Machine Learning for Graphs and Sequential Data

Deep Generative Models - Generative Adversarial Networks

lecturer: Prof. Dr. Stephan Günnemann
www.daml.in.tum.de

Summer Term 2020

Data Analytics and
Machine Learning

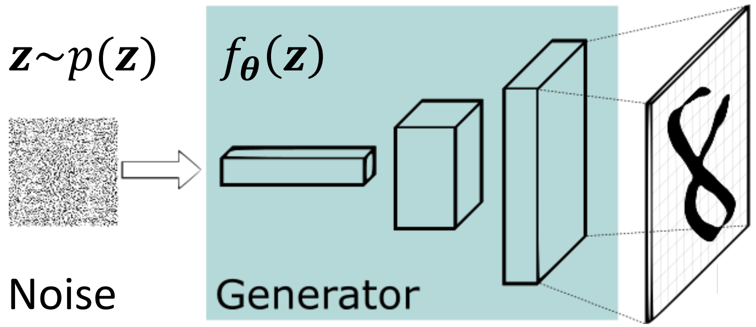
Roadmap

- Chapter: Deep Generative Models
  1. Introduction
  2. Normalizing Flows
  3. Variational Inference
  4. **Generative Adversarial Networks**
  5. Summary

Learn a Generative Model with Fewer Assumptions

- The variational autoencoder assumes a latent variable structure and specific forms of the likelihood and the variational posterior
- Can we model the data better by having fewer assumptions?

- Idea: Transform given initial noise using a flexible function
  1. Draw an initial noise vector from the prior $\mathbf{z} \sim p(\mathbf{z})$
  2. Deterministically transform $\boldsymbol{z}$ into the target data/output space $\boldsymbol{x} = f_{\boldsymbol{\theta}}(\mathbf{z})$



  – In contrast to Normalizing Flows, $f_{\boldsymbol{\theta}}(\mathbf{z})$ can be any function

Resulting Distribution

1. Draw an initial noise vector from the prior $\boldsymbol{z} \sim p(\boldsymbol{z})$
2. Deterministically transform $\boldsymbol{z}$ into the target data/output space $\boldsymbol{x} = f_{\boldsymbol{\theta}}(\boldsymbol{z})$

- This process defines a valid density on the output space

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{d}{dx_1} \dots \frac{d}{dx_d} \int_{\{f_{\boldsymbol{\theta}}(\boldsymbol{z}) \leq \boldsymbol{x}\}} p(\boldsymbol{z}) d\boldsymbol{z}$$

- Unfortunately $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ is often intractable to compute
  - The integration region $\{f_{\boldsymbol{\theta}}(\boldsymbol{z}) \leq \boldsymbol{x}\}$ itself may be very hard to determine
  - The integral itself may then be computationally demanding
  - Taking d partial derivatives in a high-dimensional space could be challenging

Learn a Generative Model with Fewer Assumptions

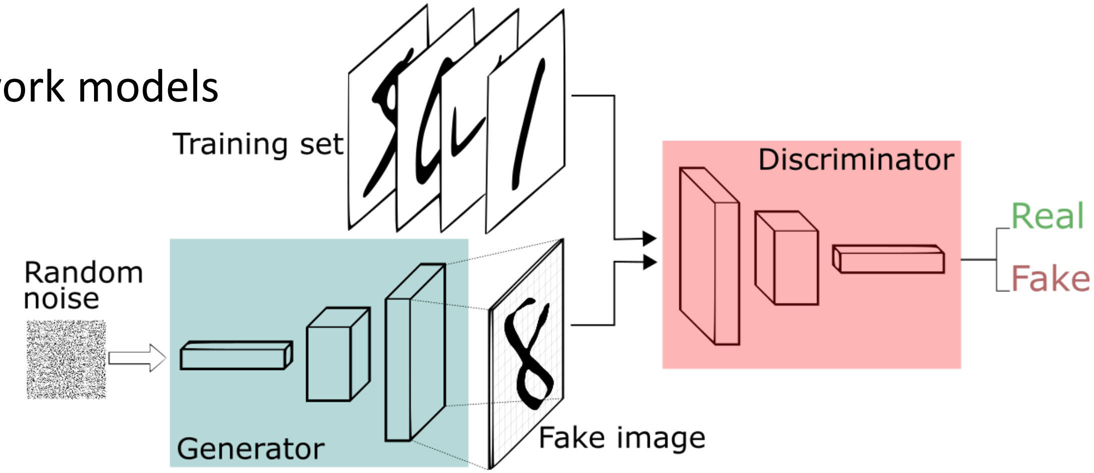
1. Draw an initial noise vector from the prior $\boldsymbol{z} \sim p(\boldsymbol{z})$
2. Deterministically transform $\boldsymbol{z}$ into the target data/output space $\boldsymbol{x} = f_{\boldsymbol{\theta}}(\boldsymbol{z})$

