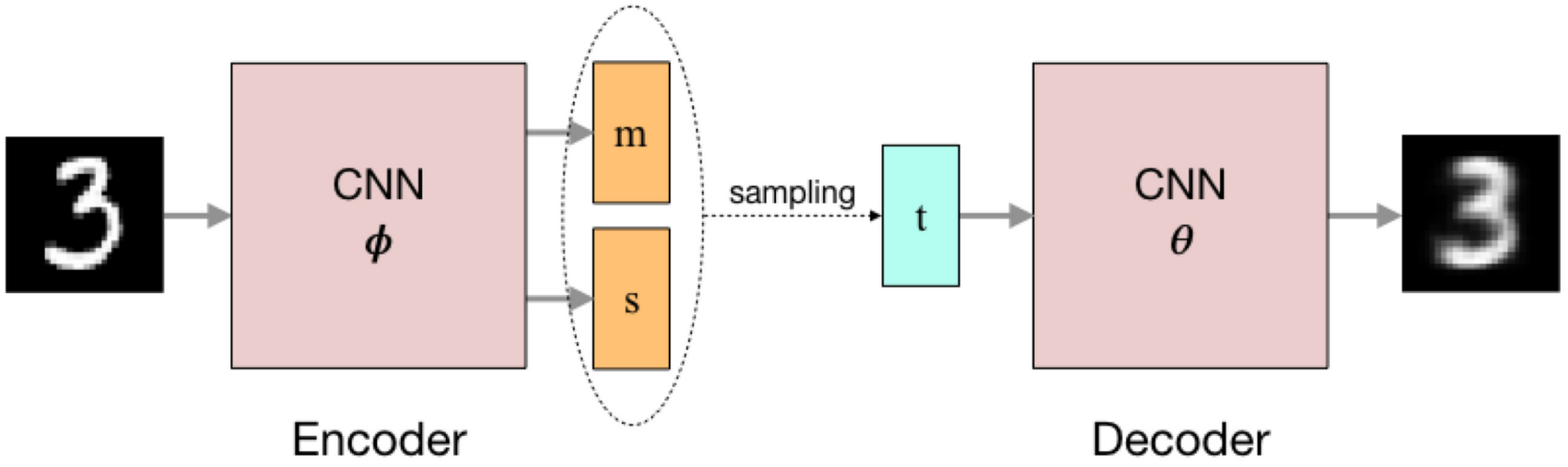
$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + \mathbb{KL}(q_\theta(z|x_i) \;||\; \text{prior}(z))$$

- **Problem:** The expectation would usually be approximated by sampling from $q_\theta(z|x)$ and averaging. This is not differentiable wrt θ and ϕ.



VAE loss function

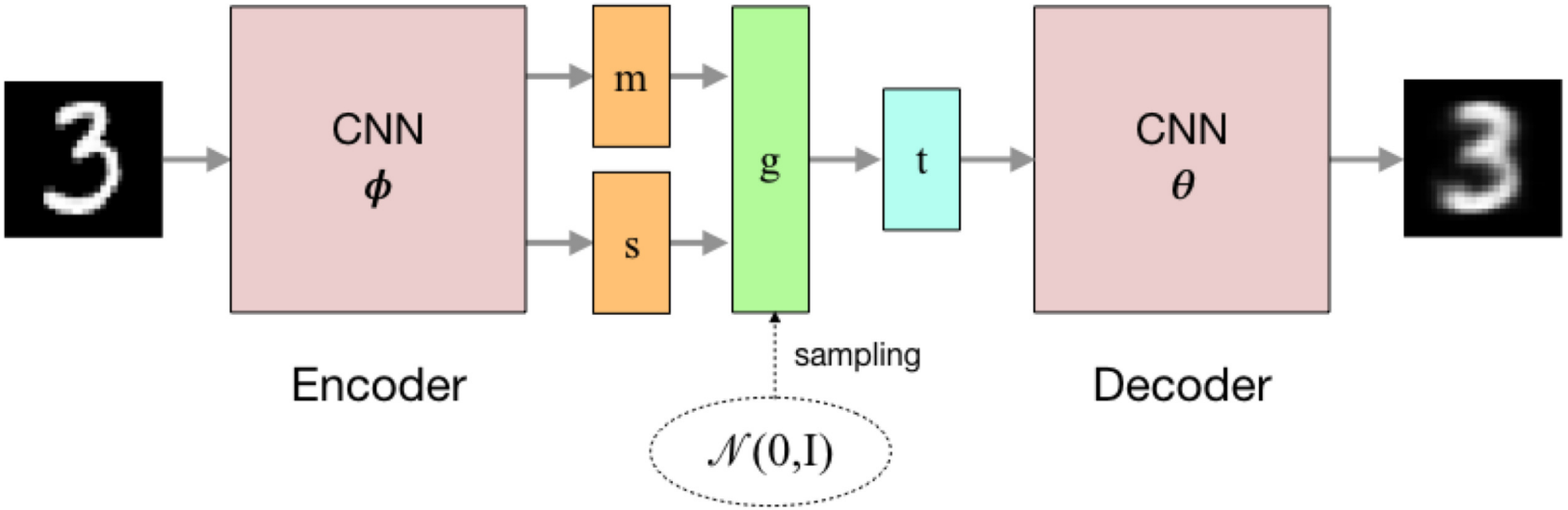
- **Problem:** The expectation would usually be approximated by choosing samples and averaging. This is not differentiable wrt θ and ϕ.



https://nbviewer.jupyter.org/github/krasserm/bayesian-machine-learning/blob/master/variational_autoencoder.ipynb

VAE loss function

- **Reparameterization:** If *z* is $N(\mu(x_i), \Sigma(x_i))$, then we can sample *z* using $z = \mu(x_i) + \sqrt{(\Sigma(x_i))}\,\epsilon$, where ϵ is *N*(0,1). So we can draw samples from *N*(0,1), which doesn't depend on the parameters.



https://nbviewer.jupyter.org/github/krasserm/bayesian-machine-learning/blob/master/variational_autoencoder.ipynb

VAE generative model

- After training, $q_\theta(z|x_i)$ is close to a standard normal, *N*(0,1) – easy to sample.

- Using a sample of *z* from $q_\theta(z|x_i)$ gives an approximate reconstruction of x_i, at least in expectation.

- If we sample any *z* from *N(0,1)* and use it as input to $p_\phi(x|z)$ we can approximate the entire data distribution *p*(*x*). I.e., we can generate new plausible samples.

Variational Autoencoders: generating data

32x32 CIFAR-10

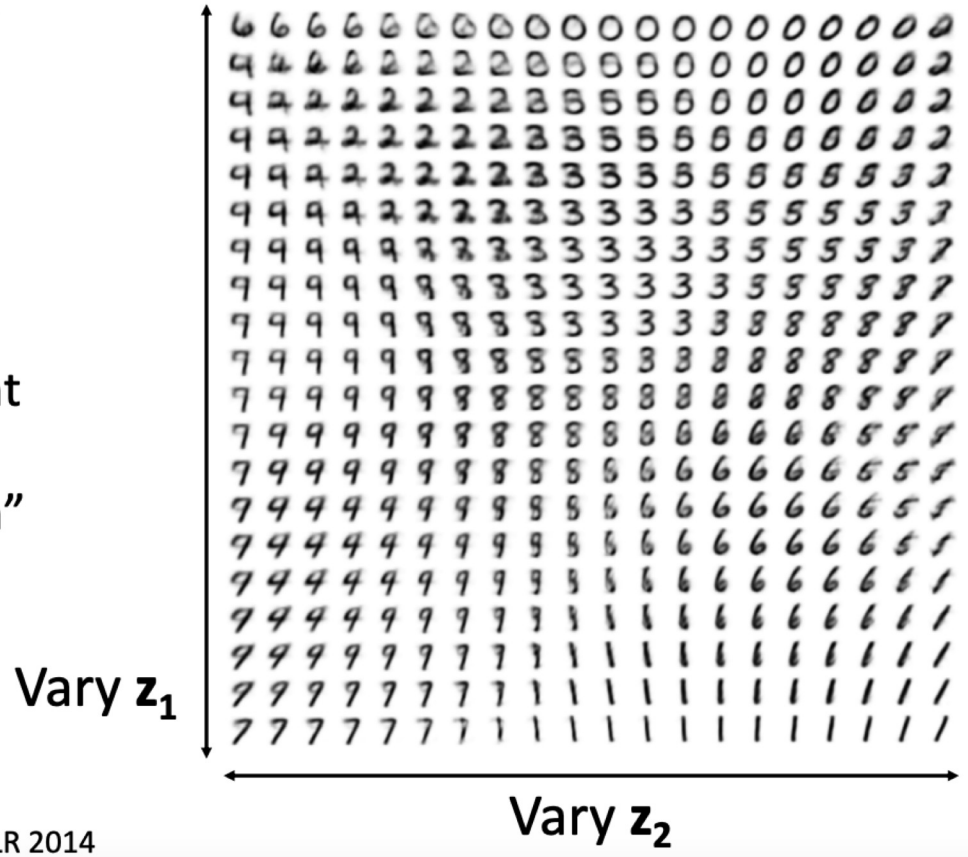
Labeled Faces in the Wild

Figures from (L) Dirk Kingma et al. 2016; (R) Anders Larsen et al. 2017.

