CS-461

Foundation Models and Generative AI

Generative Models I:
Autoregressive, Adversarial, and Autoencoder

Charlotte Bunne, Fall Semester 2025/26

Announcements

- **We will re-record last week's lecture**!
  In the recording I'll spend more time on Part 1 (Self-Supervised Learning) and mathematical background.
- **The research papers are optional, not required reading.**
  Everything you need to know for the course is covered in the slides and exercise sheets.
- Regarding the exam: The exercises and notebooks should give you an idea on the nature of the exam!

We will have a **guest lecture** next week!

Change of time and room!

5 pm in **CM 1 3**

Karsten Kreis
NVIDIA

Ruiqi Gao
DeepMind

Diffusion-based Generative Modeling: Foundations and Applications

Components of CS-461

1. Lecture

2. Exercise Sheets

3. Code Notebooks

Exam
70%

5. Papers
(optional)

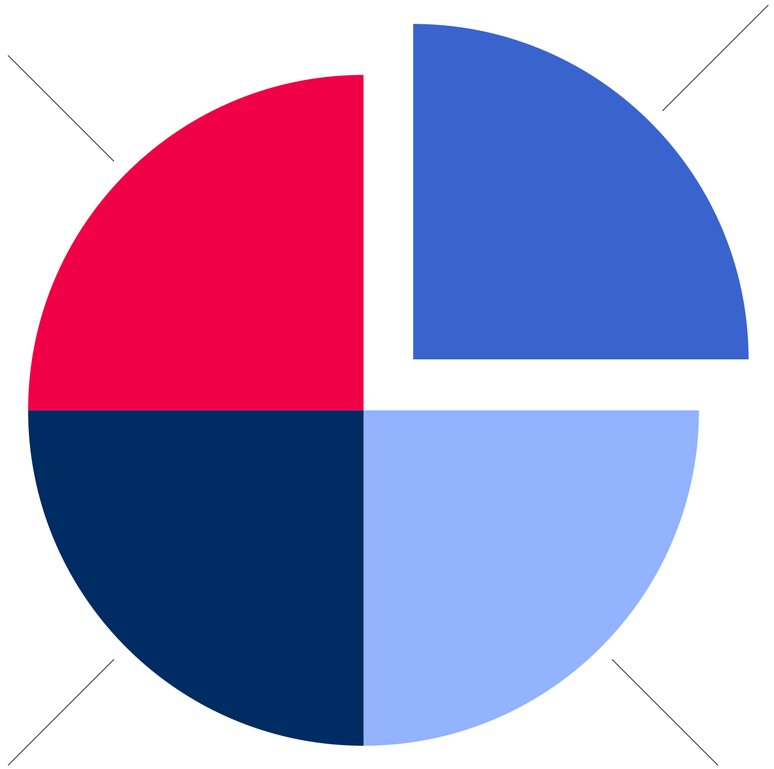
4. Code Assignments

Assigments
30% (2×15%)

Code Assignments

`gnoto`: EPFL's JupyterLab

Last Lecture: Self-Supervised Learning



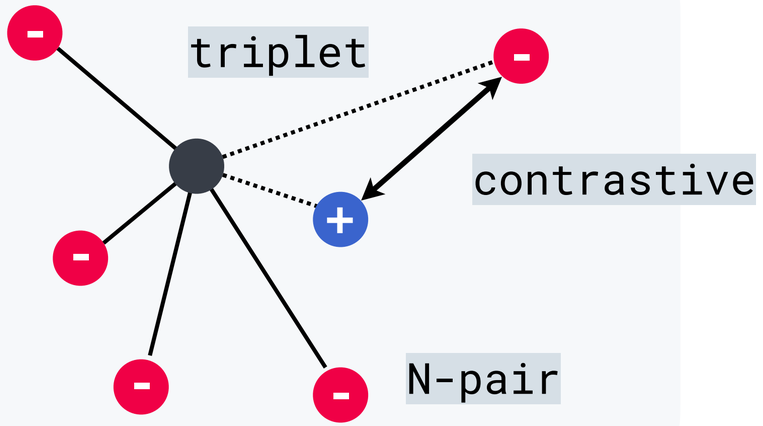
Learning

… is the **catalyst that powers foundation models**
(until we discover the next one)