➢ Sampling is easy; density evaluation is hard!

- What can we do if we can only sample but we cannot evaluate the density?
  - Without a (tractable) **likelihood function**, many widely-used tools for inference and parameter learning become **unavailable** (e.g. the previously seen variational inference)

Forward Pointer: Generative Adversarial Networks

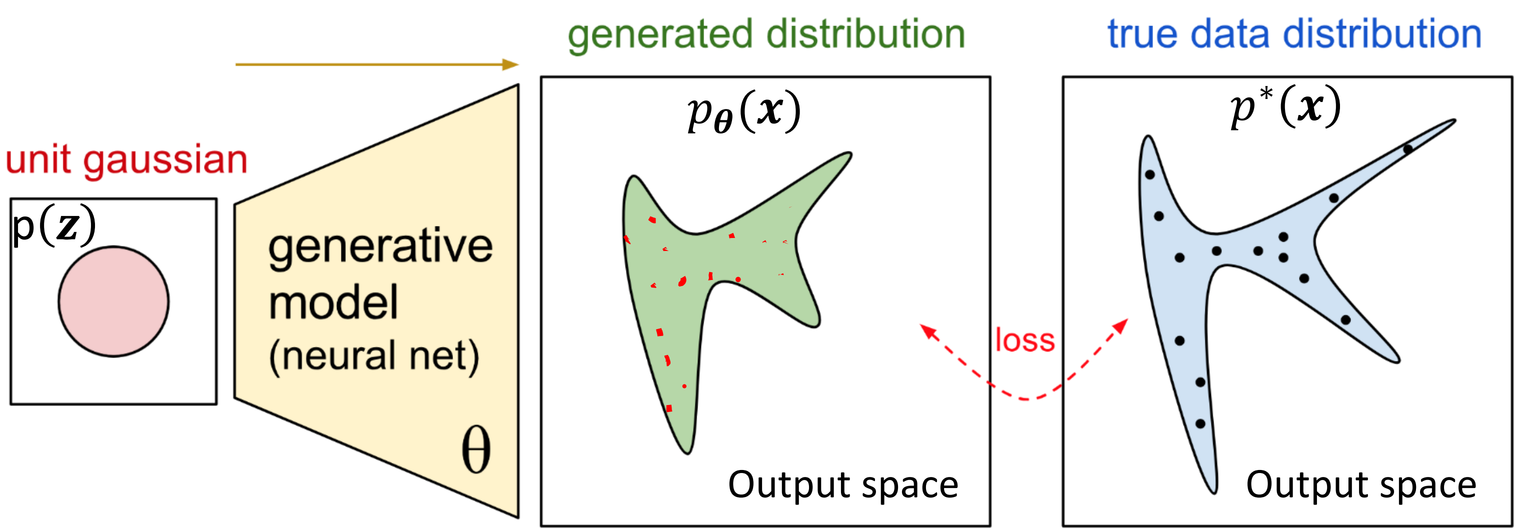
- GANs have become very popular for learning deep generative models
- Informally, the main idea is:
  - Two competing neural network models
  - Generator: takes noise as input and generates ("fake") samples
  - Discriminator: receives samples from both generator and training data and has to distinguish between the two → classify input as "real" or "fake"
  - Goal: Train the generator in such a way that the discriminator can **not** distinguish between real and "fake" samples
    - ➢ In this case, the generator generates realistic examples