Variational Autoencoders: generating data

The diagonal prior on p(z) causes dimensions of z to be independent

"Disentangling factors of variation"

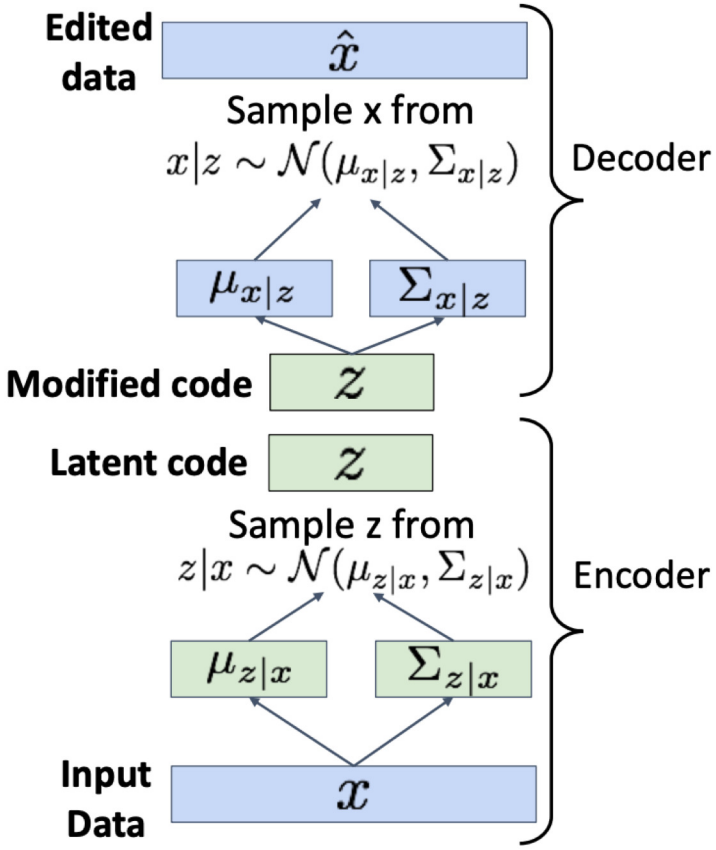


Kingma and Welling, Auto-Encoding Variational Beyes, ICLR 2014

Variational Autoencoders: generating data

After training we can **edit images**

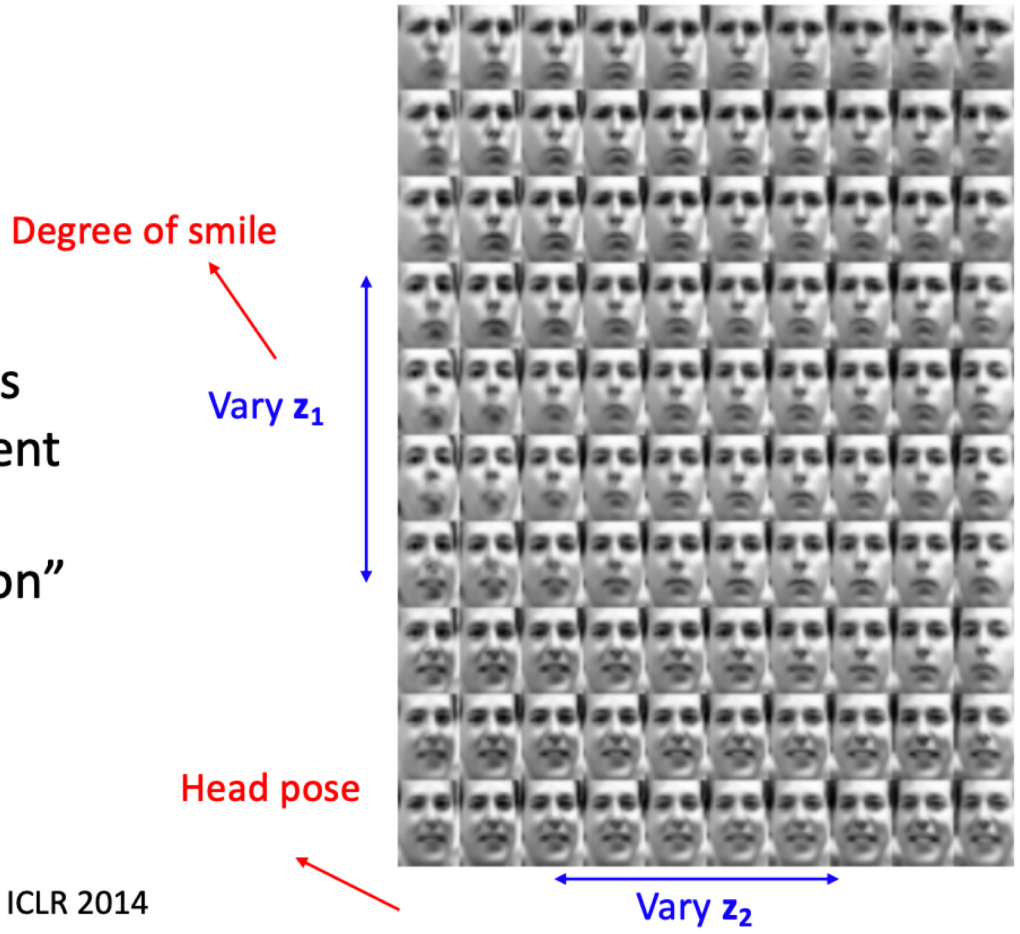
1. Run input data through **encoder** to get a distribution over latent codes
2. Sample code z from encoder output
3. Modify some dimensions of sampled code
4. Run modified z through **decoder** to get a distribution over data samples
5. Sample new data from (4)