Code Notebook 2 · Task 1 - 3

Self-Supervised Learning Concepts

Contrastive



Frames `N-pair` loss as mutual information maximization between positive pairs. `infoNCE`

Masking

$$\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{M}\left[\frac{1}{|M|}\sum_{i\in M}\ell\left(x_i,\hat{x}_i\right)\right]$$

cross-entropy

MSE

Masking methods approximate pseudo-likelihood optimization.

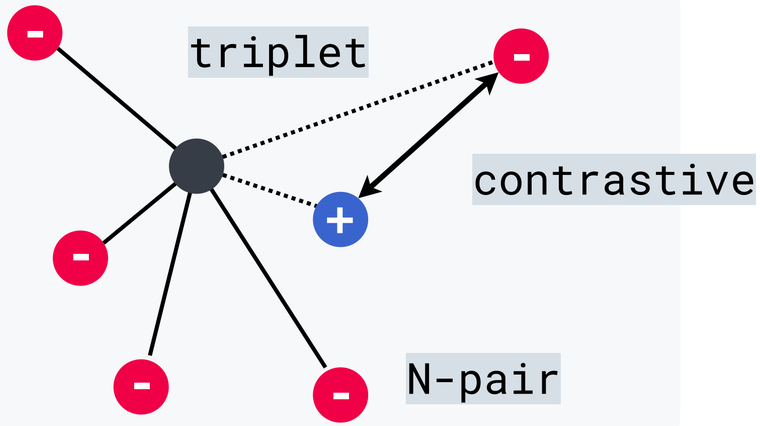
Autoregressive

$$-\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{t=1}^{T}\log p_\theta\left(x_t \mid x_{<t}\right)\right]$$

Autoregressive models optimize the exact likelihood.

Self-Supervised Learning Concepts

Contrastive



Maximizes lower bound on $\log p(x \mid y)$, implicitly learning log-likelihood ratios.

Masking

$$\mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{M}\left[\frac{1}{|M|}\sum_{i\in M}\ell\left(x_i,\hat{x}_i\right)\right]$$

cross-entropy MSE

Masking methods approximate pseudo-likelihood optimization.

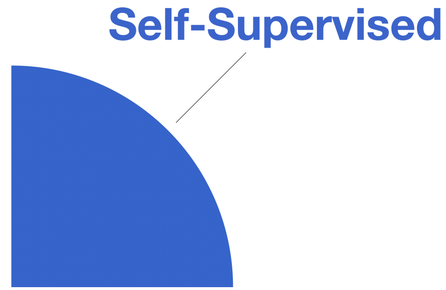
Autoregressive

$$-\mathbb{E}_{x\sim\mathcal{D}}\left[\sum_{t=1}^{T}\log p_\theta\left(x_t \mid x_{<t}\right)\right]$$

Autoregressive models optimize the exact likelihood.

Exercise 1 · Task 1

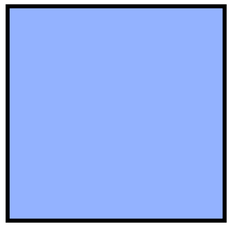
From Learning Principle to Architecture



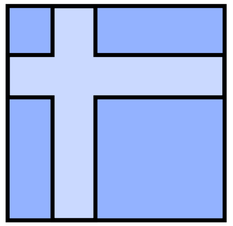
Important!

The **self-supervision objective** is *not* a training detail but the fundamental constraint that **shapes many architectural choices** from attention patterns to encoder-decoder asymmetry.