In the following, you learn the actual (and more general) concept behind it

Likelihood-free inference

- What can we do if we can **easily draw samples** from the model but we cannot evaluate the density?
- Idea: We can use **any** method that compares two **sets of samples**.
  - This process is called ***density estimation by comparison.***
- We **test** the **hypothesis** that the true data distribution $p^*(\boldsymbol{x})$ and the model distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ are **equal**



Density Estimation by Comparison

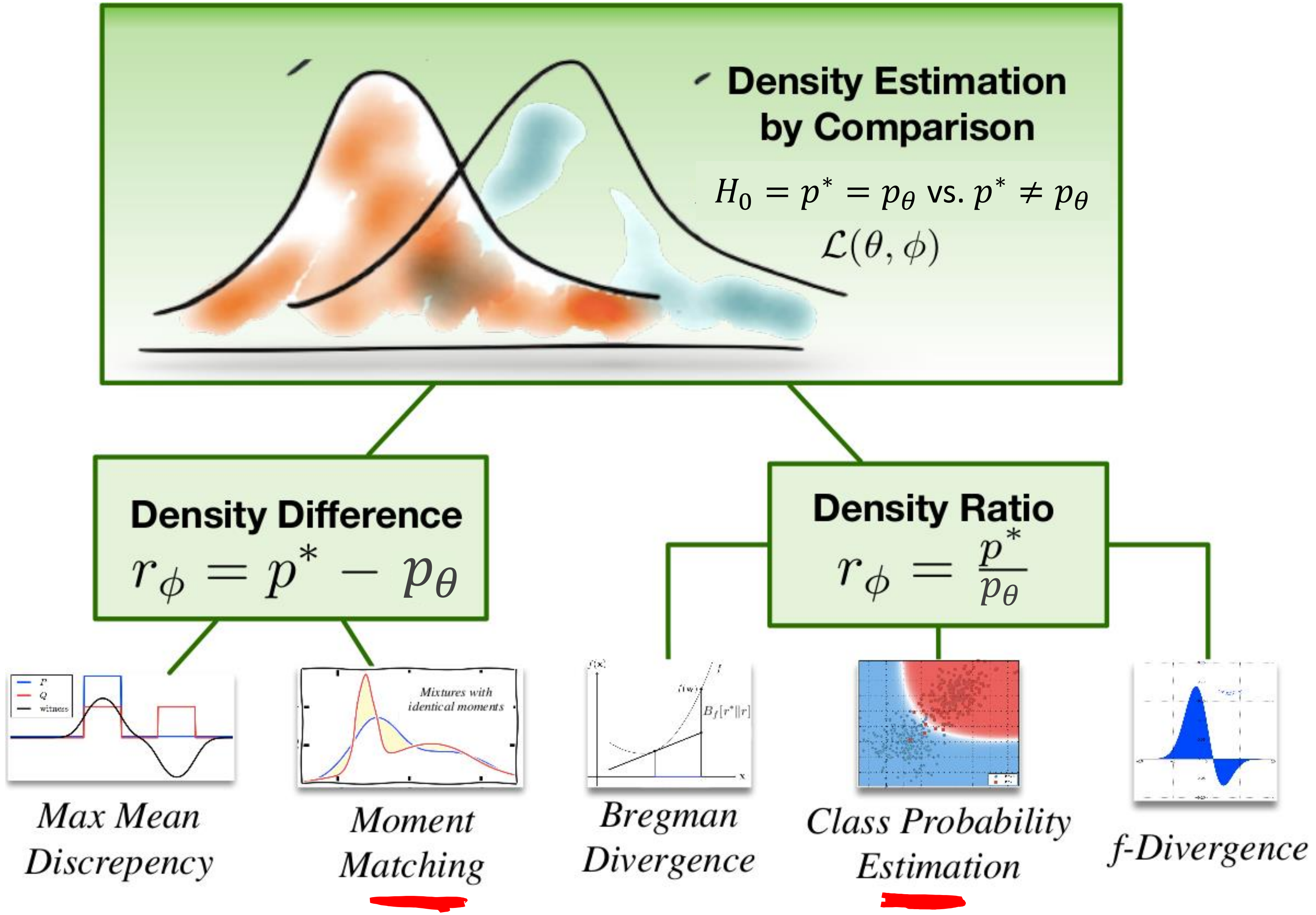


Figure: [Mohamed and Lakshminarayanan, 2016]