Variational Autoencoders: generating data

The diagonal prior on p(z) causes dimensions of z to be independent

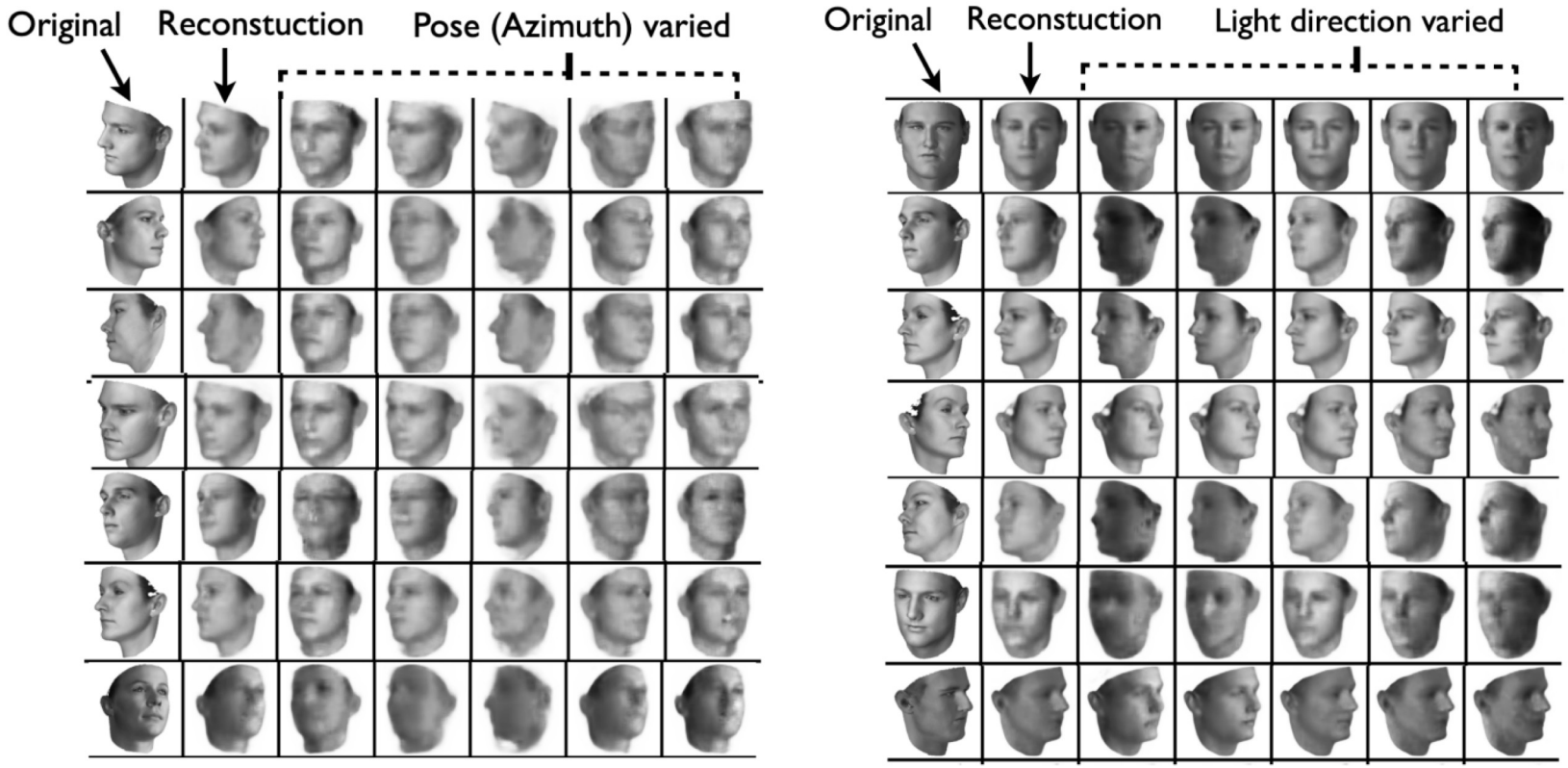
"Disentangling factors of variation"



Kingma and Welling, Auto-Encoding Variational Beyes, ICLR 2014

Variational Autoencoders: generating data

Variational Autoencoders: Image Editing



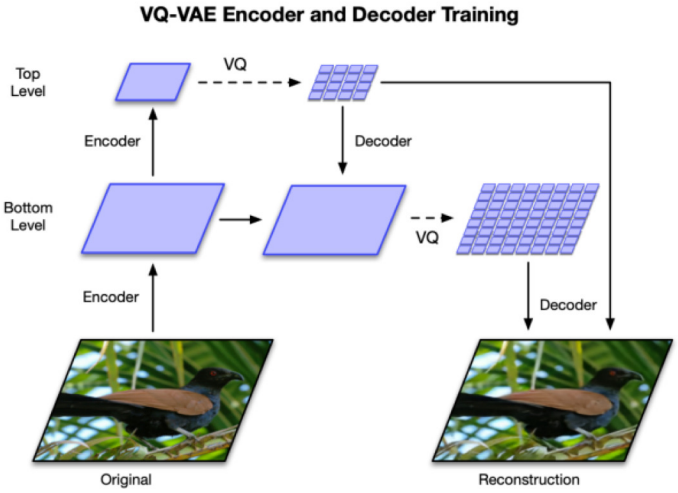
Kulkarni et al, "Deep Convolutional Inverse Graphics Networks", NeurIPS 2014

Variational Autoencoders: generating data

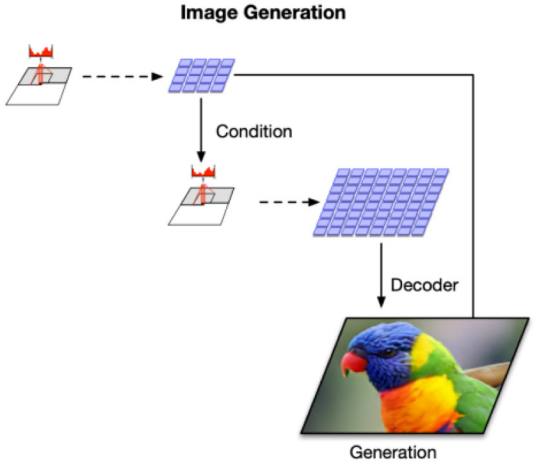
Recent variants of VAEs:

Vector-Quantized Variational Autoencoder (VQ-VAE2)

Train a VAE-like model to generate multiscale grids of latent codes



Use a multiscale PixelCNN to sample in latent code space



Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2", NeurIPS 2019

Variational Autoencoders: generating data

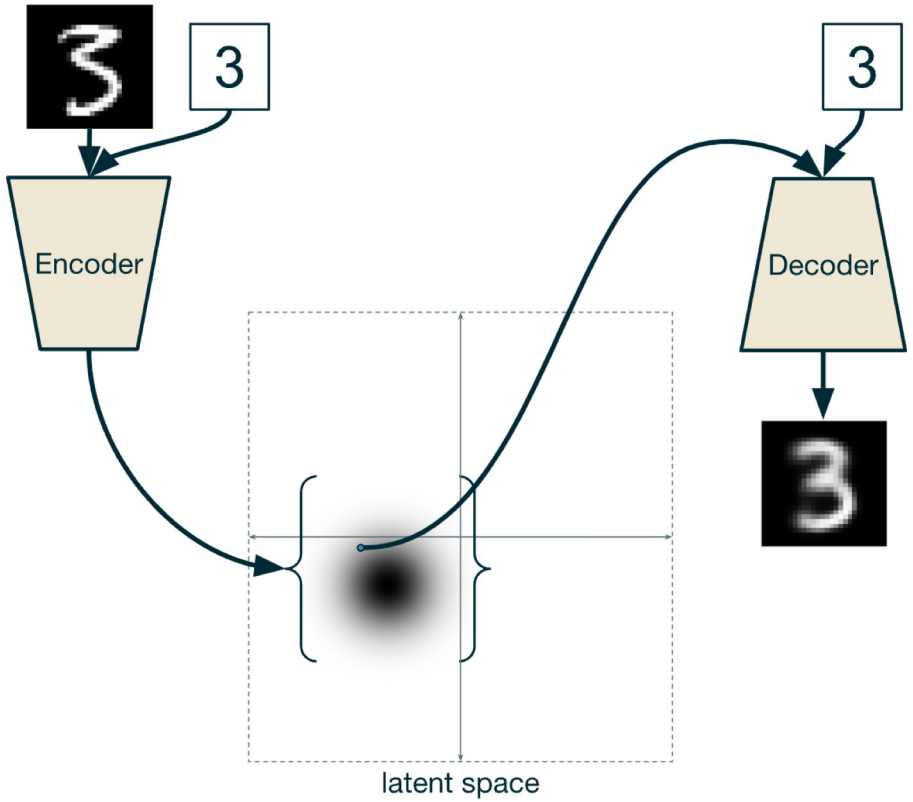
Vector-Quantized Variational Autoencoder (VQ-VAE2)