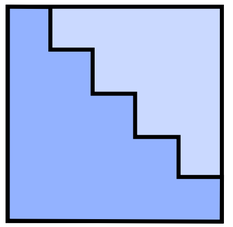
1. **Contrastive** → Must compare → Needs batch/memory
   - Requires denominator with many negatives: $\exp(\text{sim}^+)/\sum \exp(\text{sim})$
   - **Solution**: Large batches (SimCLR: 4096) or memory banks (MoCo: 65K queue)
2. **Masked** → Must see all → Needs bidirectional
   - Predict from full context: $p\left(x_{\text{mask}} \mid x_{\text{unmasked}}\right)$
   - **Solution**: Bidirectional masked attention patterns
3. **Autoregressive** → Must generate → Needs causality
   - Factorization chain: $p(x) = \prod p\left(x_t \mid x_{<t}\right)$
   - **Solution**: Causal self-attention to avoid future leakage



full attention



bidirectional masked attention



causal attention

From Learning Principle to Architecture

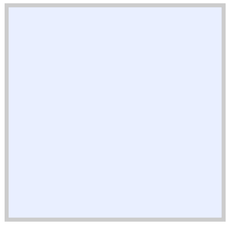
Important!

The **self-supervision objective** is *not* a training detail but the fundamental constraint that **shapes many architectural choices** from attention patterns to encoder-decoder asymmetry.

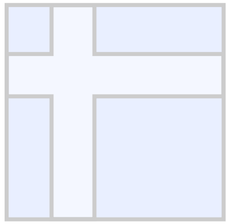
attention?

Lecture 5:
Tokenization and Building Blocks

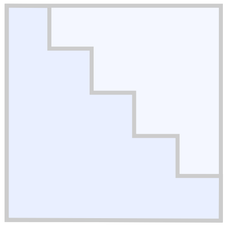
1. **Contrastive** → Must compare → Needs batch/memory
   - Requires denominator with ma
   - **Solution**: Large batches (SimC eue)
2. **Masked** → Must see all → Needs bidirectional
   - Predict from full context: $p\left(x_{\text{mask}} \mid x_{\text{unmasked}}\right)$
   - **Solution**: Full attention patterns
3. **Autoregressive** → Must generate → Needs causality
   - Factorization chain: $p(x) = \prod p\left(x_t \mid x_{<t}\right)$
   - **Solution**: Causal self-attention to avoid future leakage



full attention

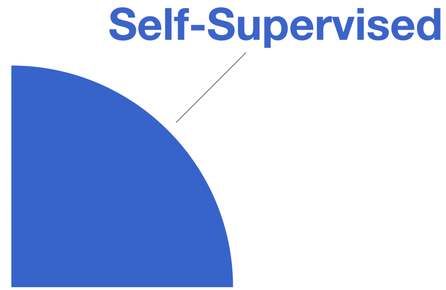


bidirectional masked attention



causal attention

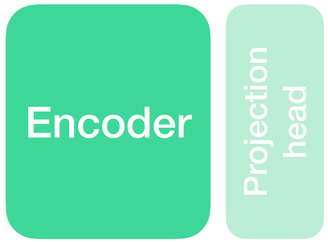
From Learning Principle to Architecture



Important!

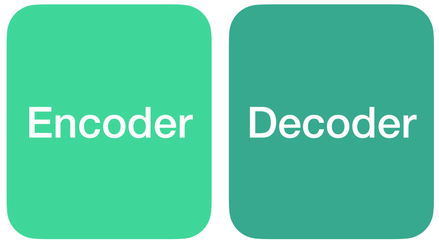
The **self-supervision objective** is *not* a training detail but the fundamental constraint that **shapes many architectural choices** from attention patterns to encoder-decoder asymmetry.

1. **Contrastive**



Encoder + projection head
for comparing global features

2. **Masked**



Encoder-decoder split that
can be asymmetric

3. **Autoregressive**

Unified decoder-only architecture
with causal masking

Lecture 5-7: Architectures

From Learning Principle to Architecture

Important!

