Winter Term 2026/27

Selected Topics in Deep Learning for Audio, Speech, and Music Processing

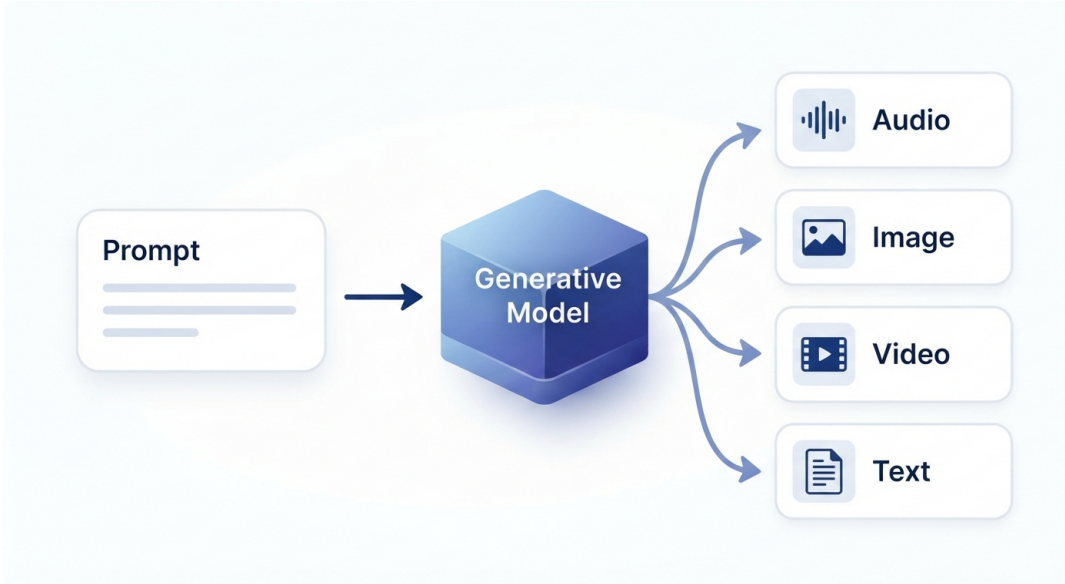
Lecture

Introduction to Generative Models for Audio Processing

Stefan Balke, **Andreas Brendel**, Ben Maman, Meinard Müller

Friedrich-Alexander-Universität Erlangen-Nürnberg

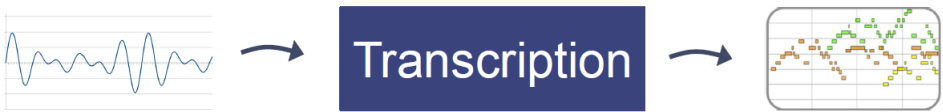
Fraunhofer IIS


Prompt
Generative Model
Audio
Image
Video
Text

Plan for this Lecture

- Generative vs discriminative models
- Conditional generation
- Deep generative modeling paradigms
- Autoregressive models
- GANs

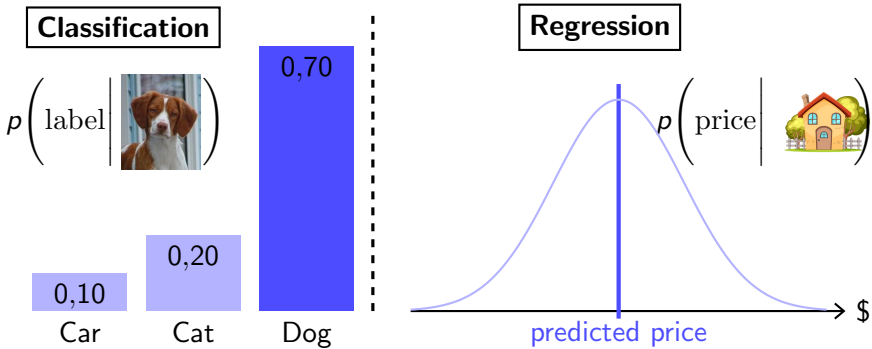
DNN Application Examples for Audio



- MIR: Music transcription, beat tracking, chord recognition, etc.
- Speech/speaker/language recognition
- Acoustic event detection
- Speech enhancement
- Acoustic parameter estimation
- Signal quality estimation
- ...

Discriminative Models (DiscMs)

- All examples: Learn mapping of feature vector $\mathbf{x}$ to most likely target value $\mathbf{y}$
  $\Rightarrow$ Learn $p(\mathbf{y}|\mathbf{x})$ and pick $\text{argmax}_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$!



- Statistical assumptions yield loss functions (here: cross entropy, MSE)
- Often, directly a discriminative function is trained
- But DiscMs don't model data generation process

We Present Data in Vector Form

Audio might be represented

- as waveform

$$\mathbf{x} \in \mathbb{R}^T$$

- complex-valued time-frequency representation

$$\mathbf{X} \in \mathbb{C}^{F \times N}$$

- magnitude/phase or real/imaginary part

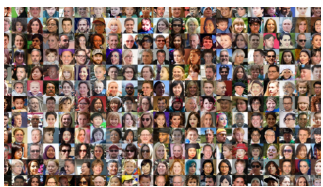
$$\mathbf{X} \in \mathbb{R}^{F \times N \times 2}$$

We represent this in the following always as (real-valued) vector

$$\mathbf{x} \in \mathbb{R}^D$$

Generation = Sampling from Data Distribution

training data **learn data distribution** **sample new object**



$\rightarrow \quad \mathbf{x} \sim p_{\text{data}}(\mathbf{x}) \quad \rightarrow \quad \mathbf{x} =$

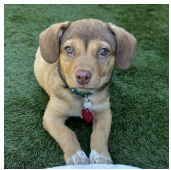


Problem: Data distribution $p_{\text{data}}(\mathbf{x})$ unknown $\Rightarrow$ Learn it!

Image: https://edition.cnn.com/2019/04/19/tech/ai-facial-recognition/index.html

How to Measure Success . . . of Generating a Dog Picture?



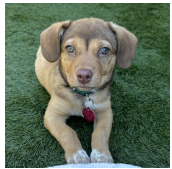
Useless $<$ **Useless image** $<$ **Wrong animal** $<$ **Great!**

These are subjective statements $\rightarrow$ **Formalize this!**

Evaluate the Data Distribution!

How "likely" would someone perceive this as a natural dog picture?



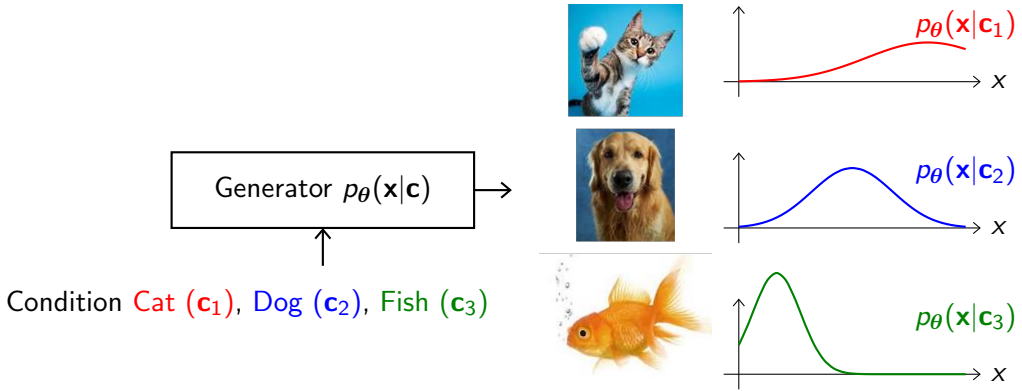



Impossible $<$ **Rare** $<$ **Unlikely** $<$ **Very likely**

$p_{\text{data}}(\mathbf{x}) = 0$ $\quad$ $p_{\text{data}}(\mathbf{x}) \ll 1$ $\quad$ $p_{\text{data}}(\mathbf{x}) < 1$ $\quad$ $p_{\text{data}}(\mathbf{x}) = 1$

Object's naturalness $\approx$ Likeliness under data distribution

Conditional Distributions

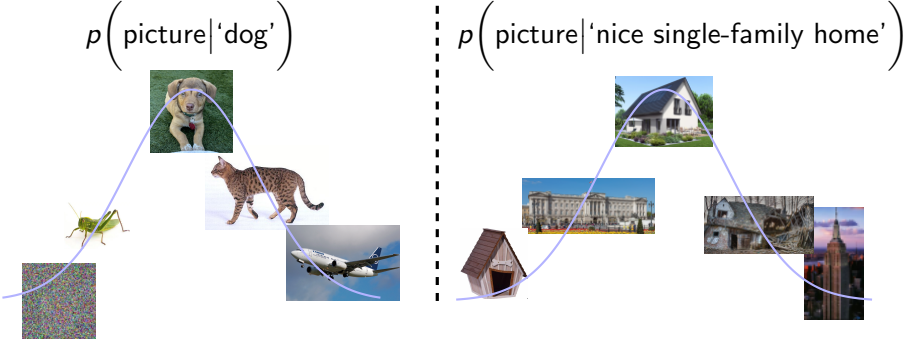


- Conditioning 'customizes' a PDF

$$p(\mathbf{x}|\mathbf{c}) = \frac{p(\mathbf{x}, \mathbf{c})}{p(\mathbf{c})}$$

  Interpretation: PDF of $\mathbf{x}$ over event subset for which $\mathbf{c}$ occurs
- **Multi-tasking:** Specialized PDF/generative model for each condition!

