Deep Learning

Advances in Generative Models

Dr Chris Willcocks

Department of Computer Science

Durham University

Lecture Overview

Durham University

Aim

- Learn state-of-the-art practical techniques

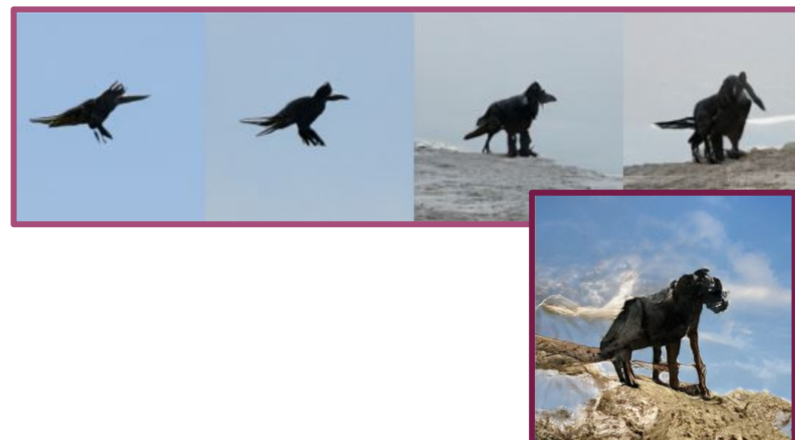
Today's lecture

- Generative adversarial networks
- Wasserstein GANs, Conditional GANs
- Autoencoders and VAEs
- Improved latent space interpolation
- Skip connections (U-Net)

GAN WGAN VAE U-Net

https://github.com/cwkx/ml-materials

Think about how you will make your ***Pegasus*** *during this lecture.*

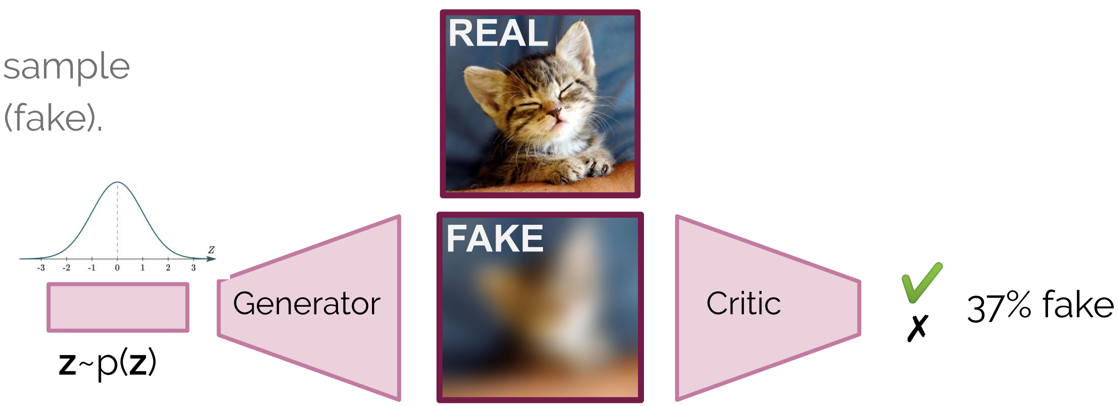


GANs
Generative Adversarial Networks

> Generative Adversarial Nets - NIPS Proceedings
> https://papers.nips.cc/paper/5423-generative-adversarial-nets
> by I Goodfellow - 2014 - Cited by 7299 - Related articles
> We propose a new framework for estimating **generative** models via **adversarial** **nets**, in which we simultaneously train two models: a **generative** model G that ...

GAN

- Two-player zero-sum non-cooperative game (minmax) with the value function *V(G,D)* of two networks:
  - **Discriminator** (Critic)
    - Estimates probability of sample being real or generated (fake).
  - **Generator**
    - Learns to map noise from prior to samples from p_{data}



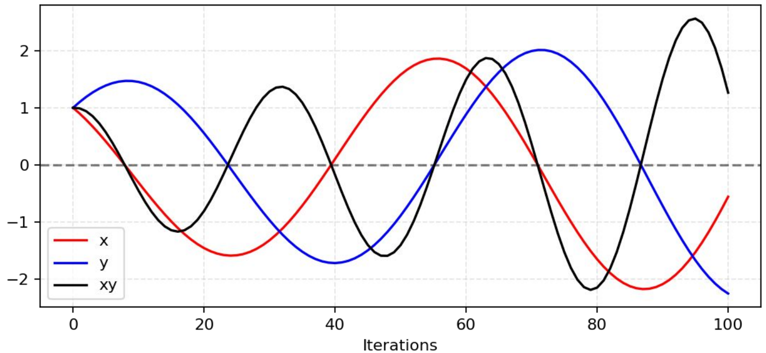
$$\min_G \max_D L(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$
$$= \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{x\sim p_g(z)}[\log(1 - D(x))].$$

Generative Adversarial Networks

“Stability does not come solely from G or D, but from their interaction through the adversarial process”

Notoriously difficult to train:

- Non-convergence
  - Hard to achieve Nash equilibrium
- Diminishing gradient
- Difficult to tune
- Mode collapse (next slide)

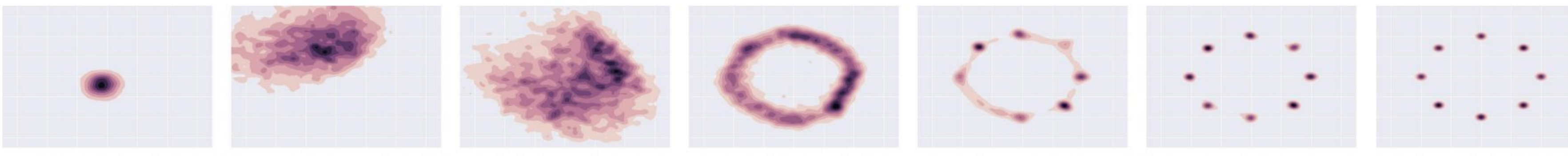


A simulation of our example for updating x to minimize xy and updating y to minimize $-xy$. The learning rate η=0.1. With more iterations, the oscillation grows more and more unstable.