256 x 256 class-conditional samples, trained on ImageNet

Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2", NeurIPS 2019

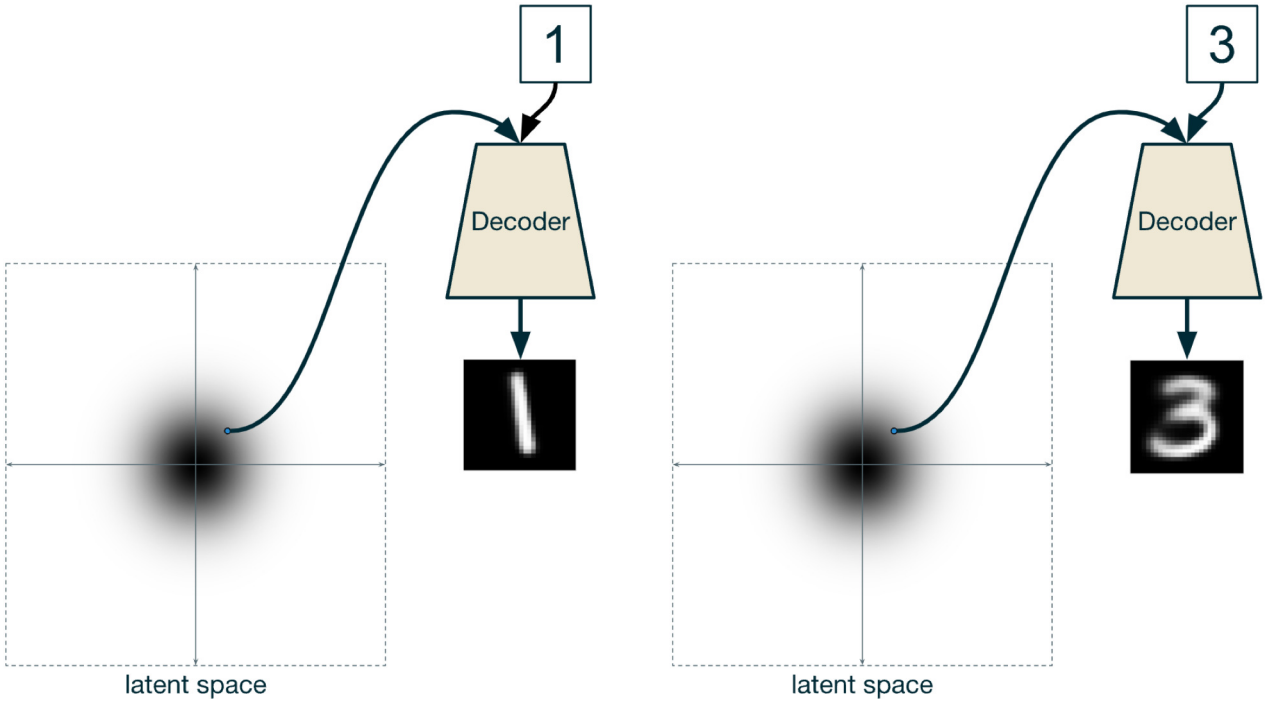
Conditional Variational Autoencoders

Inject the class label to both the encoder and decoder (as side input).



Conditional Variational Autoencoders

Allows to generate different outputs, even for the same latent vectors
(latent vectors start controlling the *style* or *appearance* of the object)



Conditional Variational Autoencoders

Allows to generate different outputs, even for the same latent vectors
(latent vectors start controlling the *style* or *appearance* of the object)

Fixing the class and modifying the latent vector

https://ijdykeman.github.io/ml/2016/12/21/cvae.html

Variational Autoencoders: generating data

Conditional image generation with VAEs : VQ-VAE2

256 x 256 class-conditional samples, trained on ImageNet

Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2", NeurIPS 2019

Variational Autoencoders: generating data

1024 x 1024 generated faces, trained on FFHQ

Razavi et al, "Generating Diverse High-Fidelity Images with VQ-VAE-2", NeurIPS 2019

Variational Autoencoders: generating data

Variational Autoencoder: Summary

Probabilistic spin to traditional autoencoders => allows generating data
Defines an intractable density => derive and optimize a (variational) lower bound

Pros:

- Principled approach to generative models
- Allows inference of q(z|x), can be useful feature representation for other tasks

Cons:

- Maximizes lower bound of likelihood: okay, but only *approximate* reconstruction.
- Samples blurrier and lower quality compared to state-of-the-art (GANs)

Active areas of research:

- More flexible approximations, e.g. richer approximate posterior instead of diagonal Gaussian, e.g., Gaussian Mixture Models (GMMs)
- Incorporating structure in latent variables, e.g., Categorical Distributions

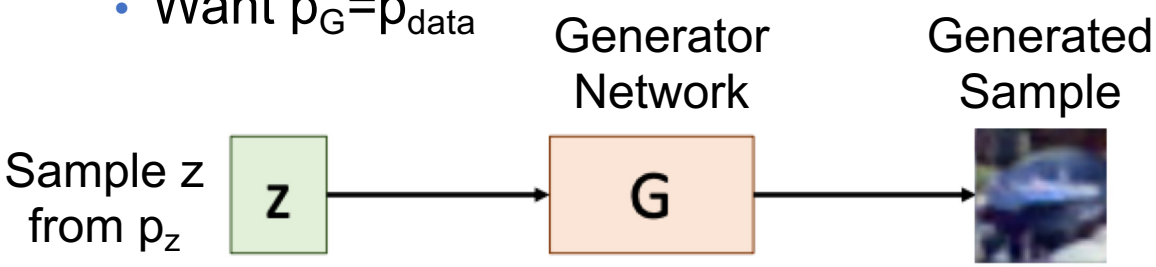
Generative Adversarial Networks

Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want p_G=p_{data}

Generative Adversarial Networks

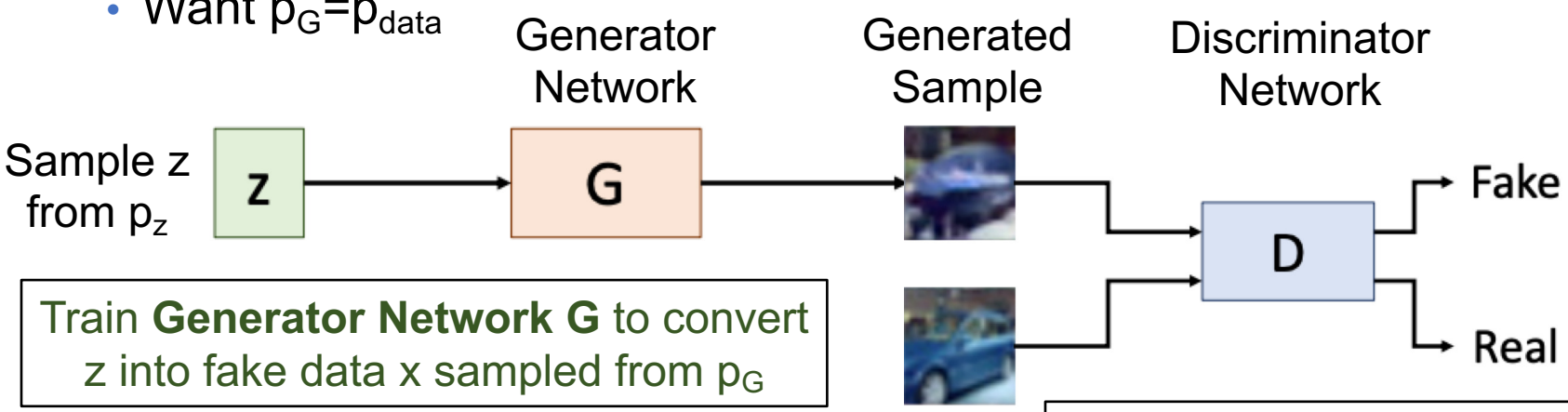
- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want $p_G=p_{data}$



Train **Generator Network G** to convert z into fake data x sampled from p_G

Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want $p_G=p_{data}$

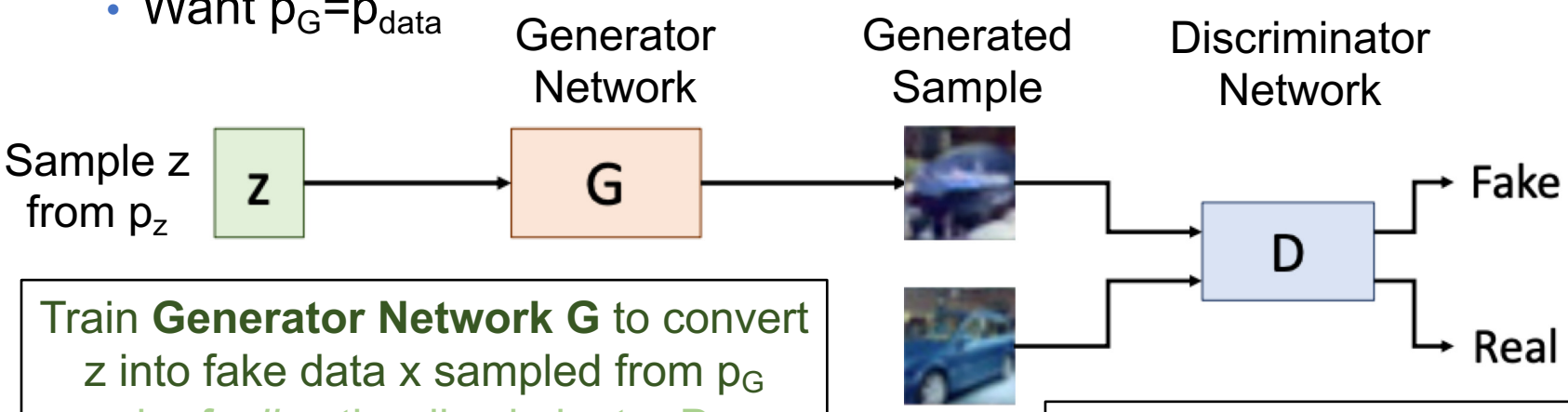


Train **Generator Network G** to convert z into fake data x sampled from p_G

Train **Discriminator Network D** to classify data as real or fake (0/1)