Density Difference Approaches (I)

$s(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$

- Moment Matching
  - Use the fact that p^* and p_θ are identical iff the expectations of any test statistic $s(\boldsymbol{x})$ are identical
  - Thus, minimize the gap $\min_\theta \left(\mathbb{E}_{p^*(\boldsymbol{x})}[s(\boldsymbol{x})] - \mathbb{E}_{p(\boldsymbol{z})}[s(\underbrace{f_\theta(\boldsymbol{z})}_{\tilde{x}})]\right)^2$
  - Approximate expectation by Monte-Carlo samples

$\mathbb{E}_{p^*}[x] \stackrel{?}{=} \mathbb{E}_{p_\theta}[x]$

$\mathbb{E}_{p^*}[x^2] \stackrel{?}{=} \mathbb{E}_{p_\theta}[x^2]$

$\vdots$

$\mathbb{E}_{p^*}[s(x)] \stackrel{?}{=} \mathbb{E}_{p_\theta}[s(x)]$

$\frac{1}{N} \sum_{i=1}^{N} s(x^{(i)})$

$z^{(1)}, \dots, z^{(N)} \sim p(z)$

$\tilde{x}^{(i)} = f_\theta(z^{(i)})$

$\frac{1}{N} \sum_{i=1}^{N} s(\tilde{x}^{(i)})$

Density Difference Approaches (II)

- Max Mean Discrepancy
  - When the statistics $s(\boldsymbol{x})$ are defined within a reproducing kernel Hilbert space, we obtain kernel-based forms of these objectives
  - These kernels are highly flexible and allow easy handling of data such as images, strings and graphs

Ratio-Based Approaches (I)

- Density ratio $r^*(\boldsymbol{x}) = p^*(\boldsymbol{x}) / p_\theta(\boldsymbol{x})$
  - In the best case always 1, i.e. the two distributions are indistinguishable
  - However, we cannot compute ratio in closed form/easily
- Idea: Approximate the true density ratio $r^*(\boldsymbol{x})$ by $r_\phi(\boldsymbol{x})$
  - Finding the approximation $r_\phi(\boldsymbol{x})$ often means solving again a learning problem

- Thus, we get the following general principle for learning
  - **Optimize Ratio loss**:
    approximate the true density ratio $r^*(\boldsymbol{x})$ (i.e. learning ϕ)
  - **Optimize Generative loss**:
    drive the density ratio towards 1 (i.e. learning θ)

  - Essentially a bi-level optimization problem, which is usually just solved alternatingly

Ratio-Based Approaches (II)

- The following three methods are an instantiation of this principle:
- Class Probability Estimation
  - Estimate the density ratio via a classifier which can distinguish between the observed data and data generated from the model
    (principle used in, e.g., **Generative Adversarial Networks (GANs)**)
- Divergence Minimization
  - Minimize an f-divergence between the true density and the generator density
  - For specific choices of the divergence function this can be equivalent to the class probability estimation case or KL divergence
- Ratio matching
  - Directly minimize the error between the true density ratio and an estimate of it:

$$\mathcal{L} = \mathbb{E}_{p_\theta}\left[\left(r_\phi(\boldsymbol{x}) - r^*(\boldsymbol{x})\right)^2\right]$$

  - Can be generalized beyond the squared error using the Bregman divergence

Learning via Class Probability Estimation (I)

$p(x,y) = p(y|x) \cdot p(x) = p(x|y) \cdot p(y)$

- Let Y denote a random variable which assigns label $Y = 1$ to samples from the true data distribution; $Y = 0$ to those from the generator distribution
- Then $p(\boldsymbol{x} \mid Y = 1) = p^*(\boldsymbol{x})$ and $p(\boldsymbol{x} \mid Y = 0) = p_\theta(\boldsymbol{x})$
- Denote $P(Y = 1) = \pi$. From Bayes we have:

$$r^*(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{p_\theta(\boldsymbol{x})} = \frac{p(Y = 1 \mid \boldsymbol{x})}{p(Y = 0 \mid \boldsymbol{x})} \frac{1 - \pi}{\pi}$$