The **self-supervision objective** is *not* a training detail but the fundamental constraint that **shapes many architectural choices** from attention patterns to encoder-decoder asymmetry.

Encoder? Decoder?

1. **Contrastive**

Encode

Encoder + projection head
obal features

Lecture 5-9: Architectures

2. **Masked**

Encoder | Decoder

Encoder-decoder split that can be asymmetric

3. **Autoregressive**

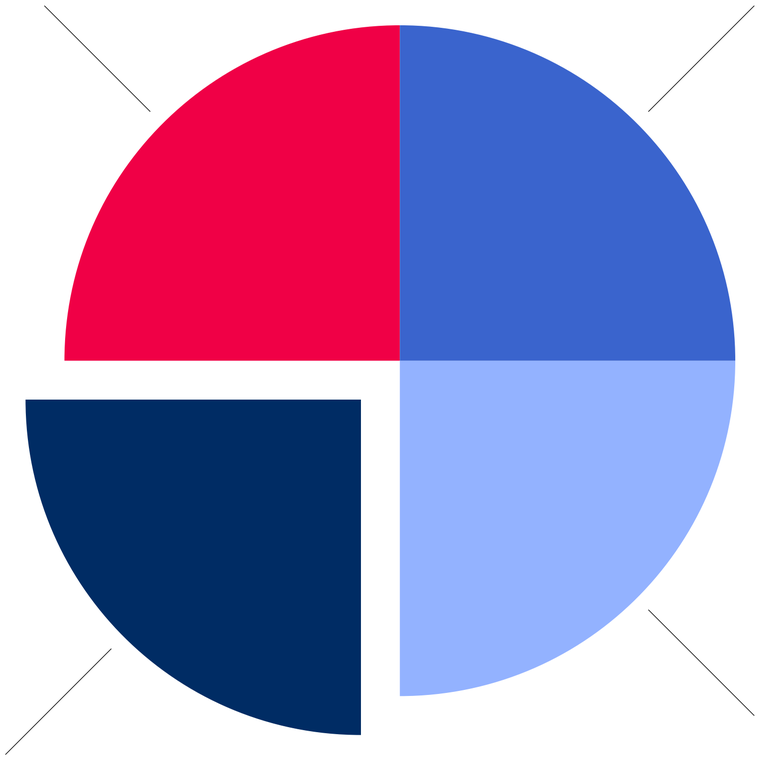
Decoder

Unified decoder-only architecture with causal masking

Lecture 5-7: Architectures

This Lecture: Unsupervised Learning

… and next …



More specifically:

Generative AI **Learning**

Unsupervised Learning

Goal: Discover structure in unlabeled data x.

