Zurich University
of Applied Sciences

Generative Adversarial Nets as generative models

Datalab Christmas lecture, December 21, 2016

Thilo Stadelmann

Generative modeling
Generative Adversarial Nets
Use case: image inpainting

With material from
- Arthur Juliani's and Brandon Amos's blog posts
- Ian Goodfellow, UC Berkeley COMPSCI 294 guest lecture

1. GENERATIVE MODELING

Probability distributions and density functions

Terminology: its probability density function (pdf) is one way to describe a distribution.

What does a pdf tell about a set of data?

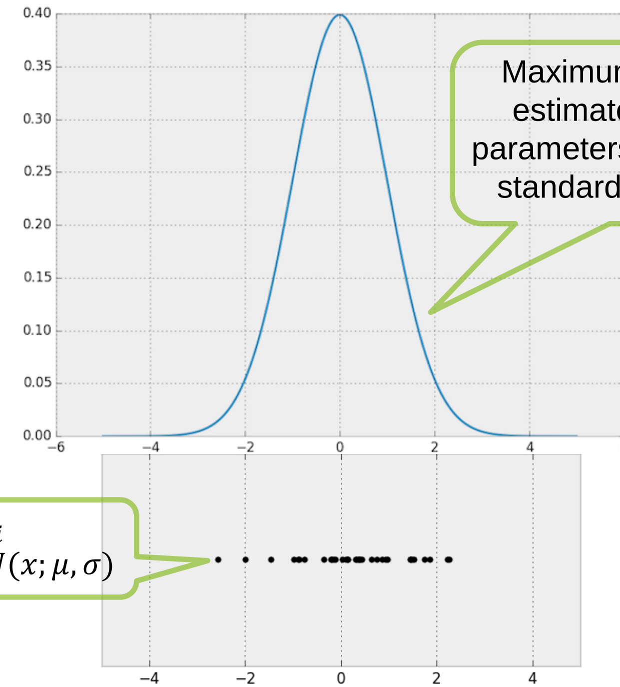
- Where to expect samples
- …with which probability
- Correlation/covariance of dimensions

➔ For data coming from some stochastic processes, the pdf tells **everything there is to know** about the data

➔ **Allow for sampling** data from the underlying distribution

An example generative model

- Recovering a known, parametric pdf: The univariate Gaussian



Maximum likelihood estimate $p(x)$ with parameters mean μ and standard deviation σ

Given data points x_i
Assumption: $x_i \sim p(x;\theta) = N(x;\mu,\sigma)$

Source: Brandon Amos, «*Image Completion with Deep Learning in TensorFlow*», 2016, https://bamos.github.io/2016/08/09/deep-completion/

Pros and cons

Flavors of generative models

- **Statistical** models that directly model the pdf (e.g., GMM, hidden Markov model HMM)
- **Graphical** models with latent variables (e.g., Boltzmann machines RBM/DBM, deep belief networks DBN)
- **Autoencoders**

Promises

- Help **learning about** high-dimensional, complicated probability **distributions** *(even if pdf isn´t represented explicitly)*
- **Simulate** possible futures for planning or simulated RL
- Handle **missing data** (in particular, semi-supervised learning)
- Some applications actually require **generation** (e.g. sound synthesis, identikit pictures, content reconstruction)

Common drawbacks

- Statistical models suffer severely from the **curse of dimensionality**
- Approximations for **intractable probabilistic computations** during ML estimation
- **Unbacked assumptions** (e.g., Gaussianity) and averaging e.g. in VAEs