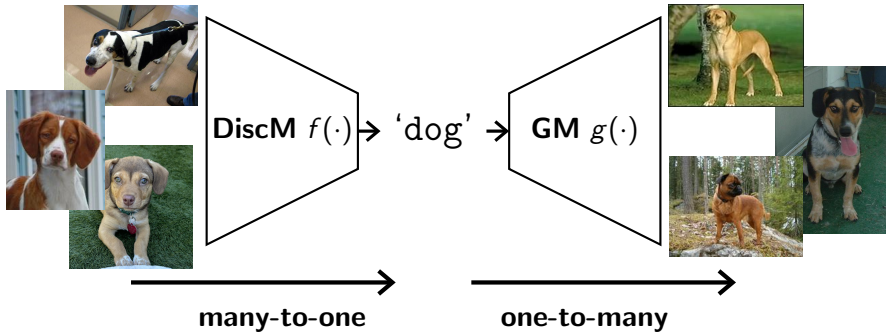
(Conditional) Generative Models (GMs)



Generative models (implicitly) model the data distribution!

Generate new objects by (implicitly) sampling from model distribution

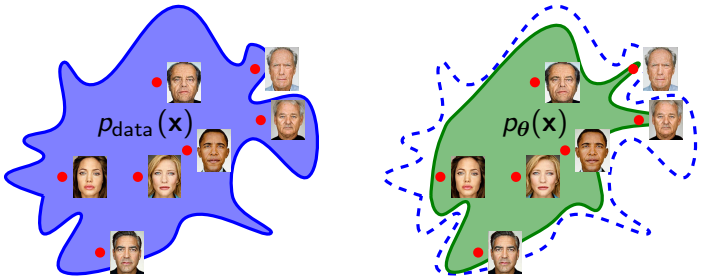
Discriminative vs. (Conditional) Generative Models (GMs)



- Address GMs' one-to-many ambiguity: Make $g(\cdot)$ **stochastic**
- Conditional GM: Input guiding information ('dog')
- (Unconditional) GM: Fixed/no guiding information

Deep Generative Models (DGMs)

How to (approx.) sample from an unknown PDF $p_{\text{data}}(\mathbf{x})$, given observations $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, $\mathbf{x}_n \sim p_{\text{data}}(\mathbf{x}_n)$?



- Fit model from a model family $\mathcal{P}_\Theta$ (DNNs!) to data

$$p_\theta(\mathbf{x}) \in \mathcal{P}_\Theta : \; p_{\text{data}}(\mathbf{x}) \approx p_\theta(\mathbf{x}), \text{ for } \mathbf{x} \in \mathcal{X}$$

- Usually generation is enough: $\hat{\mathbf{x}} = \text{Generator}(\ldots)$ with $p_{\text{data}}(\hat{\mathbf{x}})$ large

Training DGMs: Key Questions

Data set $\mathcal{X}$: Which $\mathcal{X}$ is representative for $p_{\text{data}}(\mathbf{x})$?

Model $p_{\boldsymbol{\theta}}(\mathbf{x})$: Which parametric family $p_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathcal{P}_{\boldsymbol{\Theta}}$?

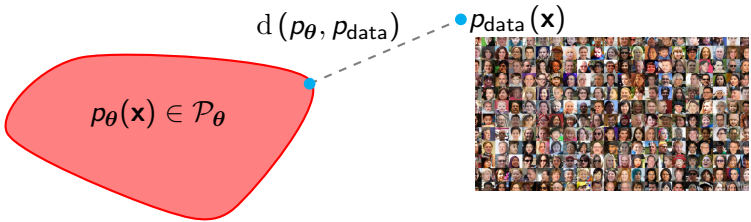
Loss $\mathcal{L}(\mathcal{X}, \boldsymbol{\theta})$: How 'far' is $p_{\boldsymbol{\theta}}(\mathbf{x})$ from $p_{\text{data}}(\mathbf{x})$?

Optimization $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta}}{\operatorname{argmin}} \mathcal{L}(\mathcal{X}, \boldsymbol{\theta})$: How to find best $\boldsymbol{\theta}$?

Sampling $\hat{\mathbf{x}} \sim p_{\boldsymbol{\theta}}(\mathbf{x})$: How to generate new objects?

Inference $p_{\boldsymbol{\theta}}(\mathbf{x}) = ?$: How to evaluate the trained PDF? (optional)

GMs: Problems



- $\mathbf{x} \in \mathbb{R}^D \sim p_{\text{data}}(\mathbf{x})$, with D huge (video, image, **audio**, ...)
- Measuring success (i.e., $\mathrm{d}(\cdot,\cdot)$) difficult
- Finding tractable model $p_{\boldsymbol{\theta}}(\mathbf{x})$ (trainable/sampling possible)
- $p_{\text{data}}(\mathbf{x})$ unknown, only samples of it observable

Aim of this lecture:

- How to (at least implicitly) learn and represent $p_{\boldsymbol{\theta}}(\mathbf{x})$?
- **How to sample** from $p_{\boldsymbol{\theta}}(\mathbf{x})$?

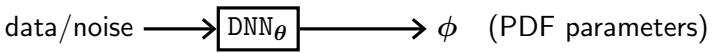
Image: https://edition.cnn.com/2019/04/19/tech/ai-facial-recognition/index.html

Literature on Deep Generative Models

- **Books:**
  - Tomczak: *Deep Generative Modeling*, 2nd ed., Springer, 2024
  - Foster: *Generative Deep Learning*. O'Reilly, 2023
  - Prince: *Understanding Deep Learning*, MIT Press, 2024
  - Murphy: *Probabilistic Machine Learning: Advanced Topics*, MIT Press, 2023
- **Courses** at other universities:
  - sites.google.com/view/berkeley-cs294-158-sp20/home
  - deepgenerativemodels.github.io/
  - courses.cs.washington.edu/courses/cse599i/20au/
- **Blogs**, e.g., lilianweng.github.io/
- **YouTube**: Many lectures, tutorials, paper discussions etc.
- Latest advancements: **Papers!**

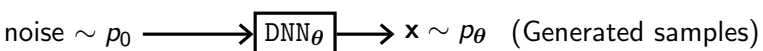
Deep Generative Modeling Paradigms

Parametric models



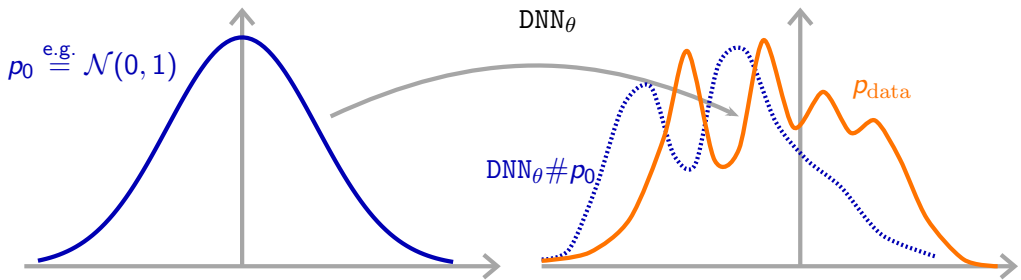
e.g., normal distribution $\mathcal{N}(\mu, \sigma^2)$ with $\phi = [\mu, \sigma]$

Transformation of a simple distribution (pushforward)



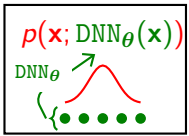
e.g., transform Gaussian noise $p_0 = \mathcal{N}(\mathbf{0}, \mathbf{I})$

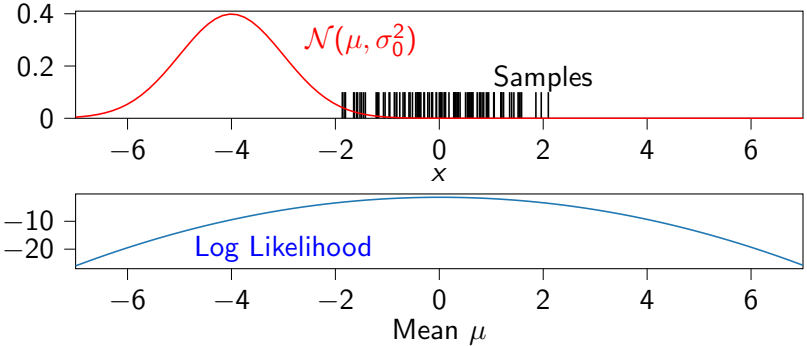
Learning the Pushforward



Have to ensure that $\texttt{DNN}_\theta \# p_0$ is a valid PDF!