Generative Adversarial Networks

- Assume data x_i drawn from distribution $p_{data}(x)$.
  - Want to sample from p_{data}
- **Idea**: Introduce a latent variable z with simple prior p(z)
  - Sample z ~p(z) and pass to a Generator Network x=G(z)
  - X is a sample from the Generator distribution p_G.
  - Want $p_G=p_{data}$

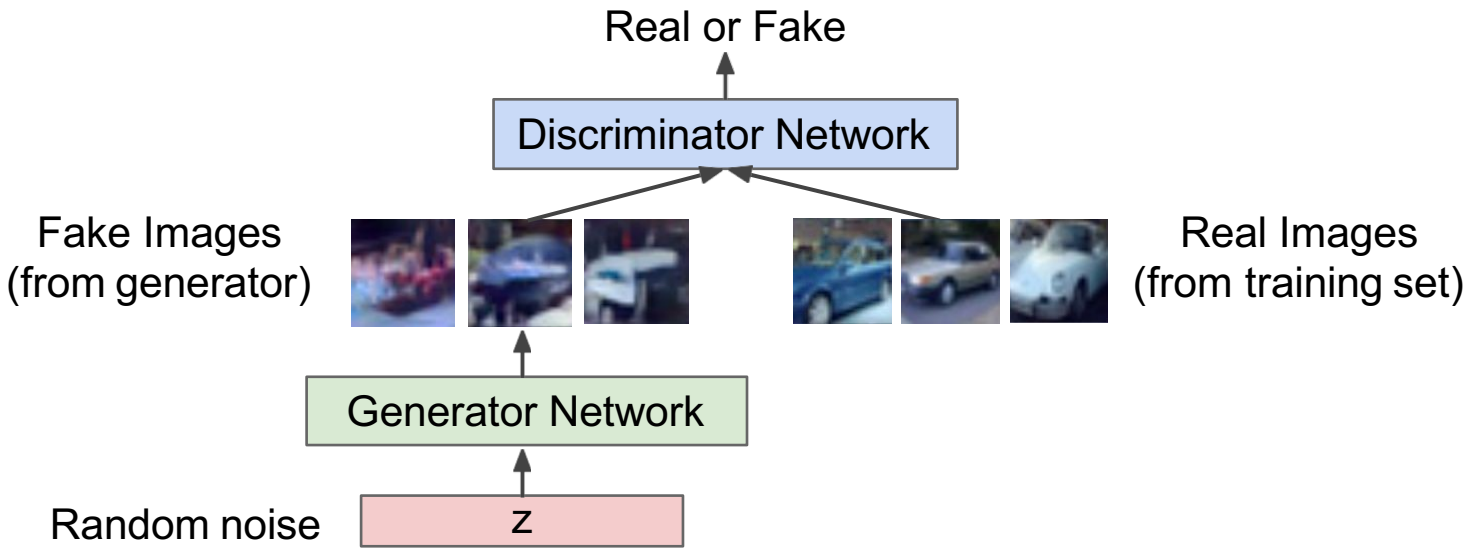


Train **Generator Network G** to convert z into fake data x sampled from p_G by *fooling* the discriminator D

Train **Discriminator Network D** to classify data as real or fake (0/1)

Training GANs: Two-player game

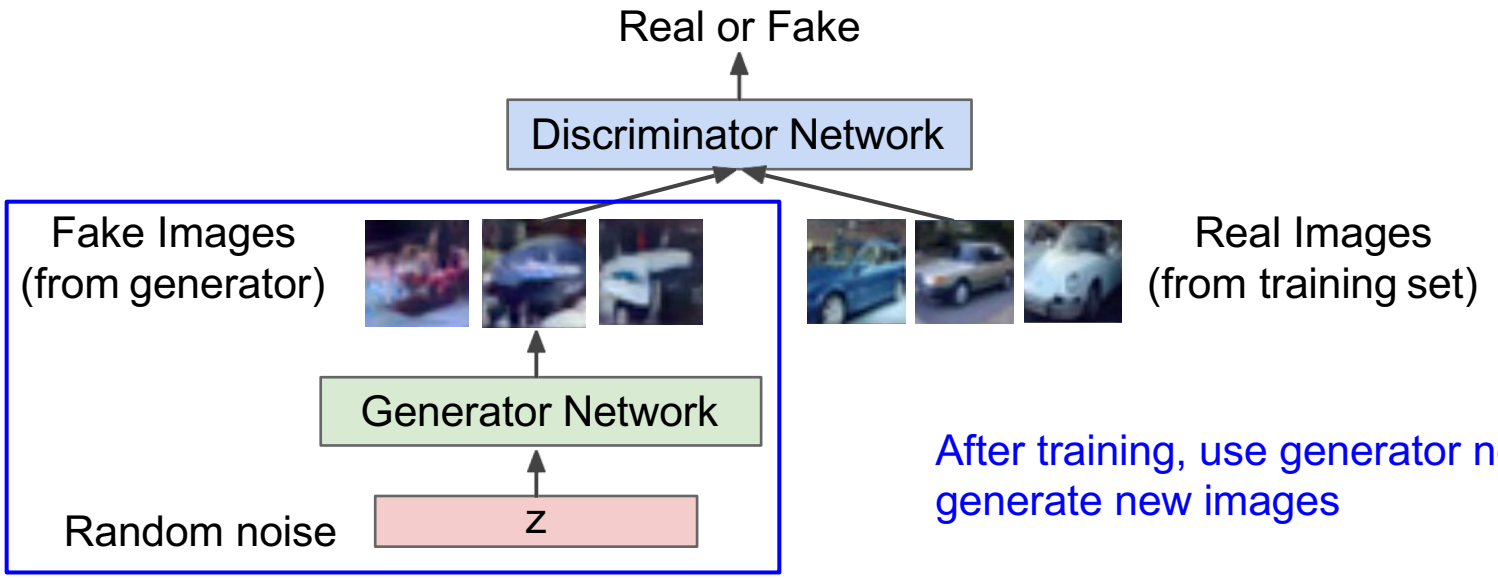
Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Training GANs: Two-player game

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



After training, use generator network to generate new images

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Putting it together: GAN training algorithm

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:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D_{\theta_d}(x^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(z^{(i)}))) \right]$$

  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 ascending its stochastic gradient (improved objective):

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log(D_{\theta_d}(G_{\theta_g}(z^{(i)})))$$

end for

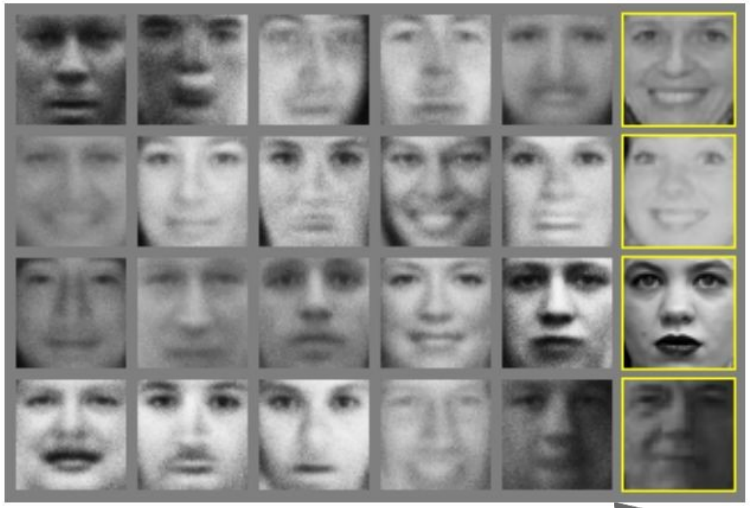
Some find k=1 more stable, others use k > 1, no best rule.

Recent work (e.g. Wasserstein GAN) alleviates this problem, better stability!

Training GANS is hard!