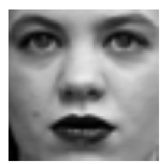
2. GENERATIVE ADVERSARIAL NETS

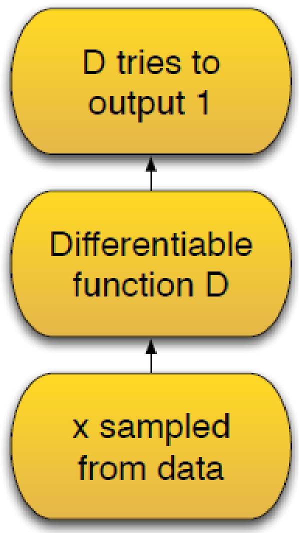
Adversarial nets

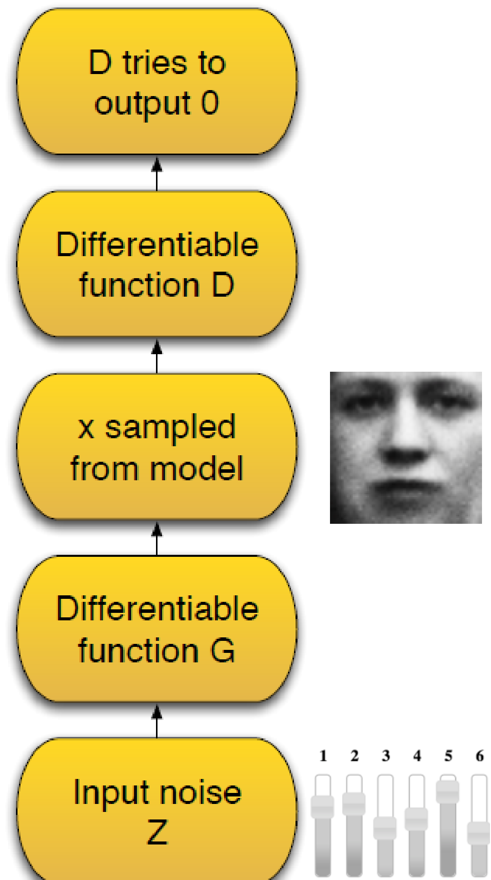
Bootstrapping implicit generative representations

Train 2 models simultaneously [1]

- **G: Generator**
  → learns to generate data
- **D: Discriminator**
  → learns $p(x\ not\ being\ generated)$







→ Both models learn while competing
→ The latent space Z serves as a source of variation to generate different data points
→ Only D has access to real data

[1] Schmidhuber, «Learning Factorial Codes by Predictability Minimization», 1992

Sources: Goodfellow, «Generative Adversarial Networks (GANs)», guest lecture at UC Berkeley COMPSCI 294, 2016-10-03, slide 15; http://www.dpkingma.com/sgvb_mnist_demo/demo.html

Zurich University
of Applied Sciences

No weenies allowed! How SpongeBob helps..

…to understand bootstrapping untrained (G)enerator & (D)iscriminator

Bouncer (D) decides on entry: for tough guys only

SpongeBob (G) wants to appear tough to be admitted

Untrained D focuses on obvious things to discriminate: e.g., physical strength

So G tries to imitate that, but fails

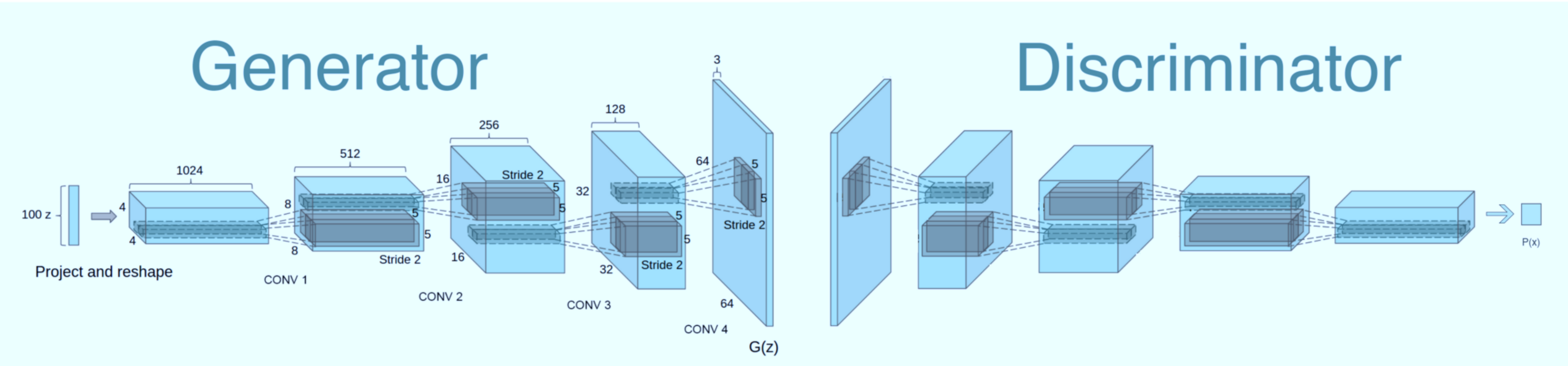
By observation, G discovers more detailed features of tough guys: e.g., fighting