Parametric Models: Estimate PDF Parameters with DNNs





- How to estimate PDF's parameters? $\rightarrow$ Train $\mathrm{DNN}_{\boldsymbol{\theta}}(\mathbf{x})$ such that

$$p_{\boldsymbol{\theta}}(\mathbf{x}_i) = p(\mathbf{x}_i; \mathrm{DNN}_{\boldsymbol{\theta}}(\mathbf{x}_i))$$

becomes as large as possible on train set $\{\mathbf{x}_i\}_{i=1}^{I}$

Parametric Models: Estimating Parameters of a PDF

- Compute (different) PDF parameters $\phi_i = \texttt{DNN}_{\boldsymbol{\theta}}(\mathbf{y}_i)$ for each training input $\mathbf{x}_i$
- Choose DNN params $\boldsymbol{\theta}$ s.t. training data is maximally probable under the model

$$\boldsymbol{\theta}^* = \text{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^{I} p(\mathbf{x}_i; \phi_i) = \text{argmax}_{\boldsymbol{\theta}} \prod_{i=1}^{I} p(\mathbf{x}_i; \texttt{DNN}_{\boldsymbol{\theta}}(\mathbf{x}_i))$$

  Assume i.i.d.: Training data is independent and indentically distributed

- Minimizing negative log-likelihood

$$\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} - \log \prod_{i=1}^{I} p(\mathbf{x}_i; \texttt{DNN}_{\boldsymbol{\theta}}(\mathbf{x}_i)) = \text{argmin}_{\boldsymbol{\theta}} - \sum_{i=1}^{I} \log p(\mathbf{x}_i; \texttt{DNN}_{\boldsymbol{\theta}}(\mathbf{x}_i))$$

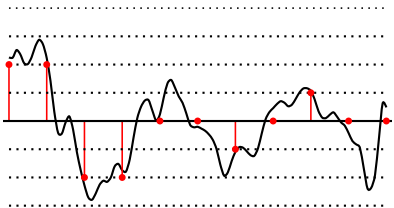
- Example: Estimate mean of Gaussian PDF $\mathcal{N}(\mu, \sigma_0^2)$

Summary: Generative Models

- Discriminative models output point predictions
- GMs (at least implicitly) represent PDFs
- **GMs model data generation process** by approximating $p_{\text{data}}(\mathbf{x})$ explicitely or implicitly
- **Sampling** from trained model PDF $\rightarrow$ Generate new objects
- Modeling paradigms: Parametric models and transform of simple distribution

Autoregressive Models

Curse of Dimensionality



- Consider N samples of an audio signal quantized with $\log_2 B$ bits per sample
- **Aim:** Model joint PMF $p(x_1, \ldots, x_N)$
- There are B^N possible sequences

- **Naive idea I (categorical distribution):** Exhaustive modeling

$$p_j \approx \frac{\#j\text{th sequence is observed}}{\text{all observed sequences}}, \quad j = 1, \ldots, J \ (J \text{ parameters})$$

Intractable: E.g., 10ms audio at 44.1kHz ($N = 441$) with 16 bit depth ($B = 2^{16}$) $\Rightarrow J = B^N - 1 = 2^{16^{441}} - 1 \approx 10^{3.13 \cdot 10^{530}}$ (for comparison: $\approx 10^{80}$ atoms in universe)

- **Naive idea II (samples are i.i.d.):** $p(x_1, \ldots, x_N) = \prod_{i=1}^{N} p(x_i) \Rightarrow J = B - 1$, but sampling just creates noise!

Divide and Conquer: Chain Rule

- **Chain rule** (recall: $p(a|b) = \frac{p(a,b)}{p(b)} \Rightarrow p(a,b) = p(a|b)p(b)$)

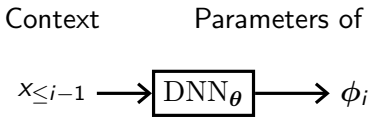
$$\begin{aligned} p(x_1, \ldots, x_N) &= p(x_N|x_{N-1}, \ldots, x_1)p(x_1, \ldots, x_{N-1}) \\ &= p(x_N|x_{N-1}, \ldots, x_1)p(x_{N-1}|x_{N-2}, \ldots, x_1) \cdots p(x_1) \\ &= \prod_{i=1}^{N} p(x_i|x_{\leq i-1}) \quad \text{with} \quad p(x_1|x_{\leq 0}) := p(x_1) \end{aligned}$$

- **Do we make progress?** We still have $J \approx B^N$!
- **Model assumption I (limit context length):** E.g., limit context to preceding RV

$$\begin{aligned} p(x_1, \ldots, x_N) &= p(x_N|x_{N-1}, \cancel{\ldots, x_1})p(x_{N-1}|x_{N-2}, \cancel{\ldots, x_1}) \cdots p(x_1) \\ &= p(x_N|x_{N-1})p(x_{N-1}|x_{N-2}) \cdots p(x_1) \end{aligned}$$

How many parameters? $J = (B-1)[(NB - B + 1]$

Modeling Conditional Distributions with DNNs

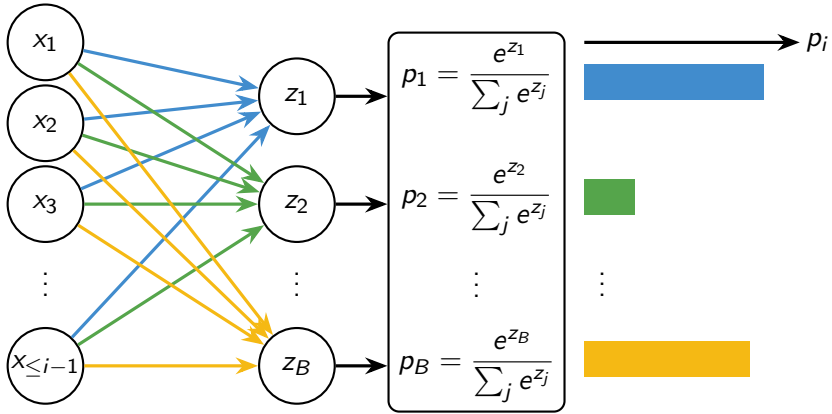


- **Model assumption II:** Approximate conditional distributions by DNNs parameterized by $\boldsymbol{\theta}$

$$p(x_1, \ldots, x_N) \approx \prod_{i=1}^{N} p_{\boldsymbol{\theta}}(x_i | x_{\leq i-1})$$

- How to ensure valid distributions? $\rightarrow$ **Model assumption III:** Choose parametric distributions

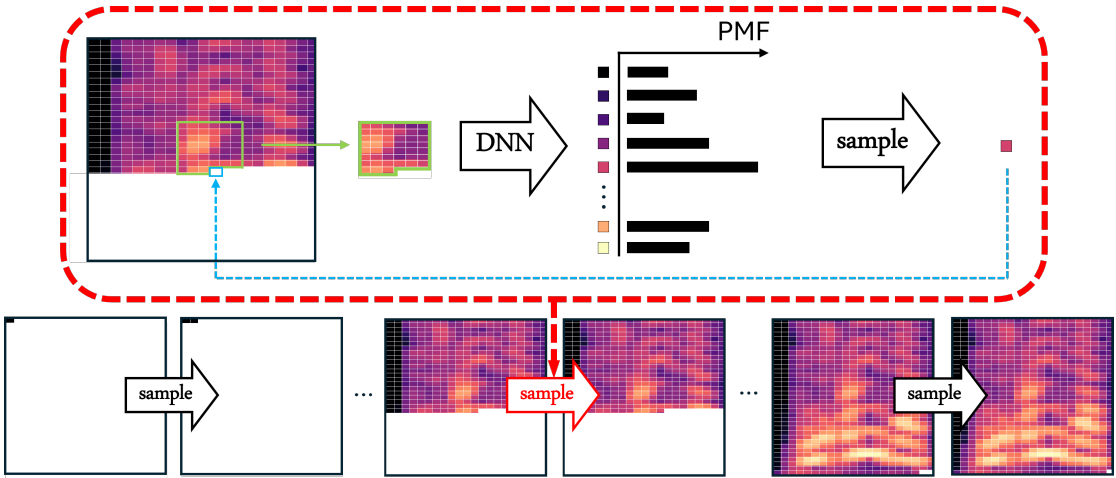
Modeling Conditional Distributions with DNNs