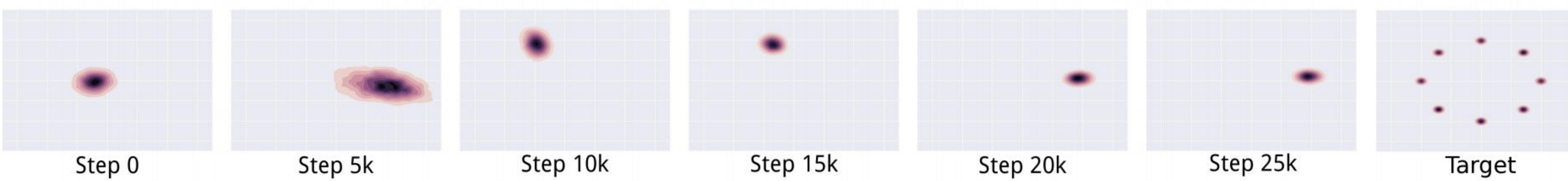
Further reading: https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

Mode Collapse

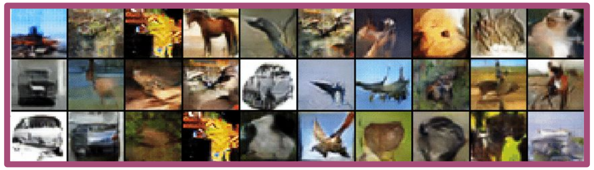
Preferred behaviour:



Common mode collapse:

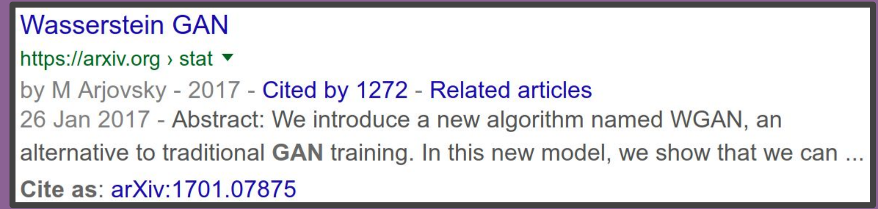


above from "Unrolled Generative Adversarial Networks"



left shows collapsed modes on CIFAR-10

Wasserstein GANs


Wasserstein GAN
https://arxiv.org › stat ▾
by M Arjovsky - 2017 - Cited by 1272 - Related articles
26 Jan 2017 - Abstract: We introduce a new algorithm named WGAN, an alternative to traditional **GAN** training. In this new model, we show that we can ...
Cite as: arXiv:1701.07875

A real-valued function $f : \mathbb{R} \to \mathbb{R}$ is called $\boldsymbol{K}$-Lipschitz continuous if there exists a real constant $K \geq 0$ such that, for all $x_1, x_2 \in \mathbb{R}$, $|f(x_1) - f(x_2)| \leq K|x_1 - x_2|$

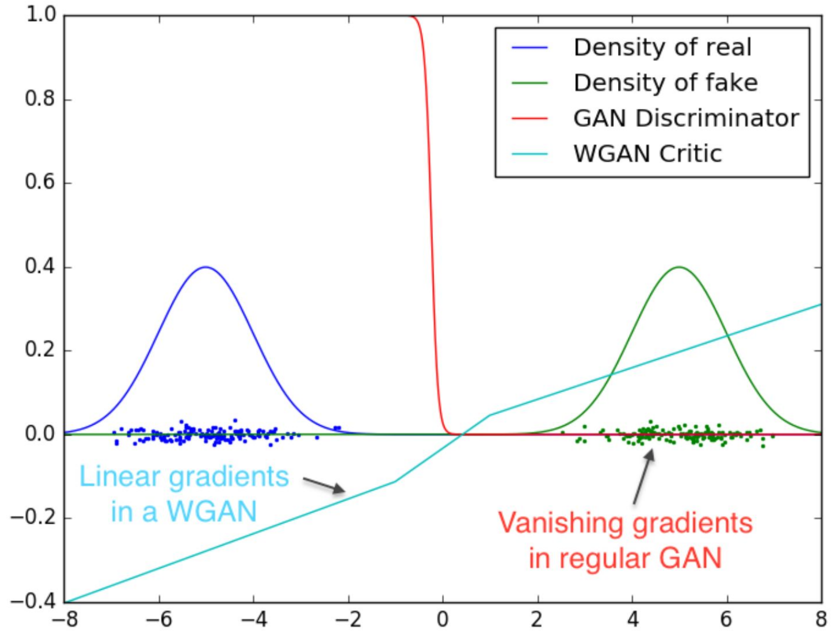
WGAN

If K-Lipschitz

The original paper proposes clamping gradients to a Hypercube.