Generator can “memorize” real images

Generated samples

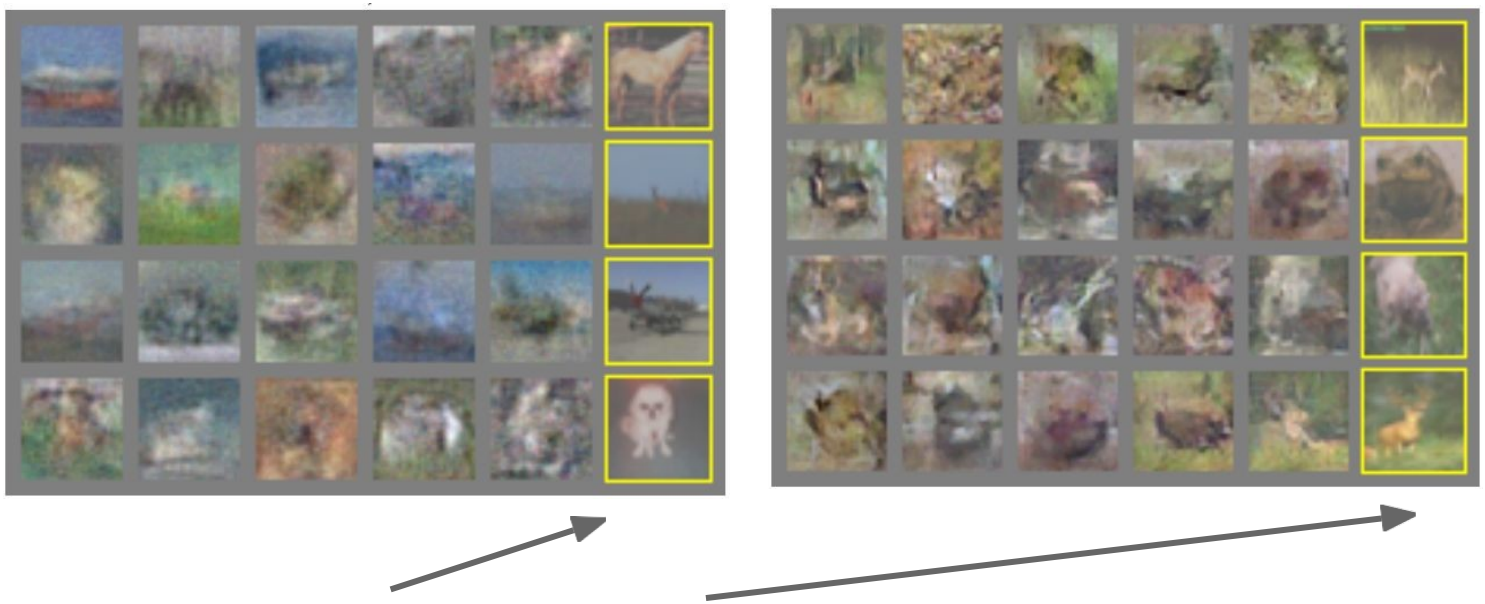


Nearest neighbor from training set

Training GANS is hard!

Generator can produce “fuzzy” images

Generated samples (CIFAR-10)



Nearest neighbor from training set

Training GANS is hard!

Generator is an upsampling network with fractionally-strided convolutions
Discriminator is a convolutional network

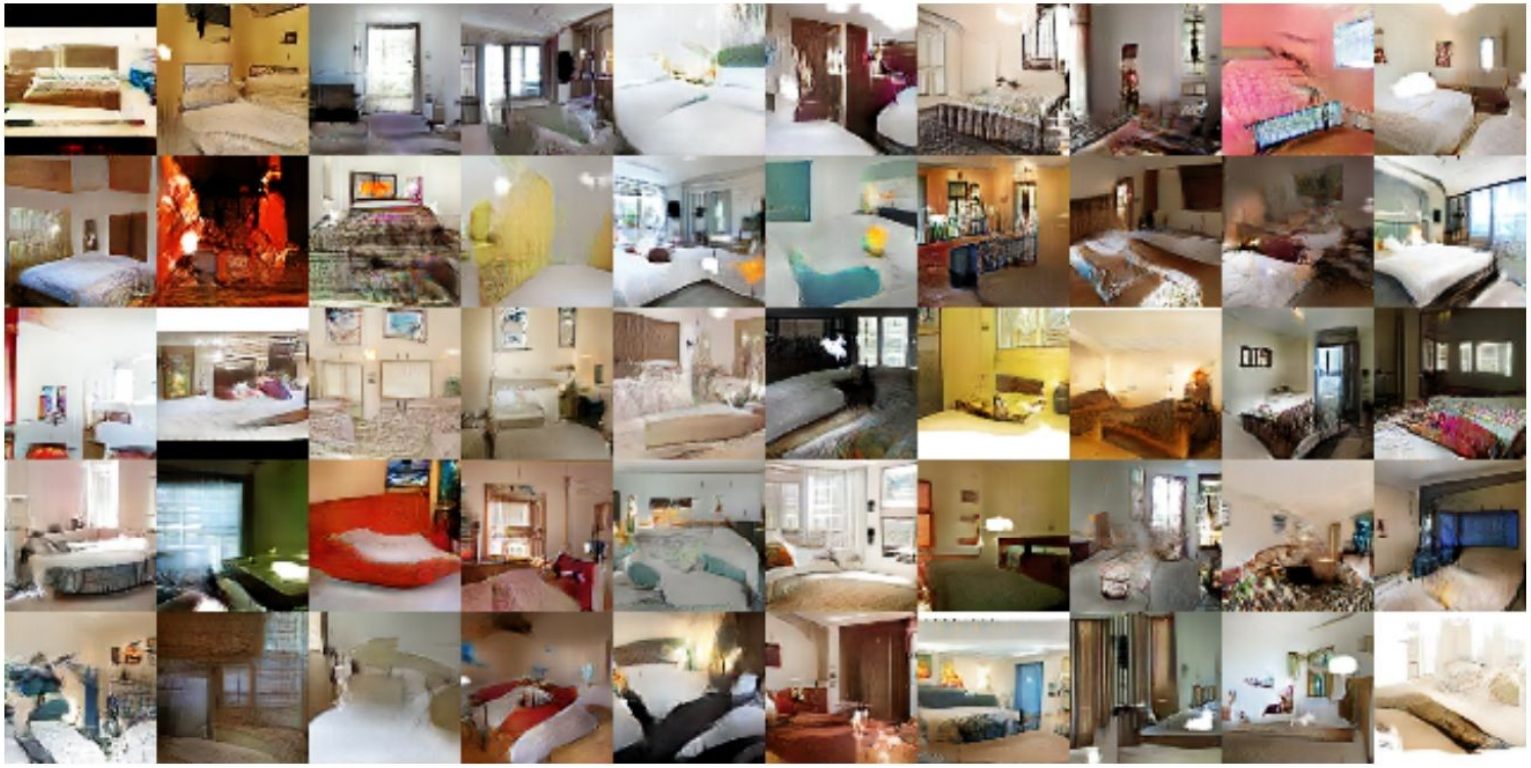
Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Generative Adversarial Nets: Convolutional Architectures

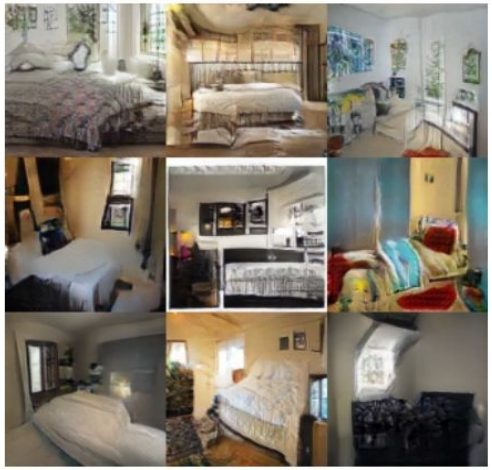
Samples from the model look much better!



Radford et al, “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks”, ICLR 2016

Examples

Better training and generation



LSGAN, Zhu 2017.

Wasserstein GAN, Arjovsky 2017. Improved Wasserstein GAN, Gulrajani 2017.

Progressive GAN, Karras 2018.