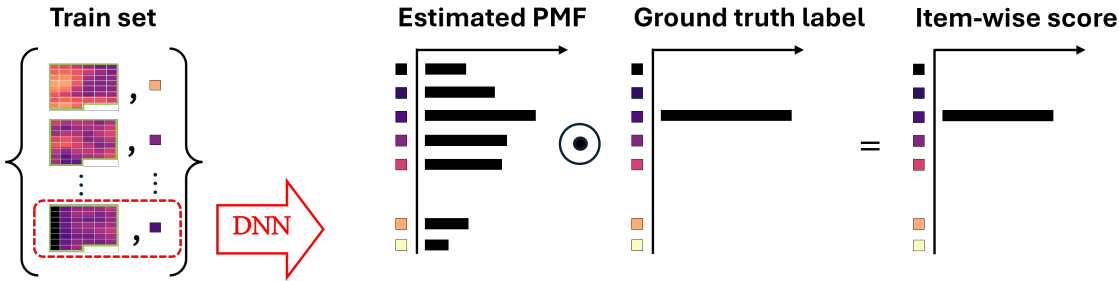
Examples for parametric distributions:

Data type	Distribution	Parameters	Realization
discrete	Categorical	$\phi_i = \{p_{ij}\}_{j=1}^{B}$	SoftMax
continuous	Gaussian	$\phi_i = \{\mu_i, \sigma_i^2\}$	linear, ReLU

Sampling from Autoregressive Models



Training Autoregressive Models

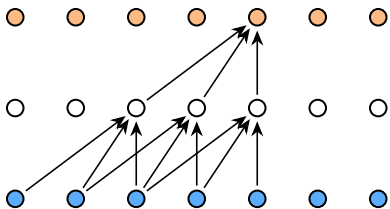


- **Teacher forcing:** Train with ground truth, do not sample during training
- **Loss:** Cross entropy or negative log likelihood

$$\text{argmin}_{\boldsymbol{\theta}} - \sum_{i=1}^{N} p^{\text{GT}}(x_i) \log p_{\boldsymbol{\theta}}(x_i) = \text{argmin}_{\boldsymbol{\theta}} - \log p_{\boldsymbol{\theta}}(x_{\text{GT}})$$

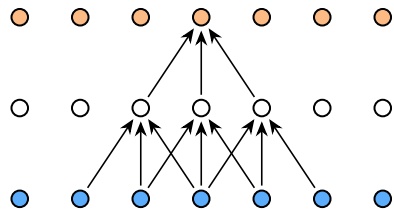
Causal Convolution

Causal Convolution



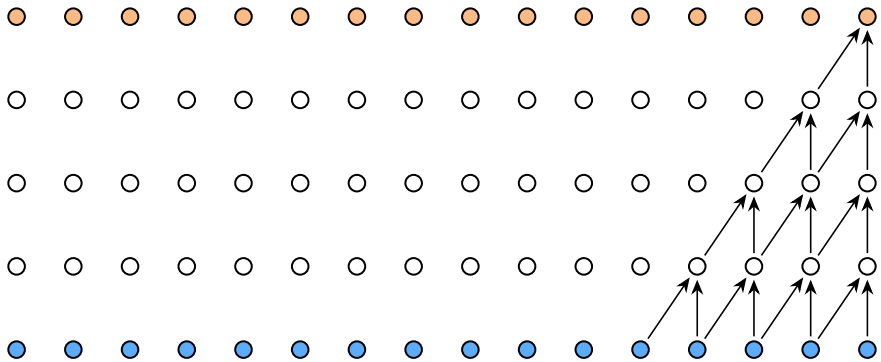
Symmetric field only looks into the past

Non-Causal Convolution



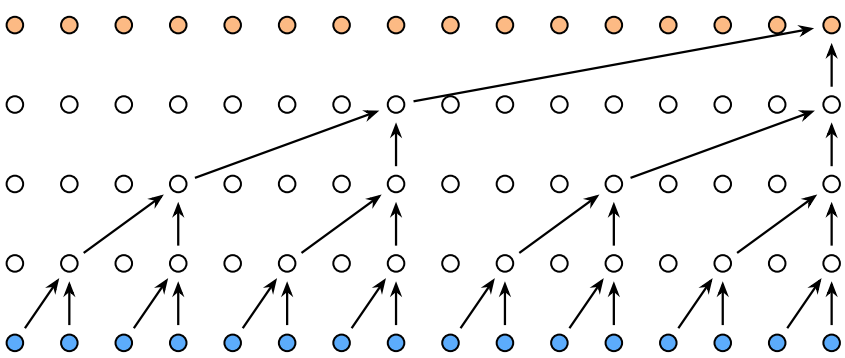
Symmetric receptive field
Problem: x_{i+1} is used for predicting x_i

Causal Autoregressive Model for Audio Generation



- Output is shifted version of input
- **Receptive field small!**

WaveNet



Dilated convolutions increase the receptive field!

Summary Autoregressive Models

- Chain rule and limitation of context length
- Autoregressive models provide density estimates in the form of parameters of distributions
- Sampling: Slow, generation sample-by-sample, feed back results to input
- Training: Maximum likelihood, teacher forcing

Pros/Cons

Pros:

- Log-likelihood is tractable and easy to calculate
- **Good sample quality**
- Teacher forcing permits fast training
- Modelling power is huge given enough scale!

Cons:

- Inference is very slow (auto-regressive)
- Teacher forcing leads to so called "broken-record" artifacts

Training DGMs: Key Questions answered for Autoregressive Models

Data set $\mathcal{X}$: Often discrete data

Model $p_{\boldsymbol{\theta}}(\mathbf{x})$: Conditional distribution, NN sees only the past

Loss $\mathcal{L}_{\text{AR}}(\mathcal{X}, \boldsymbol{\theta}, \boldsymbol{\phi})$: cross-entropy/negative log-likelihood

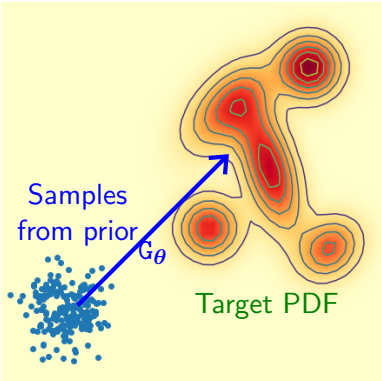
Optimization: Teacher forcing, gradient descent

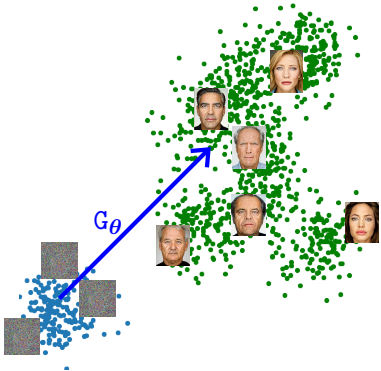
Sampling: Sample from conditional distribution and add result to the context

Inference $p_{\boldsymbol{\theta}}(\mathbf{x}) = ?$: Straight forward

Generative Adversarial Networks

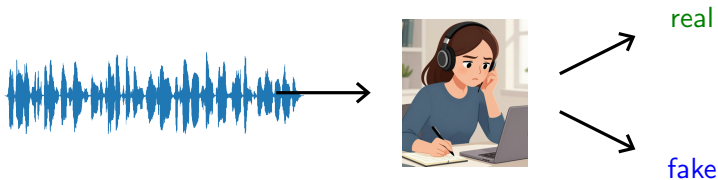
Generation by Trafo of Samples from Simple Prior





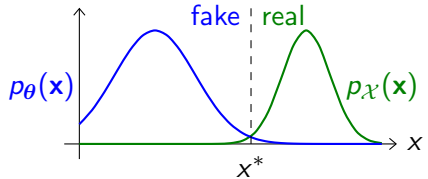
- **Want:** Trafo G_θ for $\mathbf{z} \sim p(\mathbf{z})$ such that $G_\theta(\mathbf{z}) \sim p_{\text{data}}(\mathbf{x})$
- However, generation would be enough
- Don't know $p_{\text{data}}(\mathbf{x})$, but we have samples $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$

How to Measure Success?