  – Apparently, density ratio estimation is equal to **class-probability estimation**

  – Simply speaking: we can consider a classifier for $\boldsymbol{x}$ (predicting labels Y=0 or 1)

➢ Specify a scoring function or a **discriminator** $D_\phi(\boldsymbol{x}) = p(Y = 1 \mid \boldsymbol{x})$

- e.g. logistic regression or a neural network

Learning via Class Probability Estimation (II)

➢ Specify a scoring function or a **discriminator** $D_\phi(\boldsymbol{x}) = p(Y = 1 \,|\boldsymbol{x})$

- that tells you whether a sample $\boldsymbol{x}$ is real or from the generator ("fake")
- e.g. logistic regression or a neural network

▪ For learning $D_\phi(\boldsymbol{x})$, we need a loss function, e.g., the cross-entropy loss:

$$\mathcal{L}_{\theta,\phi} = \mathbb{E}_{(\boldsymbol{x},y)\sim p(\boldsymbol{x},y)}\left[-y\log[D_\phi(\boldsymbol{x})] - (1-y)\log[\overbrace{1 - D_\phi(\boldsymbol{x})}^{p(y=0|x)}]\right]$$

$$= \pi\mathbb{E}_{p^*(\boldsymbol{x})}\left[-\log D_\phi(\boldsymbol{x})\right] + (1-\pi)\mathbb{E}_{p(\boldsymbol{z})}\left[-\log\left[1 - D_\phi(\underbrace{f_\theta(\boldsymbol{z})}_{\tilde{x}})\right]\right]$$

y

$(x^{(1)}, 1)$ $(\tilde{x}^{(1)}, 0)$

$(x^{(2)}, 1)$ $(\tilde{x}^{(2)}, 0)$

$(x^{(3)}, 1)$ $(\tilde{x}^{(3)}, 0)$

Learning via Class Probability Estimation (III)

1. Solving $\phi^*(\theta) = \underset{\phi}{\operatorname{argmin}}\, \mathcal{L}_{\theta,\phi}$

   leads to the "best" discriminator for a given generative model (θ)

- That is, we well approximate $r^*(\boldsymbol{x}) = \frac{p^*(\boldsymbol{x})}{p_\theta(\boldsymbol{x})} = \frac{p(Y=1\,|\,\boldsymbol{x})}{p(Y=0\,|\,\boldsymbol{x})} \approx \frac{D_{\phi^*(\theta)}(\boldsymbol{x})}{1-D_{\phi^*(\theta)}(\boldsymbol{x})}$

  – here w.l.o.g. we set $\pi = 0.5$

2. We aim to drive the density ratio $r^*(\boldsymbol{x})$ towards 1

   – aim: $p(Y = 1\,|\,\boldsymbol{x}) = p(Y = 0|\,\boldsymbol{x})$

- That is, find generative model (θ) such that (even) the "best" discriminator cannot distinguish the classes

➢ Solving $\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathcal{L}_{\theta,\phi^*(\theta)}$

$\max_{G} \min_{\phi} L_{G,\phi}$

Learning via Class Probability Estimation (IV)

Attention: Note the flipped sign in the loss formulation! $-\mathcal{L}_{\theta,\phi}$

- Generator and discriminator play a **minimax game**:

$$\min_{\theta} \max_{\phi} \pi \mathbb{E}_{p^*(\boldsymbol{x})}\left[\log D_{\phi}\left(\boldsymbol{x}\right)\right] + (1-\pi)\mathbb{E}_{p(\boldsymbol{z})}\left[\log\left[1 - D_{\phi}\left(f_{\theta}(\boldsymbol{z})\right)\right]\right]$$

  - Discriminator: aims to distinguish between (samples from) $p^*(\boldsymbol{x})$ and $p_\theta(\boldsymbol{x})$
    - Maximization

// minimization of cross-entropy

  - Generator: aims to generate samples that are indistinguishable
    - Minimization

// maximization of (lowest) cross-entropy

- This is a bilevel optimization problem

Learning via Class Probability Estimation (V)