- Tends to slide in corners.

https://github.com/cwkx/ml-materials/blob/master/lectures/notes/7-wasserstein.py



WGAN Gradient Penalty

Improved Training of Wasserstein GANs
https://arxiv.org › cs
by I Gulrajani - 2017 - Cited by 998 - Related articles
31 Mar 2017 - **Improved Training of Wasserstein GANs**. Generative Adversarial Networks (**GANs**) are powerful generative models, but suffer from **training** instability. The recently proposed **Wasserstein GAN** (WGAN) makes progress toward stable **training** of **GANs**, but sometimes can still generate only low-quality samples or fail to converge.
Cite as: arXiv:1704.00028

Optimal critic contains straight lines with gradient norm 1 connecting coupled points from $\mathbf{P}_r$ and $\mathbf{P}_g$

WGAN

Add another term 1-Lipschitz constraint:

$$L = \underbrace{\mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g} [D(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{x} \sim \mathbb{P}_r} [D(\boldsymbol{x})]}_{\text{Original critic loss}} + \underbrace{\lambda \mathop{\mathbb{E}}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}} \left[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2 \right]}_{\text{Our gradient penalty}} .$$

Backpropagating new loss requires computing gradient of gradients, which is quite slow.

```
torch.autograd.grad(outputs, inputs, grad_outputs=None, retain_graph=None, create_graph=False,
only_inputs=True, allow_unused=False)
```

Spectral Normalisation

arxiv.org › cs

Spectral Normalization for Generative Adversarial Networks

by T Miyato - 2018 - Cited by 806 - Related articles

16 Feb 2018 - In this paper, we propose a novel weight normalization technique called **spectral normalization** to stabilize the training of the discriminator.

State-of-the-art solution to mode collapse:

1. Directly sets discriminator weights to 1-Lipschitz
2. Much faster/more efficient
3. Easier to implement and train
    a. Wrap all layers in spectral norm layers
    b. (remove all batch norm)
    c. Overtrain discriminator

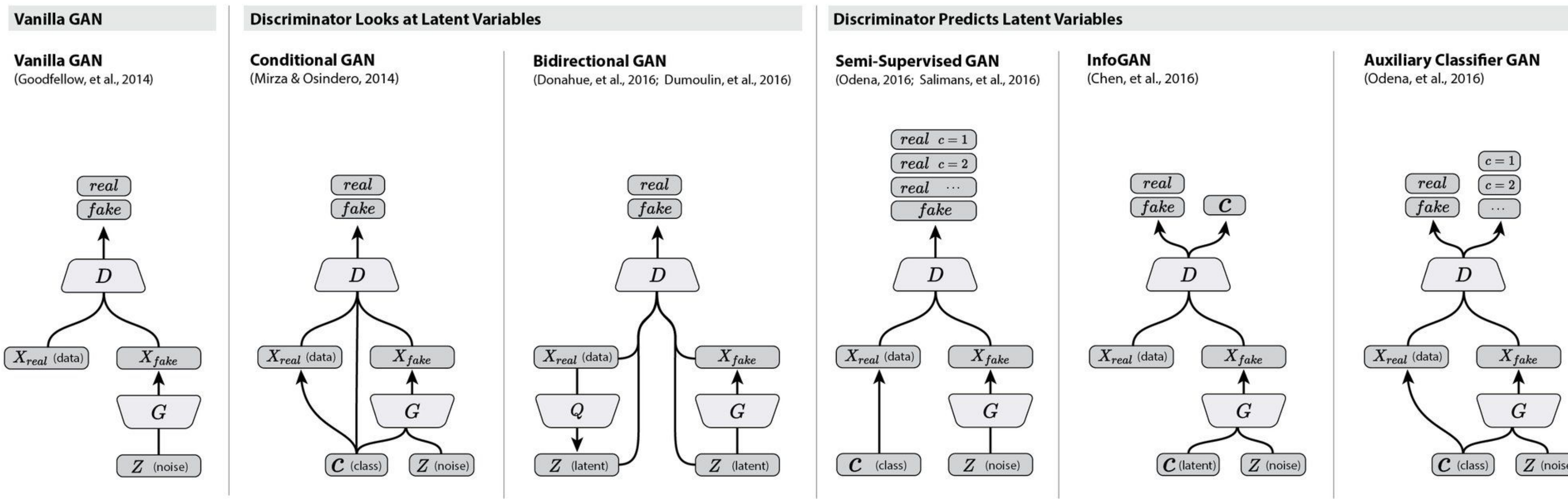
SN-GAN

$$\bar{W}_{\mathrm{SN}}(W) := W/\sigma(W)$$

"Scaling GANs to a high amount of classes has been a major open challenge and this paper has achieved an amazing 10X leap forward."
- Ian Goodfellow, OpenReview

```
class Discriminator(nn.Module):
    def __init__(self, f=64):
        super(Discriminator, self).__init__()
        self.discriminate = nn.Sequential(
            torch.nn.utils.spectral_norm(nn.Conv2d(1, f, 3, 1, 1)),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(kernel_size=(2,2)),
            torch.nn.utils.spectral_norm(nn.Conv2d(f, f*2, 3, 1, 1)),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(kernel_size=(2,2)),
            torch.nn.utils.spectral_norm(nn.Conv2d(f*2, f*4, 3, 1, 1)),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(kernel_size=(2,2)),
            torch.nn.utils.spectral_norm(nn.Conv2d(f*4, f*8, 3, 1, 1)),
            nn.LeakyReLU(0.1, inplace=True),
            nn.MaxPool2d(kernel_size=(2,2)),
            torch.nn.utils.spectral_norm(nn.Conv2d(f*8, 1, 3, 1, 1)),
            nn.Sigmoid()
        )
```