- **Want:** Indistinguishable samples from $p_\theta(\mathbf{x})$ and $p_{\text{data}}(\mathbf{x})$ (at least on train set)
- **Idea:** Test if sample more likely produced by $p_{\text{data}}(\mathbf{x})$ or $p_\theta(\mathbf{x})$
- **Automize this with an NN!**

Building a Classifier: The Discriminator



$$p(\mathbf{x}) = \begin{cases} \mathtt{D}_\phi(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{X} \\ 1 - \mathtt{D}_\phi(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{X}_{\text{fake}} \end{cases}$$

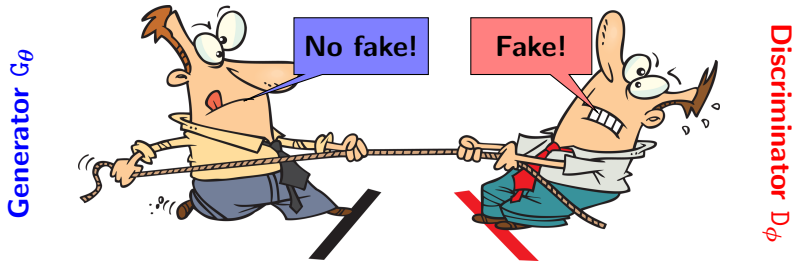
- **Generator** $\mathtt{G}_{\boldsymbol{\theta}} : \mathcal{Z} \to \mathcal{X}_{\text{fake}}$: DNN generating new (fake) data
- Train **DNN** $\mathtt{D}_{\boldsymbol{\phi}} : \mathcal{X} \cup \mathcal{X}_{\text{fake}} \to [0, 1]$ (**discriminator**/critic) as classifier

$$\begin{aligned} \text{real: } p(y = 1|\mathbf{x}) &= \mathtt{D}_{\boldsymbol{\phi}}(\mathbf{x}), && \text{for } \mathbf{x} \sim p_{\mathcal{X}}(\mathbf{x}) \\ \text{fake: } p(y = 0|\mathbf{x}) &= 1 - \mathtt{D}_{\boldsymbol{\phi}}(\mathbf{x}), && \text{for } \mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x}) \\ &= 1 - \mathtt{D}_{\boldsymbol{\phi}}(\mathtt{G}_{\boldsymbol{\theta}}(\mathbf{z})), && \text{for } \mathbf{z} \sim p(\mathbf{z}) \end{aligned}$$

- Cross-entropy loss function

$$\mathcal{L}_{\mathtt{D}}(\boldsymbol{\phi}) = \mathbb{E}_{p_{\mathcal{X}}(\mathbf{x})}\Big[\log \mathtt{D}_{\boldsymbol{\phi}}(\mathbf{x})\Big] + \mathbb{E}_{p(\mathbf{z})}\Big[\log\Big(1 - \mathtt{D}_{\boldsymbol{\phi}}\big(\mathtt{G}_{\boldsymbol{\theta}}(\mathbf{z})\big)\Big)\Big]$$

Generative Adversarial Networks (GANs)



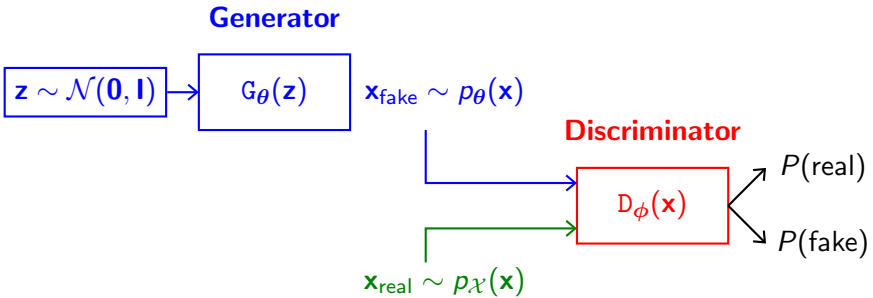
want $D_\phi\big(G_\theta(\mathbf{z})\big) = 1$

want $D_\phi\big(G_\theta(\mathbf{z})\big) = 0$
and $D_\phi(\mathbf{x}) = 1$

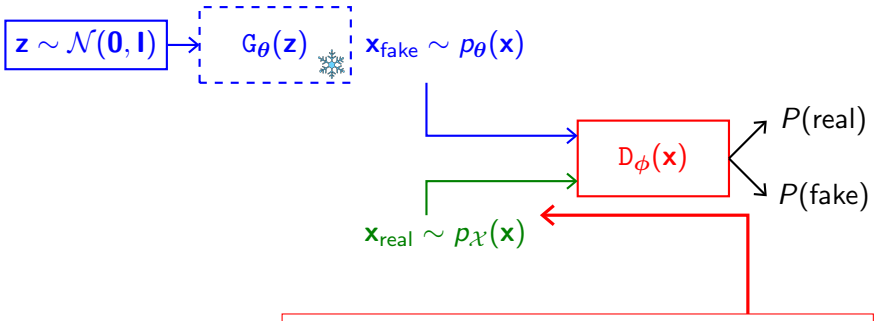
$$\min_\theta \max_\phi \mathbb{E}_{p_{\mathcal{X}}(\mathbf{x})}\Big[\underbrace{\log D_\phi(\mathbf{x})}_{\substack{\text{log-prob. for } D_\phi \text{ correctly} \\ \text{predicting real } \mathbf{x} \text{ to be real}}}\Big] + \mathbb{E}_{p(\mathbf{z})}\Big[\underbrace{\log\Big(1 - D_\phi\big(G_\theta(\mathbf{z})\big)\Big)}_{\substack{\text{log-prob. for } D_\phi \text{ correctly} \\ \text{predicting fake } \mathbf{x} \text{ to be fake}}}\Big]$$