Generative Modeling

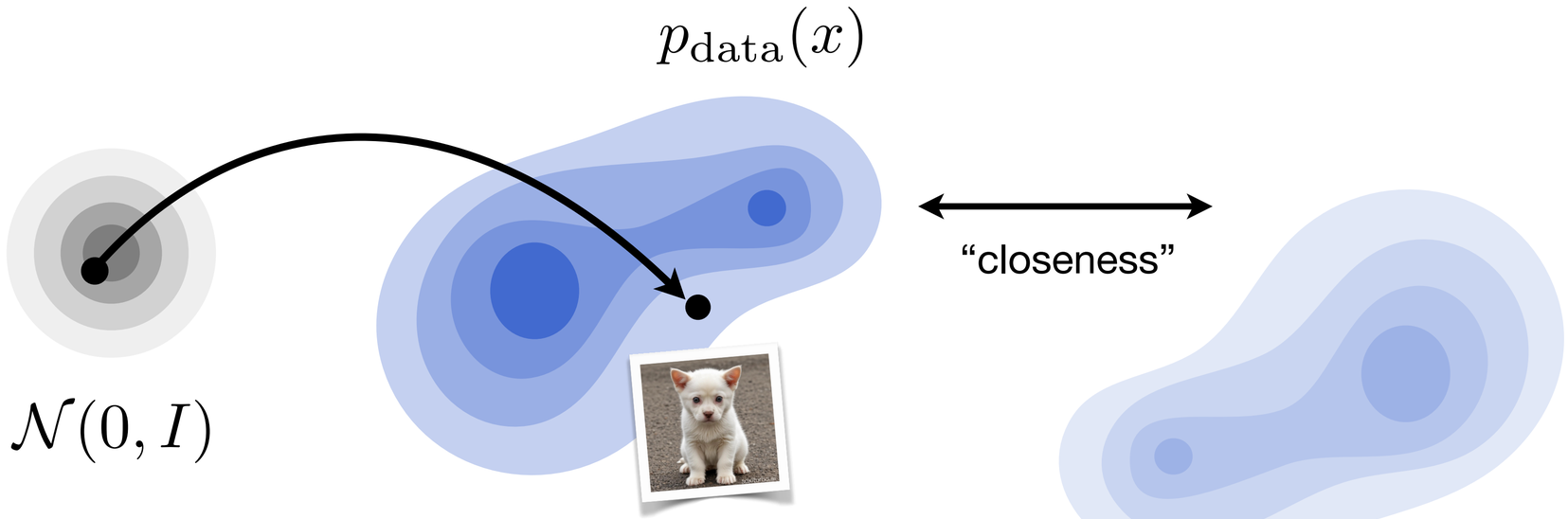
discovers structure in unlabeled data
because it learns to **model the underlying data distribution** $p(x)$ without requiring labels.

Key insight: By learning to generate data, models must discover and encode the rules, patterns, and relationships that govern that data which *is* the structure.

Generate data?

Learning a Generative Model

Generative models learn to capture the probability distribution $p(x)$ and generate new samples from that distribution.



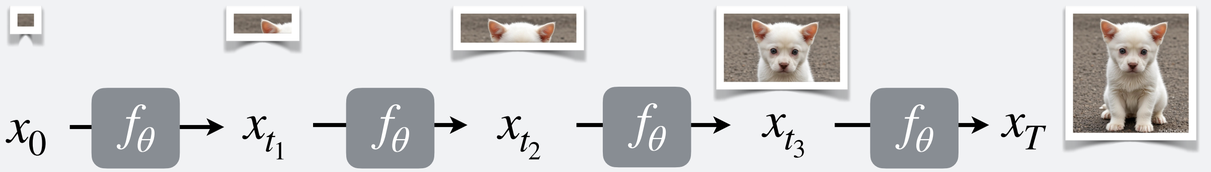
Unlike discriminative models that learn $p(y \mid x)$, generative models learn $p(x)$ or $p(x, y)$.

Four Philosophical Approaches

How do we do this?

1. Autoregressive Models

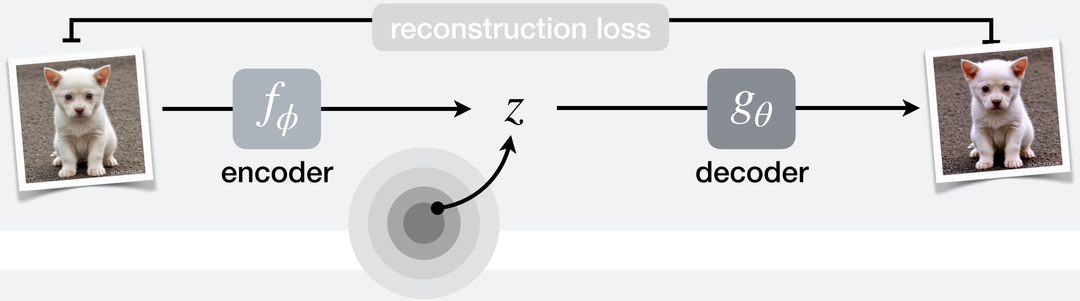
"One step at a time."