So G learns to imitate that as well

…and eventually tricks D.

Source: Arthur Juliani, «*Generative Adversarial Networks Explained with a Classic Spongebob Squarepants Episode*», 2016, https://medium.com/@awjuliani/generative-adversarial-networks-explained-with-a-classic-spongebob-squarepants-episode-54deab2fce39#.gcoxuaruk

GAN model formulation (improved)
Deep convolutional generative adversarial nets [2]



Implement both G and D as deep convnets (DCGAN)

- **No pooling**, only fractional-strided convolutions (G) and strided convolutions (D)
- Apply **batchnorm** in both
- **No fully connected** hidden **layers** for deeper architectures
- **ReLU** activation **in G** (output layer: tanh)
- **LeakyReLU** activation in **D** (all layers)

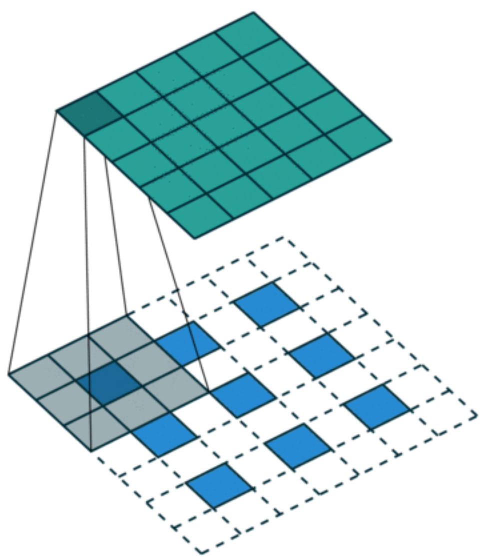
[2] Radford, Metz, Chintala, «Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks», 2016

Zurich University
of Applied Sciences

Strided what?
Convolutional arithmetic [3]

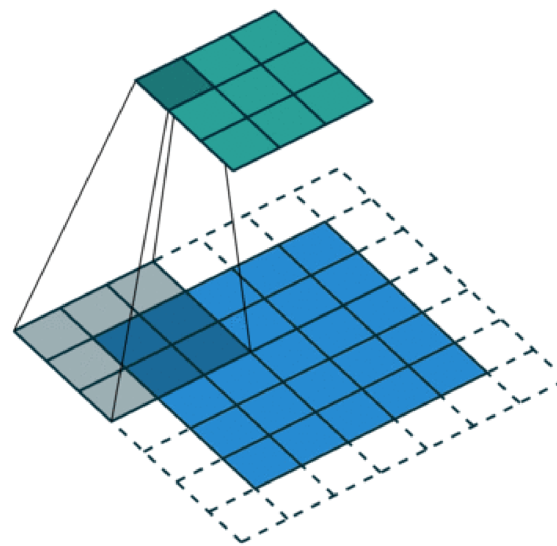
Fractionally-strided conv. in G

- Performing transposed convolution
- Used to «up-sample» from input (blue) to output (green)



Strided convolutions in D

- Stride (**stepsize**) = 2
- Used instead of (max) pooling [4]



[3] Dumoulin, Visin, «A guide to convolution arithmetic for deep learning », 2016
[4] Springenberg, Dosovitsiy, Brox, Riedmiller, «Striving for simplicity: The all convolutional net», 2014

Model training [5]

Usually $k = 1$ (or ½)