Image: adapted from: https://de.cleanpng.com/png-rnois9/

GAN Overview



GAN Gradients: Discriminator



GAN Gradients: Generator

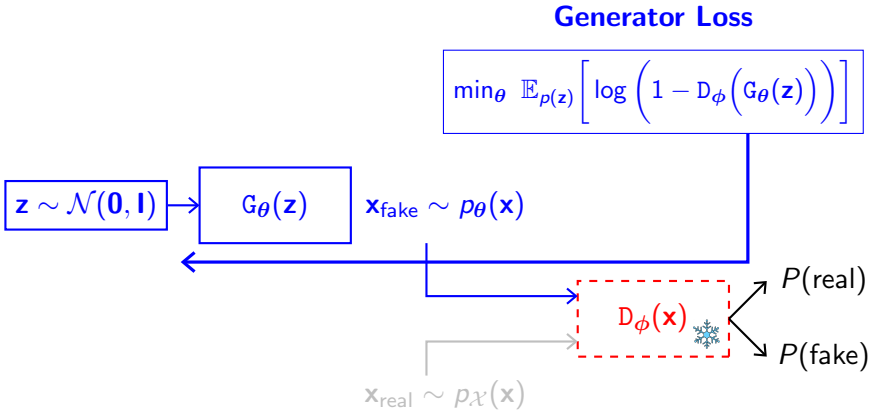


Illustration: GANs Learn Painting Like Picasso

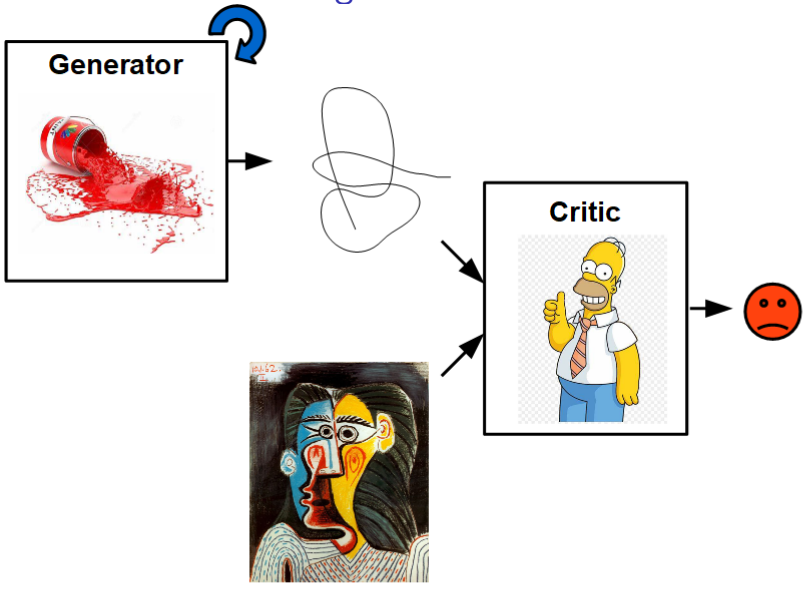


Illustration: GANs Learn Painting Like Picasso

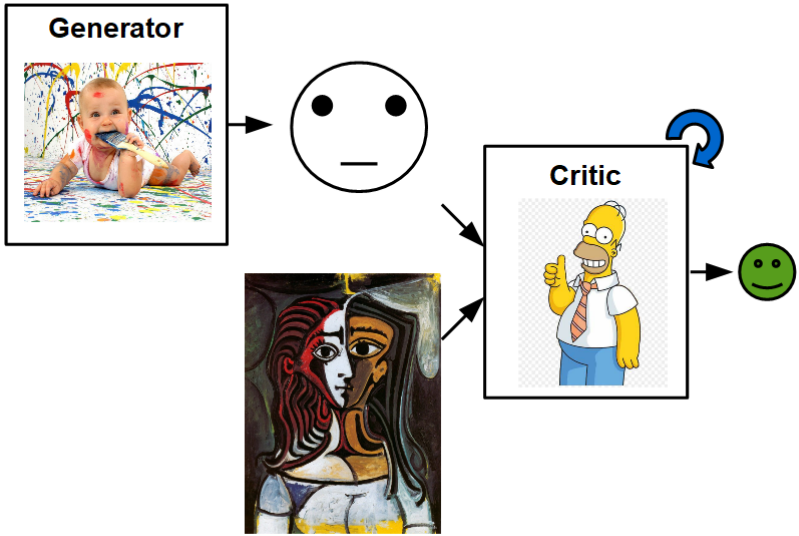


Illustration: GANs Learn Painting Like Picasso

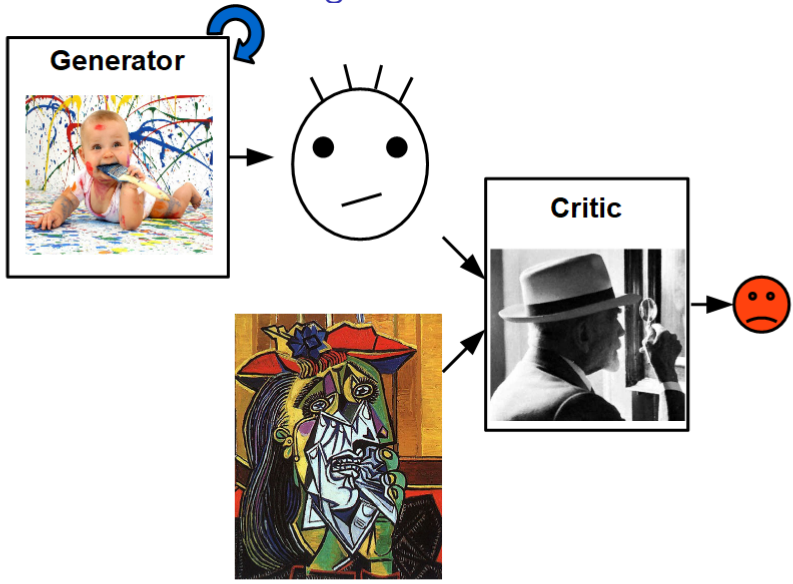


Illustration: GANs Learn Painting Like Picasso

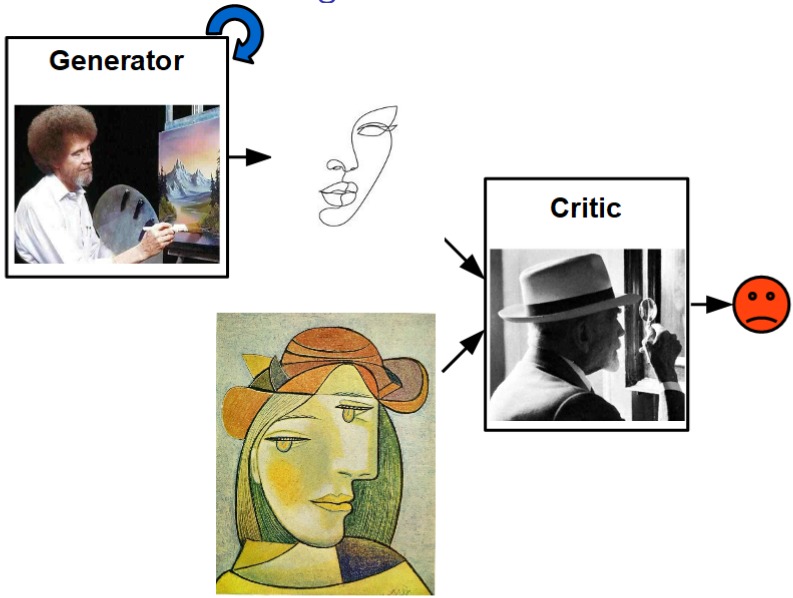
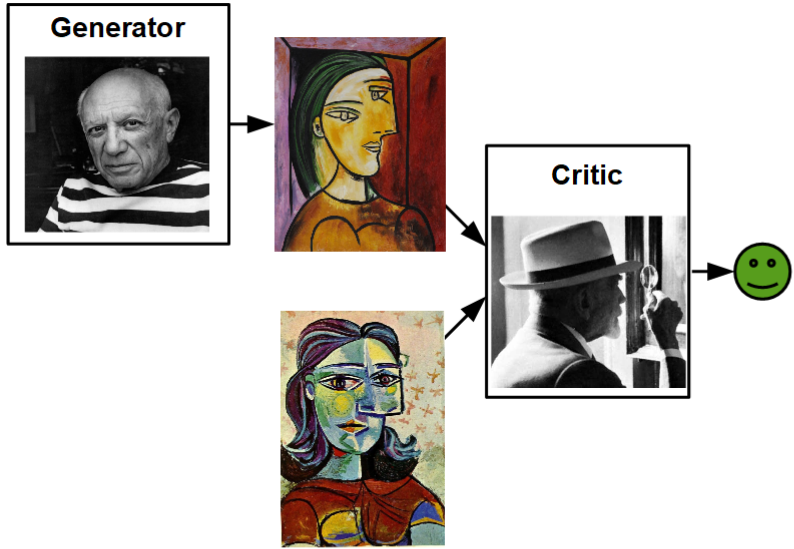


Illustration: GANs Learn Painting Like Picasso



Learning Algorithm: Train $\mathtt{G}_{\boldsymbol{\theta}}$ and $\mathtt{D}_{\boldsymbol{\phi}}$ Competitively

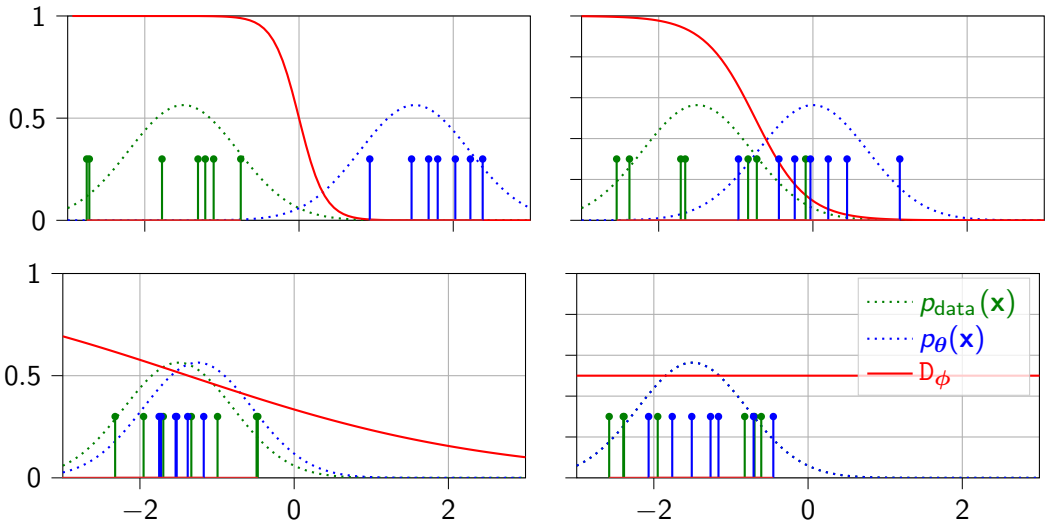
- **for** # training steps **do**
  - Discriminator update
    - Sample minibatch $\mathcal{M}_{\mathcal{Z}}$ of noise samples $\mathbf{z}_n \sim p(\mathbf{z})$
    - Sample minibatch $\mathcal{M}_{\mathcal{X}}$ of true data samples $\mathbf{x}_n \sim p_{\mathcal{X}}(\mathbf{x})$
    - Gradient ascent w.r.t. $\boldsymbol{\phi}$ (calculate forward pass for both $\mathtt{G}_{\boldsymbol{\theta}}$ and $D_{\boldsymbol{\phi}}$, but grads only for $\mathtt{D}_{\boldsymbol{\phi}}$)