GAN Conditioning & Disentanglement



(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)

Autoencoder

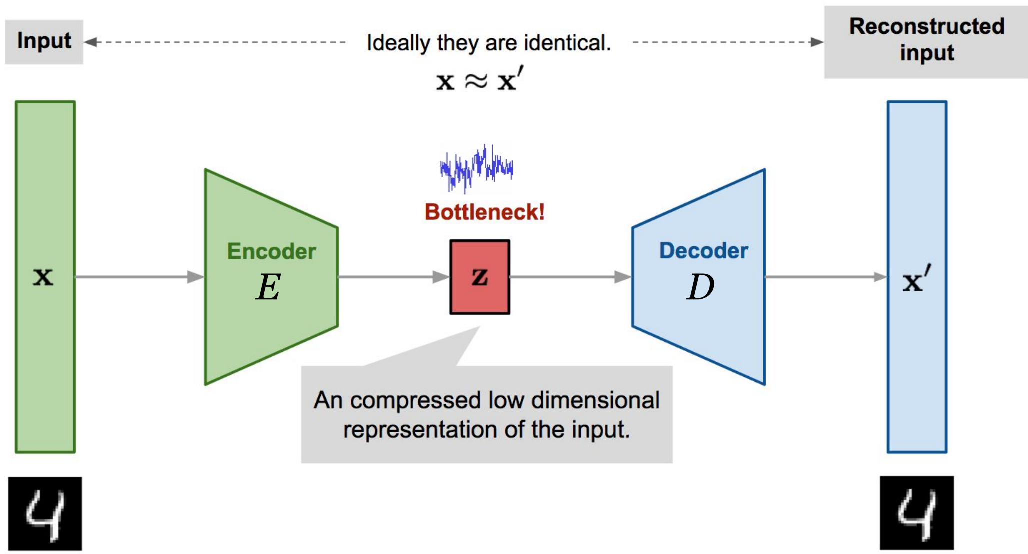
Durham University

Learns an identify function unsupervised: $\mathcal{L}_{\text{auto}} = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[\|\mathbf{x} - D_\theta(E_\theta(\mathbf{x}))\|^2 \right]$

- Dimensionality reduction
- **Compresses** input
- Easy to train

Problems

- z~q(z) is difficult to draw samples from
- non-semantic reconstructions



https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

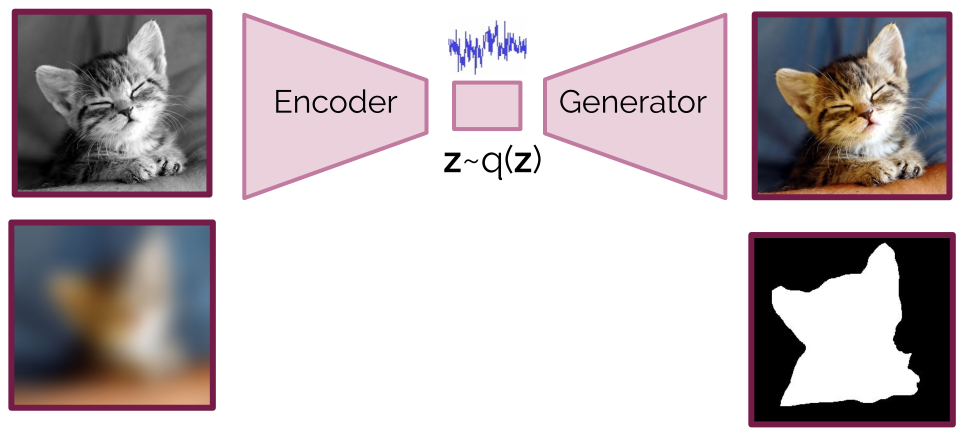
Can be Conditional & Constrained

Can be used for translation tasks $p(\mathbf{y}|\mathbf{x})$ (e.g. image-to-image)

Naive usage for:

- Deblurring
- Black and white to color
- Segmentation

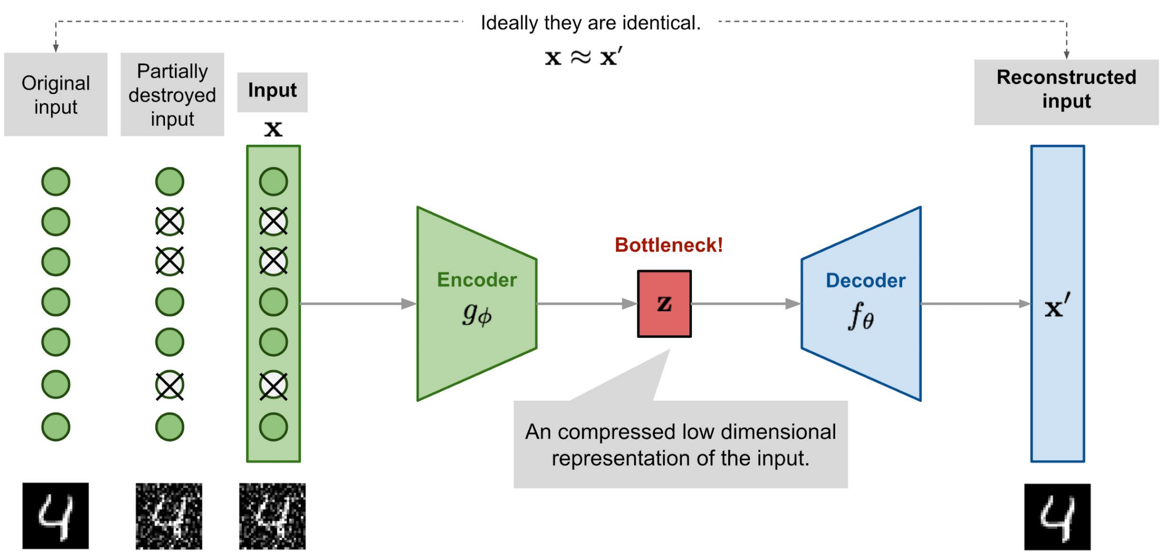
$$\mathcal{L}_{\text{translate}} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}\left[\|\mathbf{y} - D(E(\mathbf{x}))\|^2 \right]$$