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:

change θ_D to **maximize**

$$\nabla_{\theta_d} \underbrace{\frac{1}{m} \sum_{i=1}^{m}}_{\text{average}} \left[\underbrace{\log D\left(\boldsymbol{x}^{(i)}\right)}_{\text{log likelihood of } x \text{ being real} \to 0} + \underbrace{\log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)}_{\text{log likelihood } G(z) \text{ not being real} \to 0} \right].$$

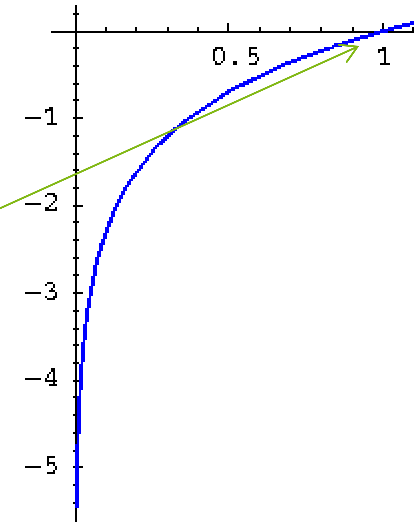
end for

- Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

change θ_G to **minimize**

$$\nabla_{\theta_g} \underbrace{\frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)}_{\text{see above}}.$$

end for



[5] Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio, «Generative Adversarial Nets», 2014

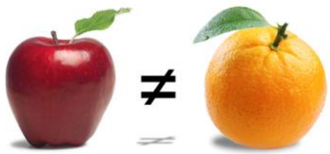
Visualizing the training process

Observations

- G starts with producing **random noise**
- Quickly arrives at what seems to be **pencil strokes**
- It takes a while for the network to produce **different images** for different z
- It takes nearly to the end before the synthesized **images per z stabilize** at certain digits

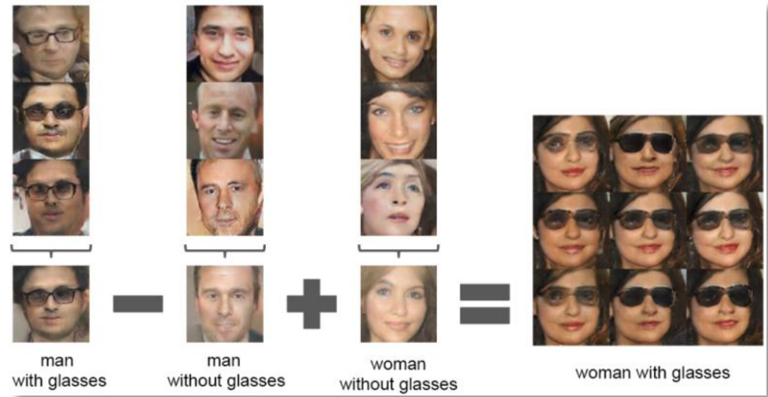
6x6 samples $G(z)$ from fixed z's every 2 mini batches (for 50k iterations). See https://dublin.zhaw.ch/~stdm/?p=400.

➔ Possible improvements?



Features of (DC)GANs

Learn semantically meaningful latent space
- Examples of z**-space vector arithmetic** from DCGAN paper [2]:



The z vectors in the left 3 columns have been averaged, then arithmetic has been performed. The middle image on the right is the output of $G(resulting\ z\ vector)$. The other 8 pictures are the result of adding noise to the resulting z vector (showing that smooth transitions in input space result in smooth transitions in output space).