$$\nabla_{\boldsymbol{\phi}} \frac{1}{N} \sum_{n=1}^{N} \left[\log \mathtt{D}_{\boldsymbol{\phi}}(\mathbf{x}_n) + \log \left(1 - \mathtt{D}_{\boldsymbol{\phi}} \left(\mathtt{G}_{\boldsymbol{\theta}}(\mathbf{z}_n) \right) \right) \right]$$

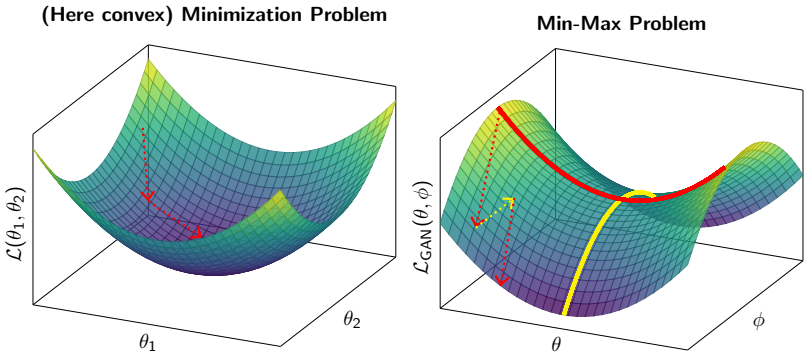
  - Generator update
    - Sample minibatch $\mathcal{M}_{\mathcal{Z}}$ of noise samples $\mathbf{z}_n \sim p(\mathbf{z})$
    - Gradient descent w.r.t. $\boldsymbol{\theta}$ (calculate forward pass and grads for both $\mathtt{G}_{\boldsymbol{\theta}}$ and $\mathtt{D}_{\boldsymbol{\phi}}$, use grads of $\mathtt{D}_{\boldsymbol{\phi}}$ for adjusting $\mathtt{G}_{\boldsymbol{\theta}}$)

$$-\nabla_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \log \left(1 - \mathtt{D}_{\boldsymbol{\phi}} \left(\mathtt{G}_{\boldsymbol{\theta}}(\mathbf{z}_n) \right) \right)$$

1D Toy Example



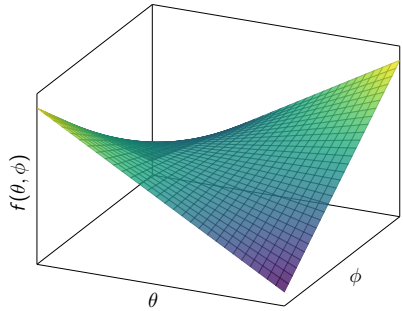
Finding Saddle Points

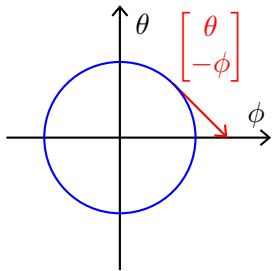


Problems for GAN optimization:

- Concurring optimization goals
- Danger of 'slipping off': Balancing min-max updates hard

Finding Saddle Points is Tough





- ('Simultaneous') gradient

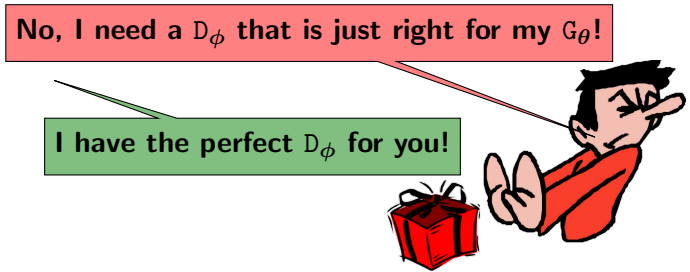
$$\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\ \mathcal{L}_{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathcal{X}) = \begin{bmatrix} \nabla_{\boldsymbol{\phi}}\ \mathcal{L}_{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathcal{X}) \\ -\nabla_{\boldsymbol{\theta}}\ \mathcal{L}_{\text{GAN}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathcal{X}) \end{bmatrix}$$

- Toy example

$$f(\theta, \phi) = \theta\phi, \qquad \nabla_{\theta,\phi}\ f(\theta, \phi) = \begin{bmatrix} \nabla_{\phi}\ f(\theta, \phi) \\ -\nabla_{\theta}\ f(\theta, \phi) \end{bmatrix} = \begin{bmatrix} \theta \\ -\phi \end{bmatrix}$$

Infinitesimal LR: Orbiting, Larger LR: Diverging

The Discriminator Dilemma



- Recall: G_θ is trained 'through' D_ϕ, 1D example:

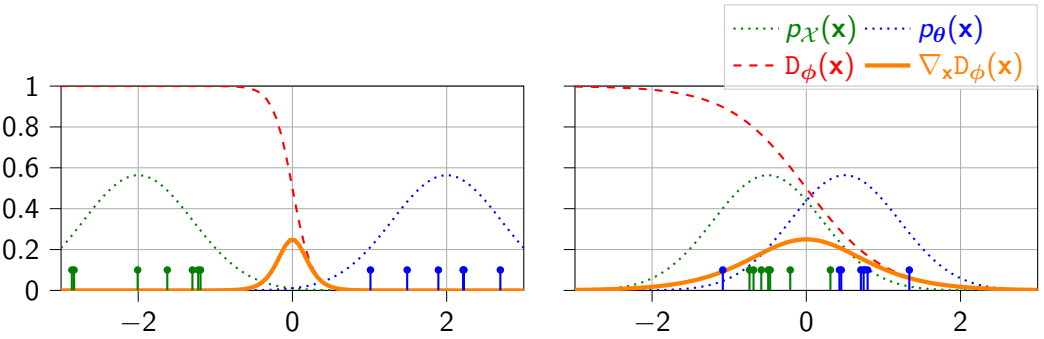
$$\frac{d}{d\theta} D_\phi\left(G_\theta(z)\right) = \frac{d}{dx} D_\phi(x) \frac{d}{d\theta} G_\theta(z)$$

- If D_ϕ **too strong**, ∇_θ vanishes $\Rightarrow$ G_θ doesn't train
- If D_ϕ **too weak**: G_θ does not get 'taught'

$D_\phi(\mathbf{x})$ has to fit to the training status of $G_\theta(\mathbf{z})$

Image: https://clipart-library.com/

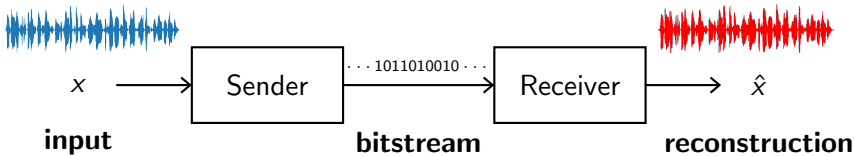
The Discriminator Dilemma: Illustration



- $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ and $\mathbf{x} \sim p_\theta(\mathbf{x})$ distinguished easily $\Rightarrow \mathrm{D}_\phi(\mathbf{x})$ strong
- $\|\nabla_{\mathbf{x}}\mathrm{D}_\phi(\mathbf{x})\| \approx 0$ almost everywhere
- G_θ does not train

- $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$ and $\mathbf{x} \sim p_\theta(\mathbf{x})$ hard to distinguish $\Rightarrow \mathrm{D}_\phi(\mathbf{x})$ weaker
- $\|\nabla_{\mathbf{x}}\mathrm{D}_\phi(\mathbf{x})\| > 0$
- G_θ trains