Examples

https://thispersondoesnotexist.com/

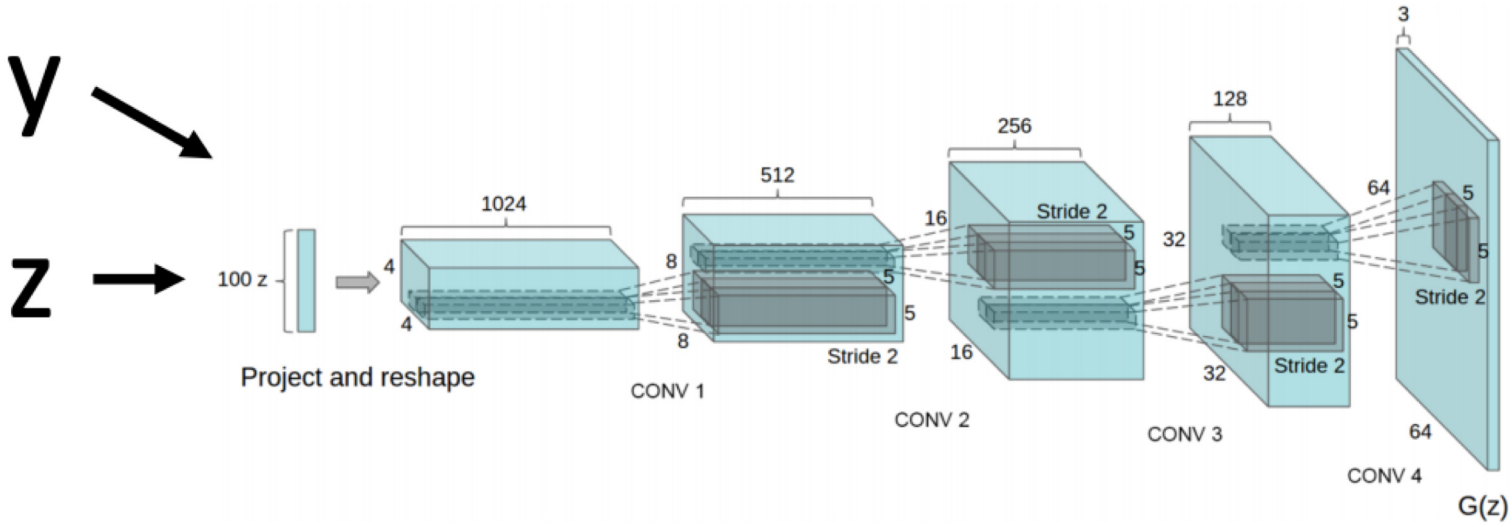
https://www.whichfaceisreal.com/

2019: BigGAN

Brock et al., 2019

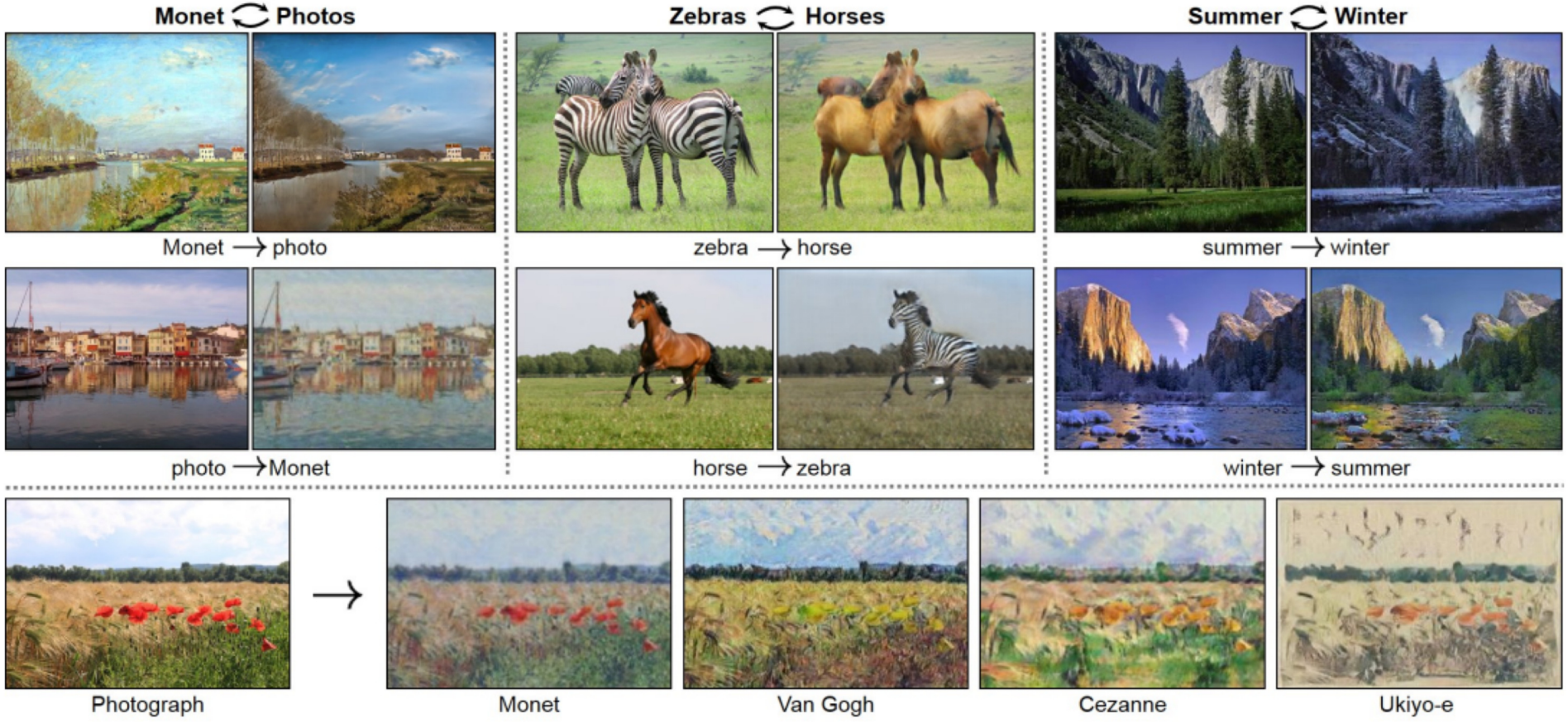
Conditional GANs

- Conditional GAN: learn p(y|x) instead of p(x)
  - Make generator and discriminator both take label y as an additional input



Conditional GANs

Conditional GANs



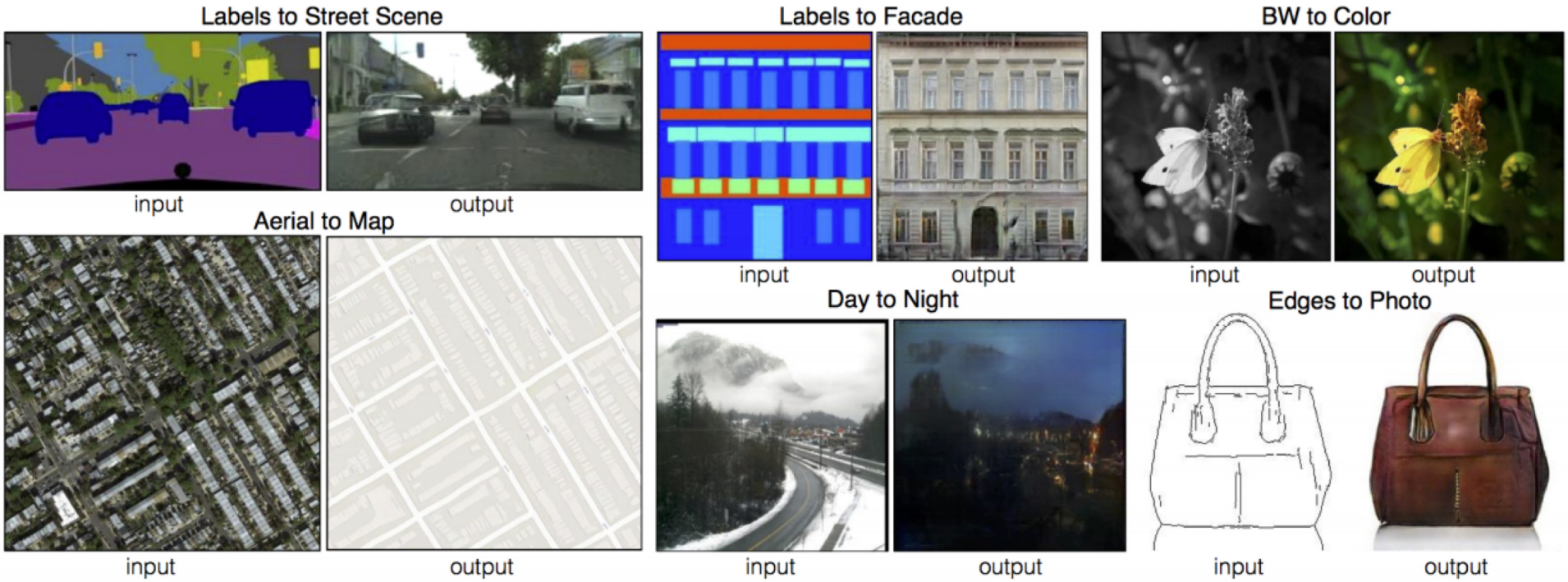
CycleGAN

Conditional GANs

CycleGAN

Jun-Yan Zhu, Taesung Park et. al. ICCV 2017
https://www.youtube.com/watch?v=9reHvktowLY