Denoising Autoencoder

Repairs a partially corrupted input (just as humans are able to distinguish objects).

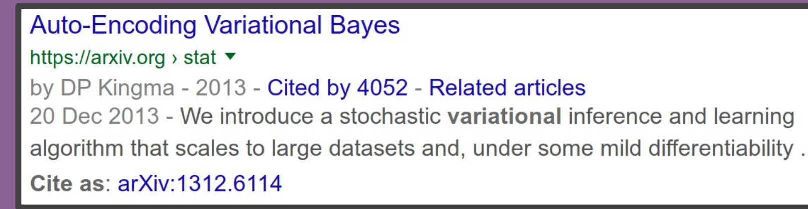
$$\mathcal{L}_{\text{denoise}} = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}},\hat{\mathbf{x}}\sim\mathcal{M}(\hat{\mathbf{x}}|\mathbf{x})}\left[\|\mathbf{x} - D(E(\hat{\mathbf{x}}))\|^2\right]$$



Extended approaches:

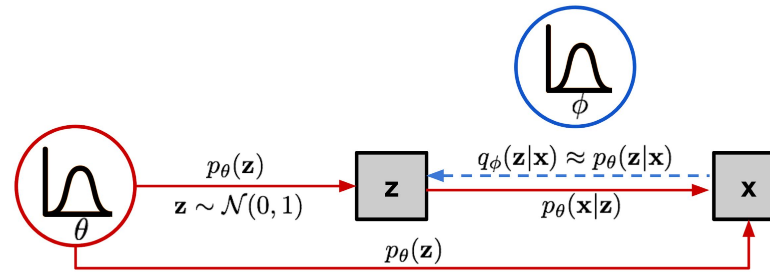
- Inpainting
- Augmentation
- Layerwise corruption

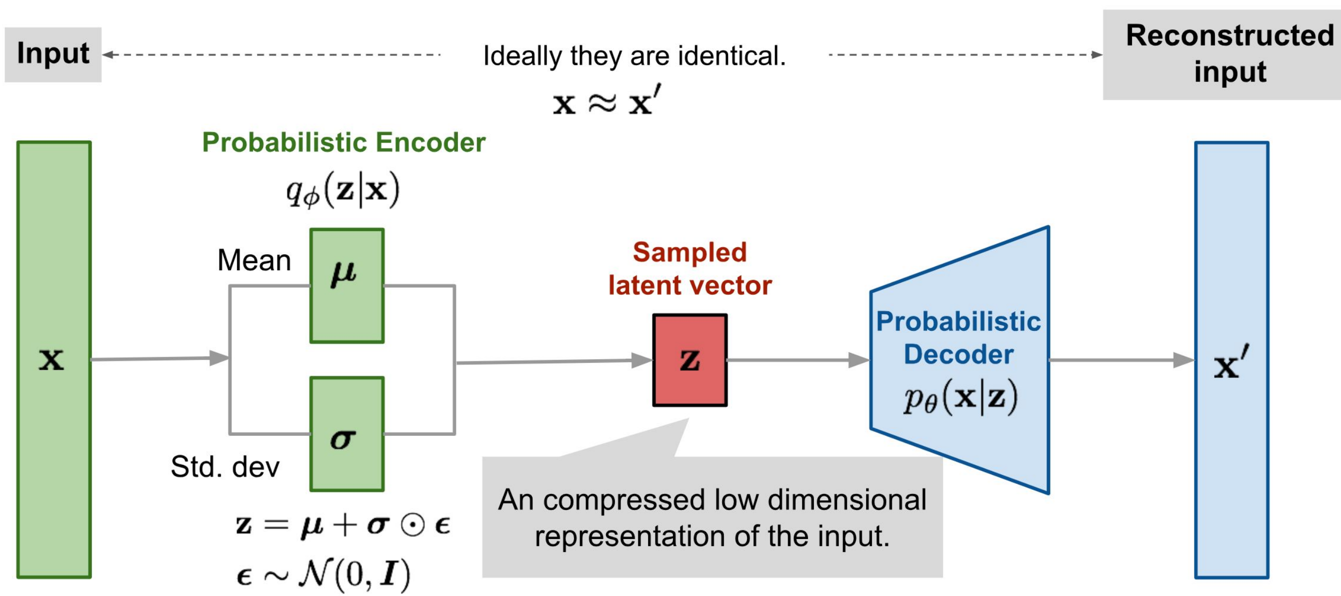
Variational Autoencoder


Auto-Encoding Variational Bayes
https://arxiv.org › stat
by DP Kingma - 2013 - Cited by 4052 - Related articles
20 Dec 2013 - We introduce a stochastic **variational** inference and learning algorithm that scales to large datasets and, under some mild differentiability ...
Cite as: arXiv:1312.6114

Instead of mapping input into a *fixed* vector, what if we want to map into a distribution?