Training is not guaranteed to converge
- D and G play a **game-theoretic game** against each other (in terms of slide 12: minimax)
- **Gradient descent isn't meant to find** the corresponding Nash Equilibria (saddle point of joint loss function, corresponding to minima of both player's costs) [6]
- How to **sync D's and G's training** is experimental (if G is trained too much, it may collapse all of z's variety to a single convincing output)
- The improvements of [2] and [7] make them **stable enough for first** practical **applications**
- **Research** on adversarial training of neural networks is still **in its infancy**

[6] Goodfellow, Courville, Bengio, «Deep Learning», ch. 20.10.4, 2016
[7] Salimans, Goodfellow, Zaremba, Cheung, «Improved Techniques for Training GANs», 2016

3. USE CASE: IMAGE INPAINTING

Based on material from Brandon Amos, *«Image Completion with Deep Learning in TensorFlow»*, 2016

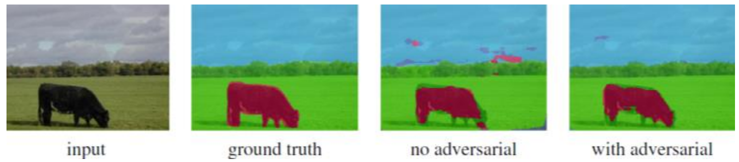
https://bamos.github.io/2016/08/09/deep-completion/

GAN use cases

Research is just **starting to gain momentum**; we expect more to see in the future

- Generate images from text
  Reed et al., *«Generative Adversarial Text to Image Synthesis»*, 2016

- Segment images into semantically meaningful parts
  Luc et al., *«Semantic Segmentation using Adversarial Networks»*, 2016



- Complete missing parts in images
  Yeh et al., *«Semantic Image Inpainting with Perceptual and Contextual Losses»*, 2016
  ➔ see next slides

Image inpainting as a sampling problem
…approached by machine learning

Training: Regard **images as samples of** some underlying probability distribution $\boldsymbol{p_G}$

1. Learn to represent this distribution using a GAN setup (G and D)

--

Testing: Draw a **suitable sample** from p_G by…

1. **Fixing parameters** $\boldsymbol{\Theta_G}$ and Θ_D of G and D, respectively
2. **Finding** input $\hat{\boldsymbol{z}}$ to G such that $\mathrm{G}(\hat{z})$ fits **two constraints**:
   a) **Contextual**: Output has to **match** the **known parts of** the **image** that needs inpainting
   b) **Perceptual**: Output has to **look generally «real»** according to D's judgment
3. …**by** using gradient-based **optimization on** $\hat{\boldsymbol{z}}$

Powerful idea: application of trained ML model may again involve optimization!

Reconstruction formulation

Given

- Uncomplete/corrupted image $x_{corrputed}$
- Binary mask M (same size as$x_{corrputed}$, 0 for missing/corrupted pixels)
- Generator network $G()$, discriminator network $D()$

$$\underset{\text{Input}}{\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}} \odot \underset{\text{Binary Mask}}{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}} = \underset{\text{Output}}{\begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}}$$

Problem

- Find $\hat{z}$ such that $x_{reconstructed} = M \odot x_{corrputed} + (1 - M) \odot G(\hat{z})$
  ($\odot$ is the element-wise product of two matrices)

Solution

- Define contextual and perceptual loss as follows:

  $L_{contextual}(z) = \left\| M \odot G(z) - M \odot x_{corrupted} \right\|_1$ (distance between known parts of image and reconstruction)

  $L_{perceptual}(z) = \log\left(1 - D\big(G(z)\big)\right)$ (as before: log-likelihood of $G(z)$ being real according to D)

  $L(z) = L_{contextual}(z) + \lambda \cdot L_{perceptual(z)}$ (combined loss)

➔ Optimize $\hat{z} = \underset{z}{\arg\min}\, L(z)$

Results

See it move: https://github.com/bamos/dcgan-completion.tensorflow

Review

- **Generative models capture** important aspects of the data-generating **distribution**
- They can be **used to sample** from even if the **pdf isn't** modeled **explicit**ly

- **GANs** have been shown to **produce realistic output** on a wide class of (still smallish) image, audio and text generation tasks

- **Finding Nash equilibria** in high-dimensional, continuous, non-convex games **is an** important **open** research **problem**

- **Image inpainting works by optimizing the output** of a fully trained generator to fit the given context & realism criteria, **using** again **gradient descent**

  ➔ Applying machine learned models might involve optimization (~training) steps again

  ➔ **This is in line with human learning**: Once trained to draw, hand-copying a painting involves "optimization" on the part of the painter