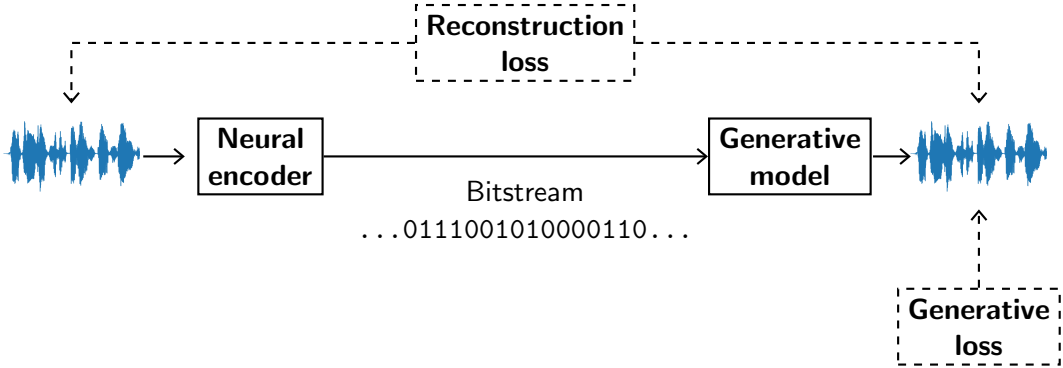
Audio Signal Compression



Compression: Find discrete signal representation by maximizing quality and minimizing bitrate

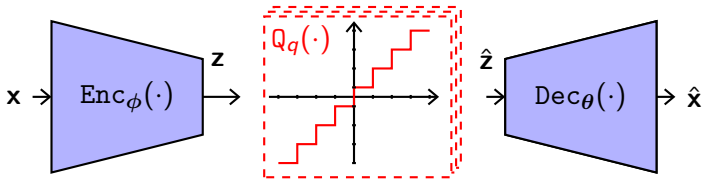
- **Redundancy**: Source properties known to sender and receiver result in redundant information that need not be transmitted
- **Irrelevancy**: Human listeners do not perceive all deviations of the received signal from the original

Neural Speech Coding



Bitstream $\mapsto$ speech waveform

Quantization in Neural Audio Codecs



- Neural audio codec = Neural encoder + quantizer + neural decoder
- Want to learn discrete representation $\hat{\mathbf{z}}$ end-to-end
- $\mathtt{Q}_q(\cdot)$ **not differentiable!** $\rightarrow$ Solutions: Soft approximations, special updates, ...
- Bitrate = sample rate $\times$ bit depth $\times$ channels
  e.g., CD (44.1 kHz, 16 bits, 2 channels): $44100 \times 16 \times 2 = 1411.2$ kbps

Rate-Distortion-Naturalness Trade Off



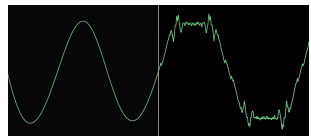
(how it sounds)

Naturalness

bitrate ↓ ⇔ naturalness ↓

distortion ↓ ⇔ naturalness ↓





Bitrate

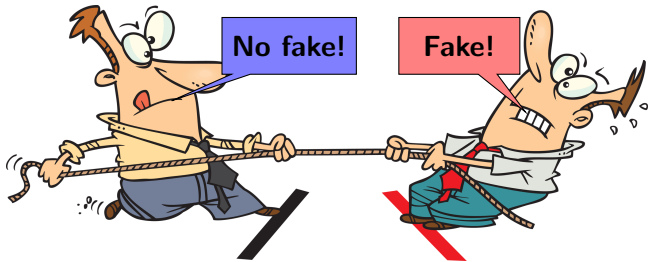
(file size, transmission bandwidth)

bitrate ↓ ⇔ distortion ↑

Distortion

(signal fidelity, accuracy)

GANs: Learn the Loss

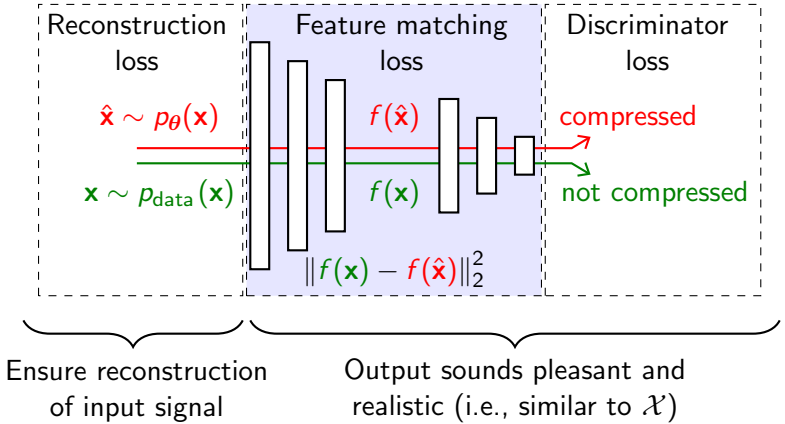


Generator Dec_θ	bad at generating natural audio	improving	generates almost perfect audio
Discriminator D_ν	bad at picking real audio	improving	picks on finest differences

training

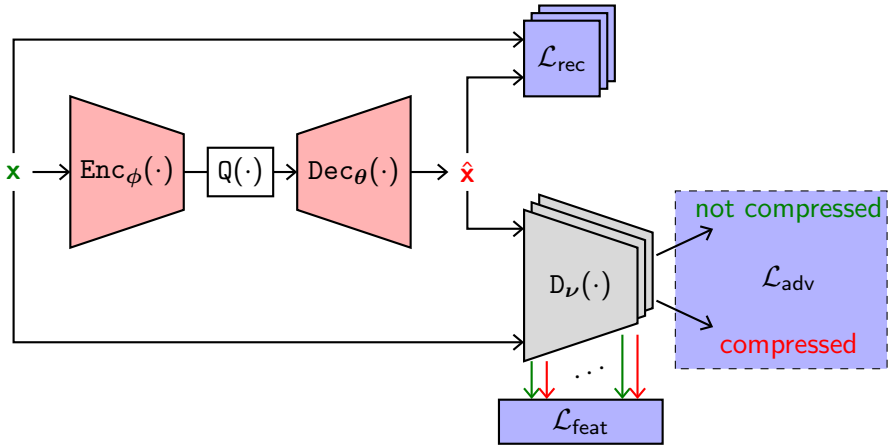
beginning — as training goes — end

Discriminator Realization



- Avoid overfitting of $\texttt{Dec}_\theta$ to $\texttt{D}_\nu$ with feature matching losses
- Discriminators at different resolutions (e.g., downsampling) to make decisions based on different aspects

Neural Audio Codec: Losses



$$\mathcal{L} = \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{feat}}\mathcal{L}_{\text{feat}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{quant}}\mathcal{L}_{\text{quant}}$$

Pros/Cons

Pros:

- Design of $\mathtt{G}_{\theta}$ has **almost no restrictions**
- **Good sample quality**
- Computationally **efficient** data generation possible

Cons:

- **Difficult training**: Long, unstable, hard to interpret
- Danger of **mode collapse**
- Need to design discriminator $\mathtt{D}_{\phi}$

Training DGMs: Key Questions answered for GANs

Data set $\mathcal{X}$: Continuous data

Model $p_{\boldsymbol{\theta}}(\mathbf{x})$: Implicit, sample-based representation

Loss $\mathcal{L}_{\text{GAN}}(\mathcal{X}, \boldsymbol{\theta}, \boldsymbol{\phi})$: Generator loss learned by discriminator

Optimization: Min-max game between $\mathtt{G}_{\boldsymbol{\theta}}$ and $\mathtt{D}_{\boldsymbol{\phi}}$

Sampling: $\mathtt{G}_{\boldsymbol{\theta}}$ forward pass, $\hat{\mathbf{x}} \sim p_{\boldsymbol{\theta}}(\mathbf{x})$: $\hat{\mathbf{z}} \sim p(\mathbf{z}) \rightarrow \hat{\mathbf{x}} \sim p_{\boldsymbol{\theta}}(\mathbf{x}|\hat{\mathbf{z}})$

Inference $p_{\boldsymbol{\theta}}(\mathbf{x}) = ?$: Possible, but difficult and not insightful

Summary

- GANs are likelihood-free $\rightarrow$ **Compare samples**
- **Min-Max Game**: `D` tries to identify fakes, `G` tries to fool `D`
- **Saddle points** result from min-max game & **are hard** to find
- **Balancing `D` and `G`:**
  `D` should not be too strong or too weak for `G`
- Neural audio coding as application for GANs