2. Autoencoders

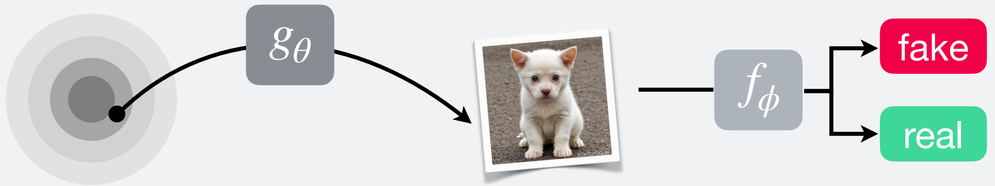
"Reduce to the essence and rebuild."

Variational Autoencoder



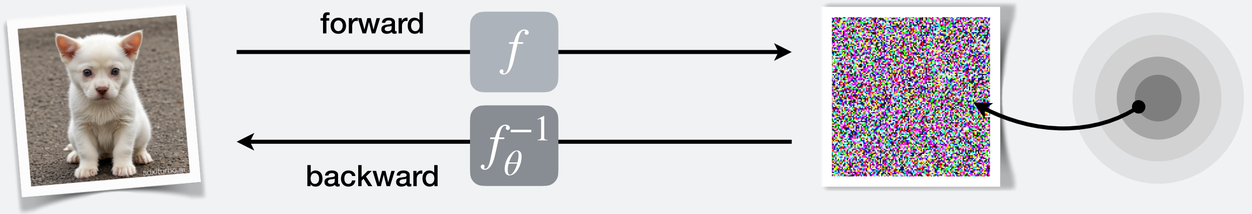
3. Generative Adversarial Models

"Fake it till you make it."



4. Diffusion Models

"Mess it up, then `Ctrl+Z`."



⚠️ **Disclaimer:** Generative Models in Context

We could easily spend an entire course on generative models.

We'll focus on what you need to know to **understand and build modern foundation models**:

- Fundamental paradigms that underpin different approaches of generative models approaches.
- Key trade-offs between quality, speed, and controllability.
- Why different architectures suit different scales.
- How these concepts combine into different architectures.

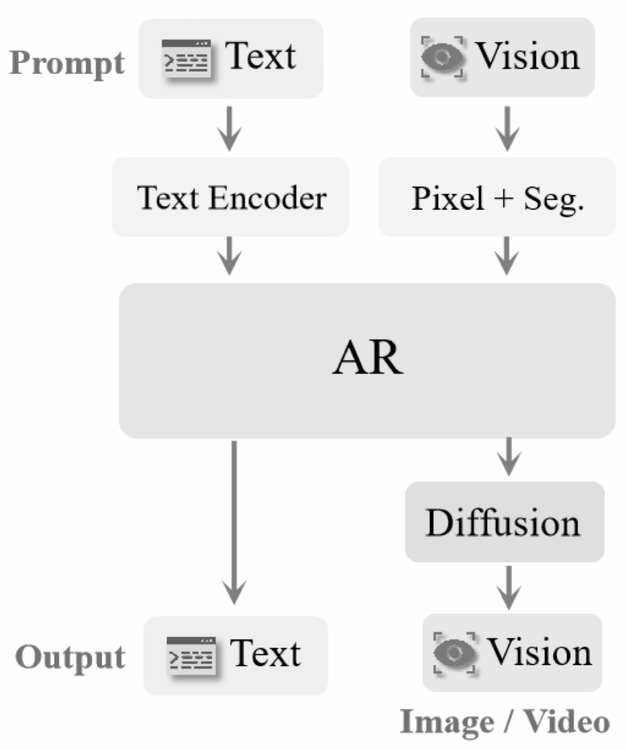
💡 **Our Goal**:

1. Master the complete toolkit of building blocks and design principles that architects use when creating foundation models.
2. Give you the vocabulary and concepts to understand most generative architectures.

Take Image Generation in GPT–4o

Modern generative AI systems rarely rely on a single paradigm!