Conditional GANs

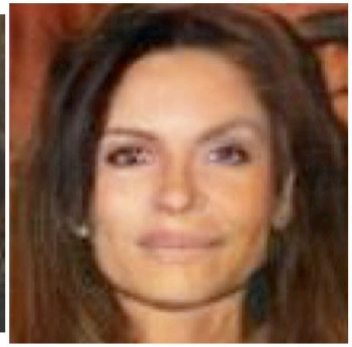


Pix2Pix

2017-2019: Explosion of GANs

2014

2015



2016

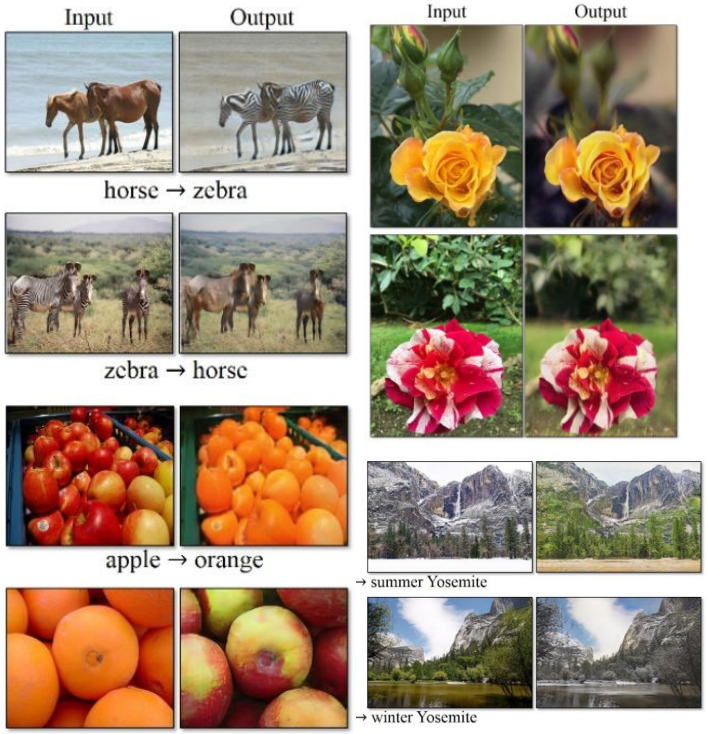
2017

2018

https://arxiv.org/abs/1406.2661
https://arxiv.org/abs/1511.06434
https://arxiv.org/abs/1606.07536
https://arxiv.org/abs/1710.10196
https://arxiv.org/abs/1812.04948

2017-2019: Explosion of GANs

Source->Target domain transfer



CycleGAN. Zhu et al. 2017.

Text -> Image Synthesis

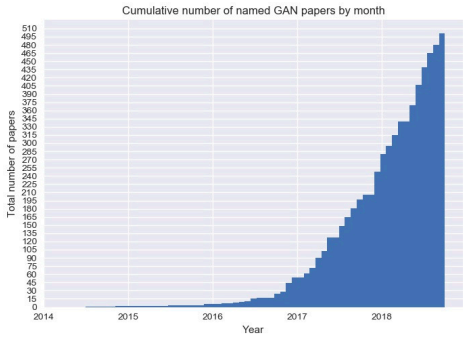
Reed et al. 2017.

Many GAN applications

Pix2pix. Isola 2017. Many examples at https://phillipi.github.io/pix2pix/

2017-2019: Explosion of GANs

"The GAN Zoo"



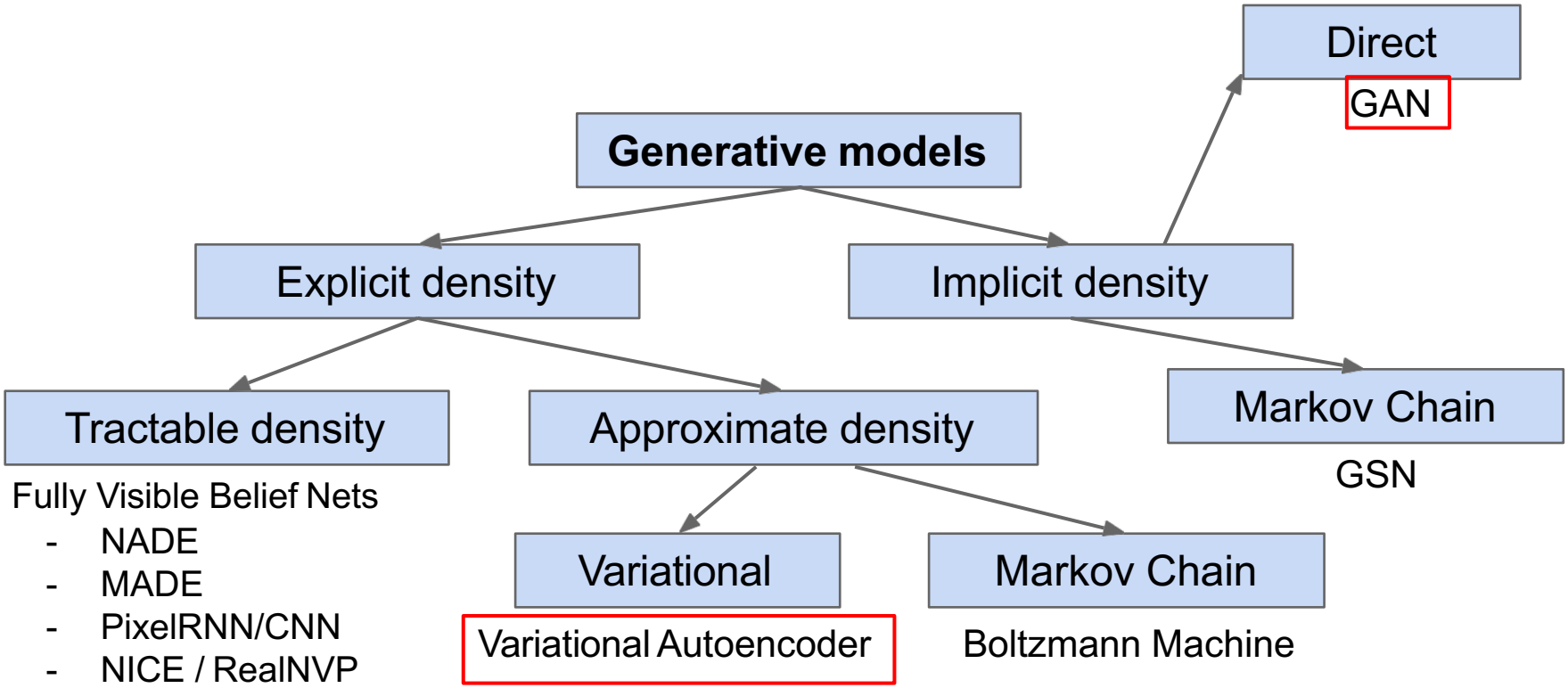
- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks

See also:
https://github.com/soumith/ganhacks for tips and tricks for trainings GANs

- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo

Taxonomy of Generative Models



Useful Resources on Generative Models

Ian Goodfellow: Generative Adversarial Networks (NIPS 2016 tutorial)

Ian Goodfellow: Adversarial Machine Learning, AAAI-19 Invited Talk

Are GANs Created Equal? A Large-Scale Study
https://arxiv.org/abs/1711.10337

CS 236: Deep Generative Models (Stanford)

CS 294-158 Deep Unsupervised Learning (Berkeley)

Conclusions

Towards Unsupervised Learning:

- Supervised vs. Unsupervised Learning

Synthesis-based techniques:

- Auto-encoders. Useful basic building block. Can lead to informative image features without supervision. Rarely used on its own.
- Variational Auto-Encoders. Models with explicit probability distributions. Well-defined, but can leads to approximate results.
- Generative Adversarial Networks (GANs). Powerful generative models, typically without explicit density modeling. Can lead to high quality images, but can be difficult to train.
- **Many** extensions with more and more realistic results in recent years.