Prior $p_\theta(\mathbf{z})$, **Likelihood** $p_\theta(\mathbf{x}|\mathbf{z})$, **Posterior** $q_\theta(\mathbf{z}|\mathbf{x})$

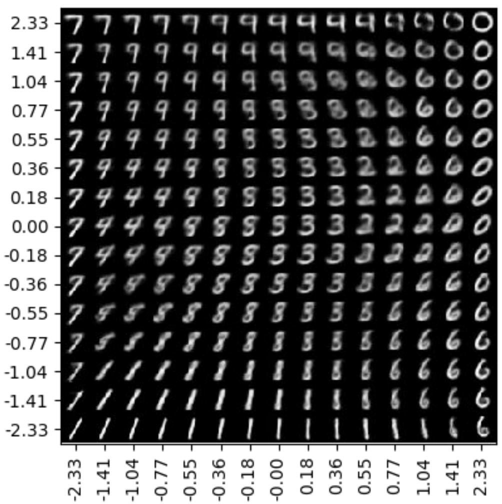


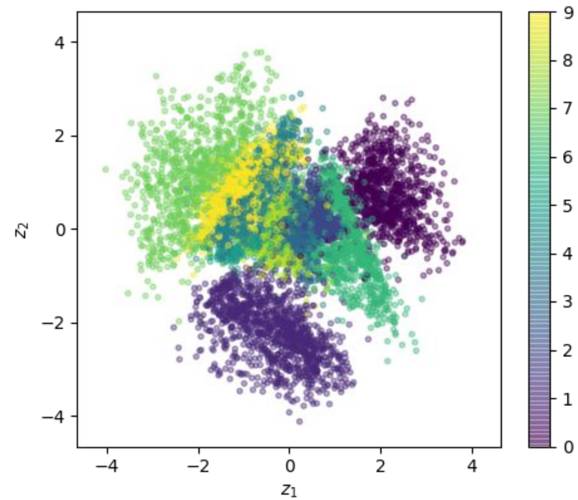


VAE

Manifold learnt by a VAE

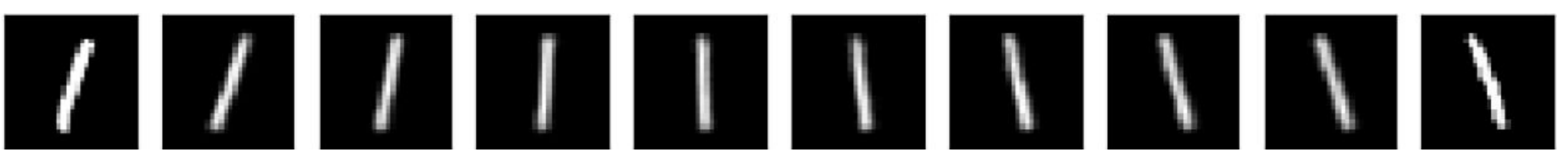
If we embed to 2D encoding, we can plot points for all inputs:





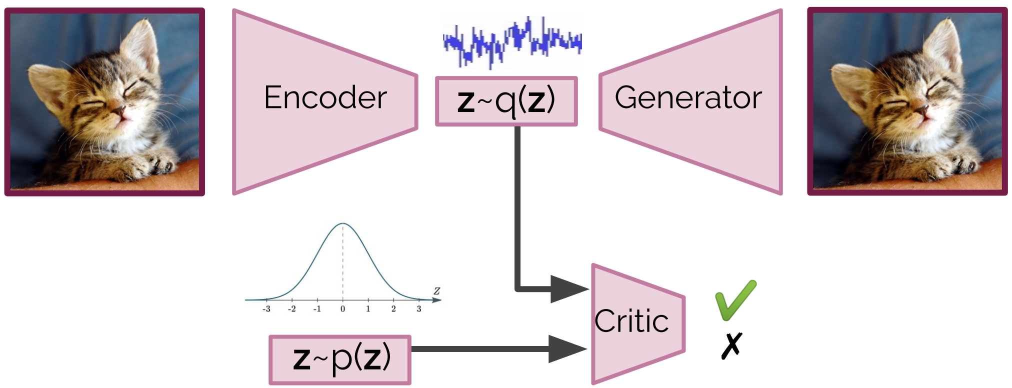
How can we improve this **manifold**?