Instead, they *combine* insights from VAEs, diffusion models, and autoregressive approaches and more to achieve state-of-the-art performance.



Note: *The architecture of GPT-4o has not been released, so we can only speculate based on trends and capabilities.*

2021 **DALL·E 1:** Autoregressive over VQ-VAE image tokens.

2022 **DALL·E 2:** Diffusion in latent space, CLIP-guided.

2023 **DALL·E 3:** Improved diffusion, tighter GPT prompt integration.

2025 **GPT-4o:** Back to autoregressive, i.e., sequential image generation, no longer diffusion.

Course Schedule

Week	Part I	Week	
1	Introduction and Overview	8	Multi-Moda
2	**Learning at Scale**: Supervised, Self-Supervised, and Beyond	9	**Architecture** Foundation M
3	**Generative Models I**: Autoregressive, Adversarial, and Autoencoder	10	In-Context Lea
4	**Generative Models II**: Diffusion Models and Beyond	11	Adaptation, Fine
5	**Generative Models III**: Recap on Generative Models and Generalizations Tokenization Across Modalities and Building Blocks	12	World Models and
6	**Architectures I**: Language and Vision Foundation Models	13	**Architectures III**: FI Foundation Models, R Reasoning, and Decis
7	*Semester Break*	14	Foundation Models and Agentic Systems
		15	Outlook and Summary


BUNNE LAB

On Building and Designing Foundation Models

September 2025

From association to reasoning: foundation models must learn to ask "what if." An essay on their design and building blocks.

Foundation models are not accidents of scale. They emerge from a chain of decisions: data shape objectives, objectives shape architectures, and architectures determine what can be scaled. To design them well is to recognize how these components interact and constrain one another. While transformers have proven remarkably general across domains, their success does not come from blindly repeating a formula. It comes from adapting learning objectives, architectures, and scaling strategies to the structure of the data at hand. At the same time, scale can sometimes substitute for carefully chosen inductive bias, raising a central question: when is tailoring essential, and when does brute force suffice?

A Design Flow

The process can be thought of as a cascade:

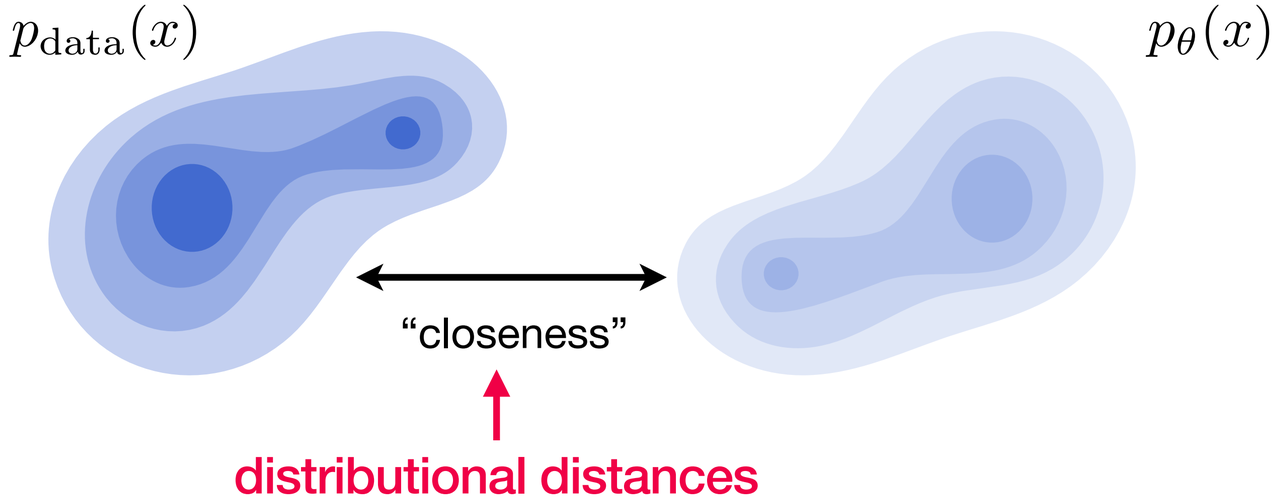
Data + Questions → Learning Framework → Architecture → Scale → Engineering