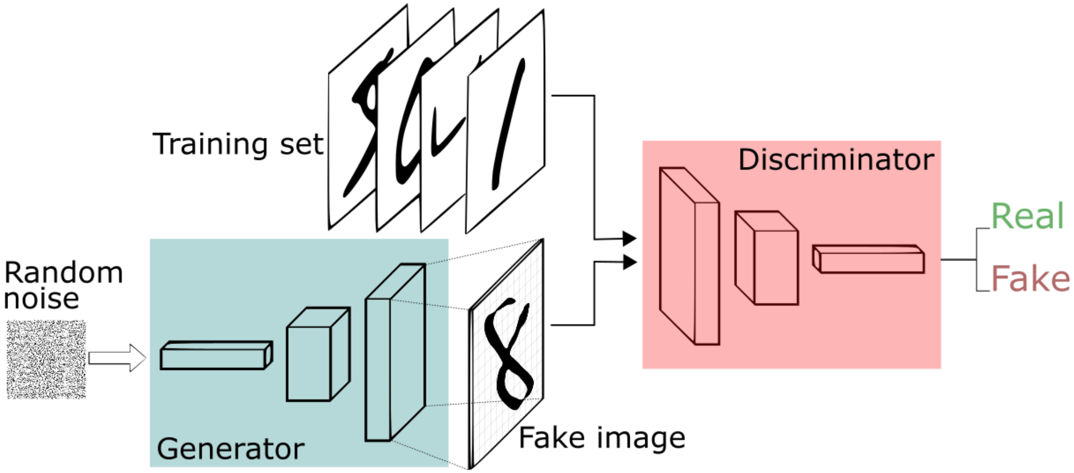
- This bilevel problem is typically tackled via alternating optimization
  - usually does not lead to the optimal solution
  - depending on the setting, difficult to train (instable)

- **Ratio loss** (discriminator loss) optimization:
$$\min_{\phi} \pi \mathbb{E}_{p^*(\boldsymbol{x})}\left[-\log D_\phi\left(\boldsymbol{x}\right)\right] + (1-\pi)\mathbb{E}_{p(\boldsymbol{z})}\left[-\log\left[1 - D_\phi\left(f_\theta(\boldsymbol{z})\right)\right]\right]$$

- **Generative loss** optimization:
$$\min_{\theta} \mathbb{E}_{p(\boldsymbol{z})}\left[\log\left[1 - D_\phi\left(f_\theta(\boldsymbol{z})\right)\right]\right]$$

GANs – Generative Adversarial Networks

- Main idea
  - Two competing neural network models
  - Generator: takes noise as input and generates ("fake") samples
  - Discriminator: receives samples from both generator and training data and has to distinguish between the two
  - They play a minimax game against each other
  - Competition drives the generated samples to be indistinguishable from real



- In short: GAN approach =
  Learning of a generator via class probability estimation where the generator and the discriminator are neural networks

GANs – Examples

- Synthetically generated images

Figure 5: 1024 × 1024 images generated using the CELEBA-HQ dataset. See Appendix F for a larger set of results, and the accompanying video for latent space interpolations.

From "Progressive Growing of GANs for Improved Quality, Stability, and Variation", Tero Karras et.al.

Questions – GANs

1. What can we say about the ratio $r^*(\boldsymbol{x}) = p^*(\boldsymbol{x}) / p_\theta(\boldsymbol{x})$ when:
   a) The generator and the discriminator are optimal
   b) The generator only is optimal
   c) The discriminator only is optimal

2. For a given data $\boldsymbol{x}$, what is the probability for the discriminator to be correct when:
   a) The generator and the discriminator are optimal
   b) The generator only is optimal
   c) The discriminator only is optimal

References

- [1] Mohamed, Shakir, and Balaji Lakshminarayanan. “Learning in Implicit Generative Models.”: https://arxiv.org/pdf/1610.03483.pdf
- [2] Andrew Davison blog : https://casmls.github.io/general/2017/05/24/ligm.html

External Sources

Web tutorial:

- Andrew Davison blog : https://casmls.github.io/general/2017/05/24/ligm.html
  Eric Jang blog : https://blog.evjang.com/2018/01/nf1.html

Survey papers:

- Mohamed, Shakir, and Balaji Lakshminarayanan. "Learning in Implicit Generative Models.": https://arxiv.org/pdf/1610.03483.pdf