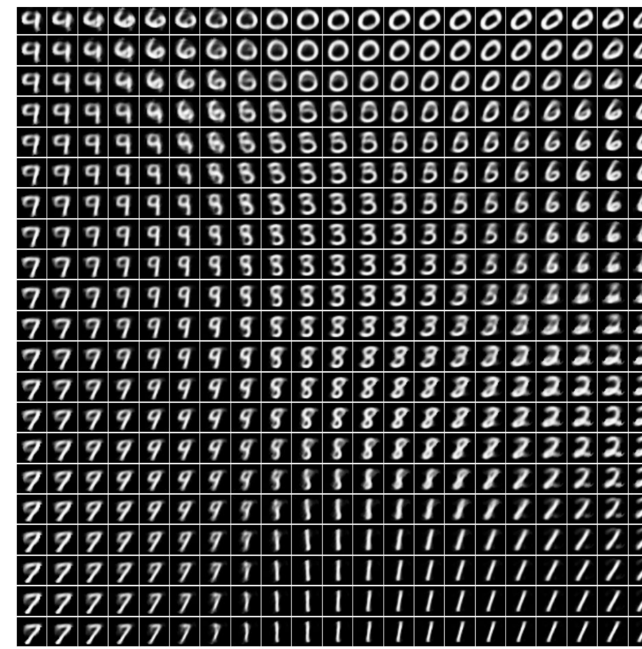
- Labels = easy!
- Generate more samples
  - Adversarial strategies



Adversarial Autoencoders

Uses an adversarial strategy to make aggregate posterior distribution look like a prior distribution:





...there's also a conditional version.

Adversarially Improved Interpolation

Durham University

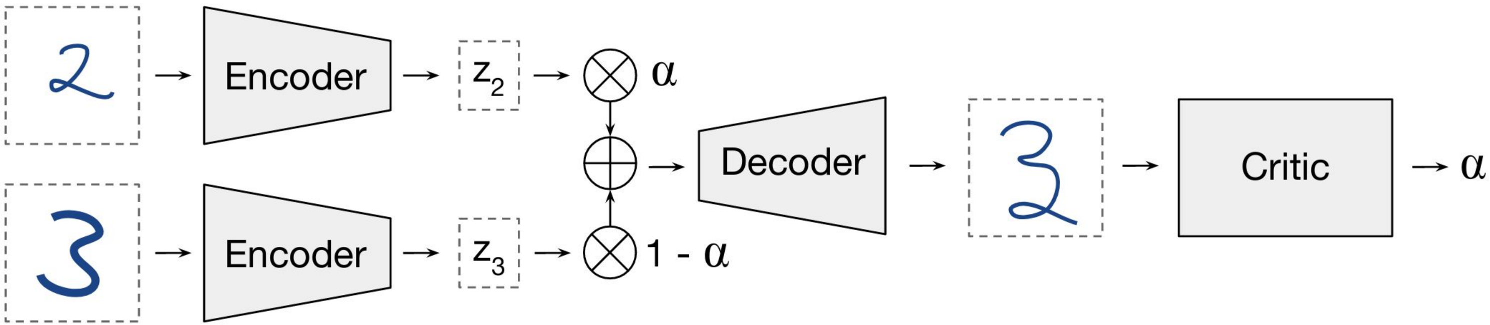
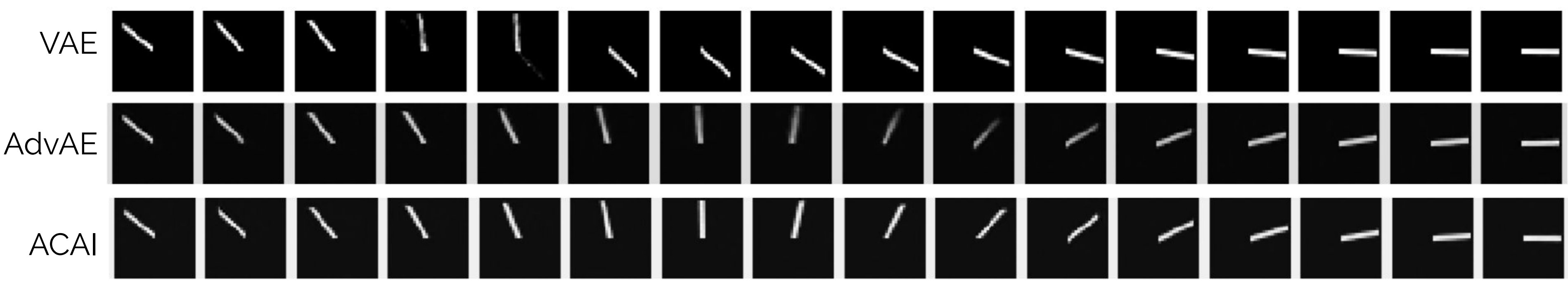
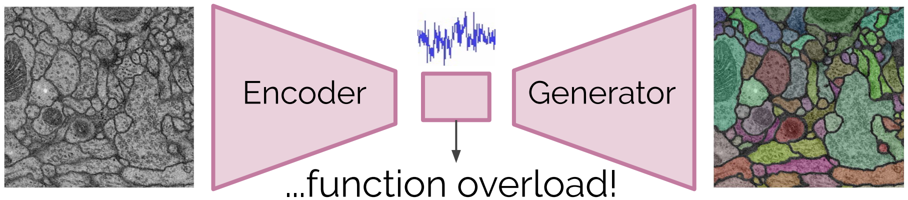


Figure 1: Adversarially Constrained Autoencoder Interpolation (ACAI). A critic network tries to predict the interpolation coefficient α corresponding to an interpolated datapoint. The autoencoder is trained to fool the critic into outputting $\alpha = 0$.



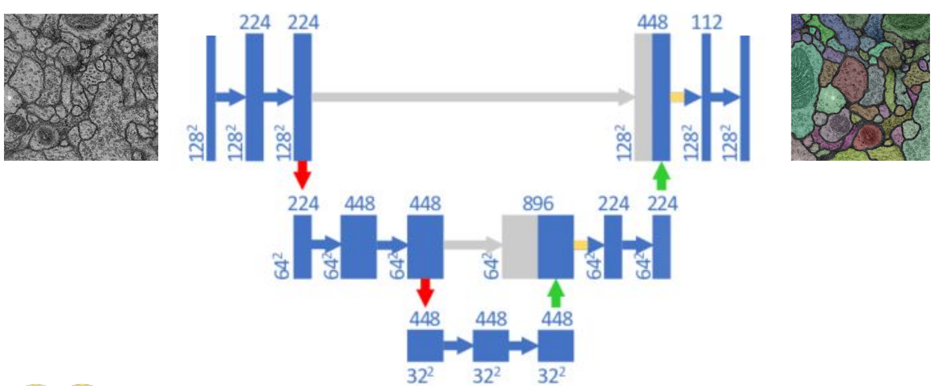
Skip connections (U-Net)

Durham University



Autoencoders are not well-suited for spatially-aligned image-to-image tasks (blurry outputs).

...so we add skip-connections concatenating encoder outputs to the decoder:



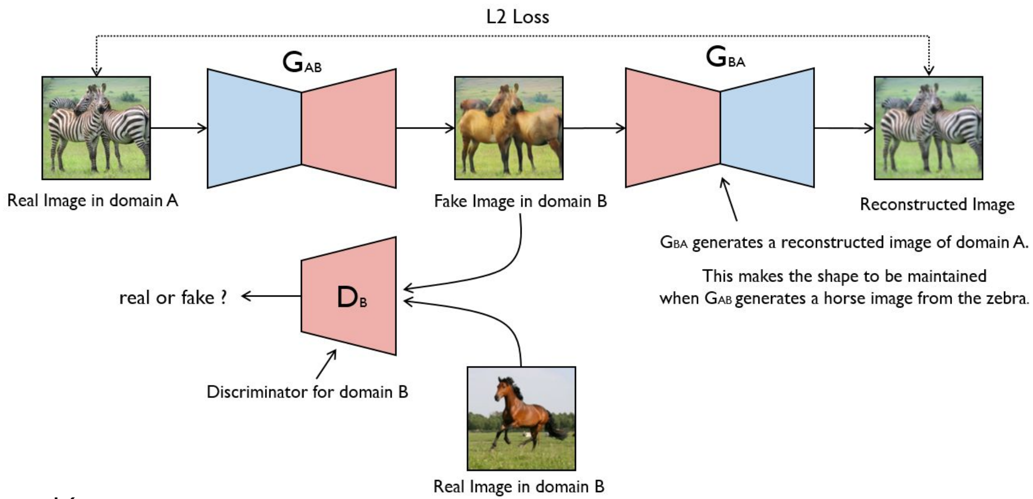
- Can't be used for reconstructive approaches, why?
- Great for segmentation

[Demo link]

Segnet

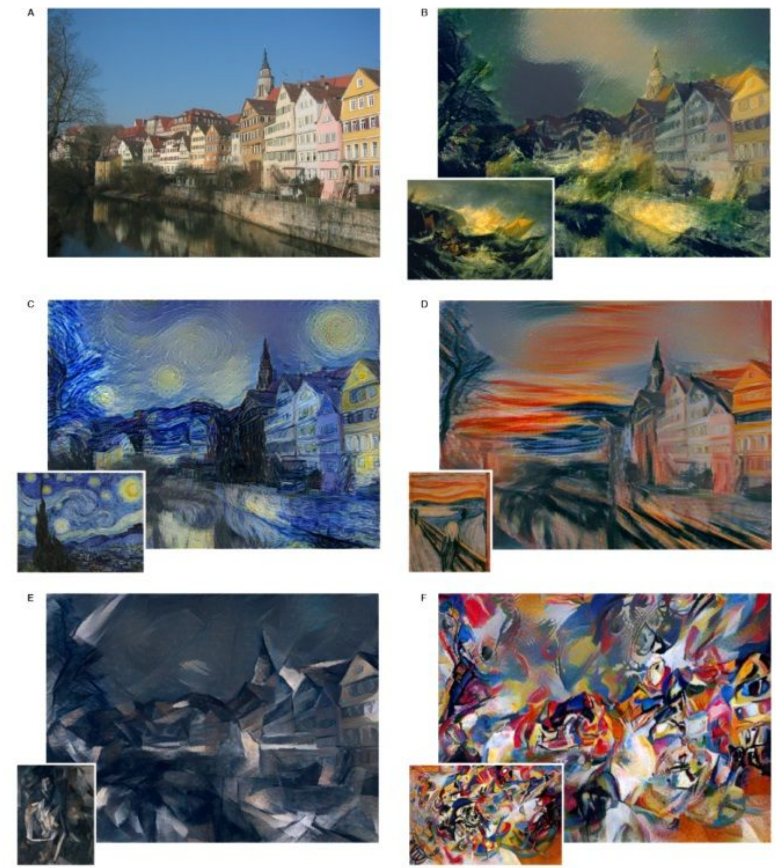
Style Transfer & Unpaired Translation

Durham University



Key papers:

- CycleGAN
- “A Neural Algorithm for Artistic Style”
- “Perceptual Losses for Real-Time Style Transfer and Super-Resolution”
- “Image-to-Image Translation with Conditional Adversarial Networks”



Take away points

- If we can generate it properly, we can understand the structure of it
- GANs have been a huge breakthrough
  - GAN stability has improved a lot
- Data distribution vs network design
- Semantic reasoning vs realism