1. **Data and questions.** The starting point is not a model but the patterns in the data and the questions to be answered. Sequential data may call for causal modeling; long-range dependencies may demand bidirectional context; spatial data has no inherent order but strong locality; sparse, high-dimensional measurements may resist sequence modeling altogether. Multimodal settings require learning alignments and shared structure across disparate inputs.

Generative Modeling as Density Estimation

Learn p_θ to construct the "best" approximation of the underlying distribution p_{data} given samples of dataset $\mathscr{D}$.

We want to find p_θ that is as close as possible to the data distribution p_{data}.



Kullback-Leibler Divergence

Jensen-Shannon Divergence

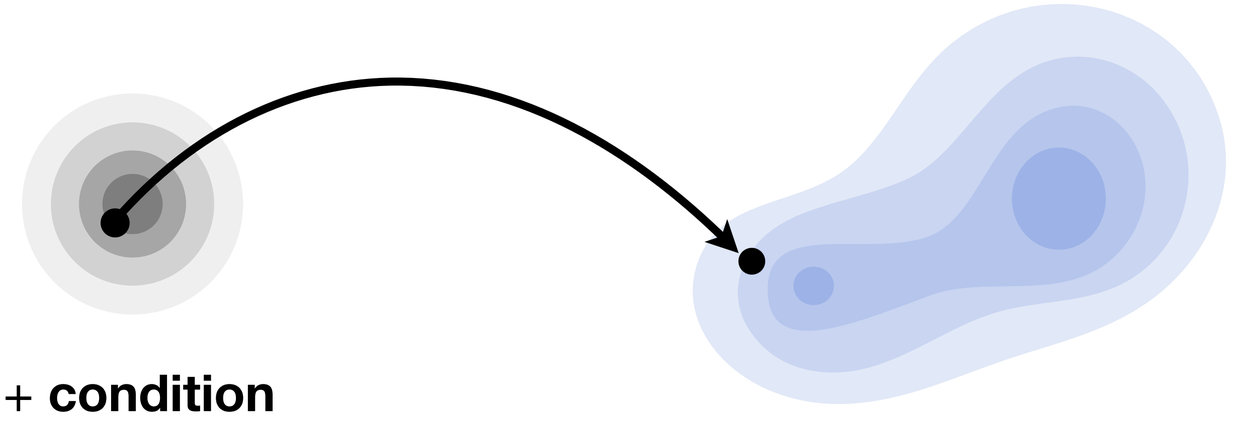
Wasserstein Distance

e.g., autoregressive models and variational autoencoders

e.g., generative adversarial networks

Conditional Generative Models

Core Concept: Control generation with additional information c.



+ **condition**

$p_\theta(x \mid c)$

instead of $p_\theta(x)$

e.g., text promt

image

class

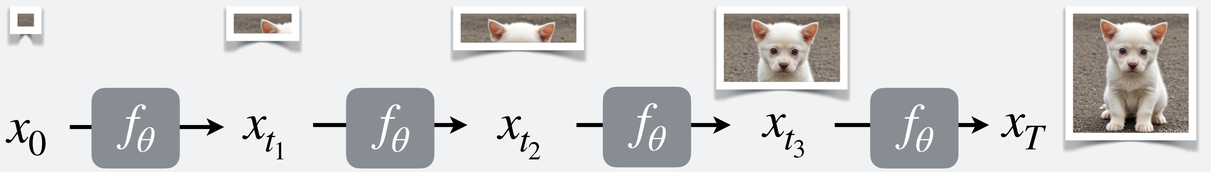
How?

- via Concatenation: $[x, c] \rightarrow$ Model$_\theta$
- via Cross-Attention: x attends to c
- via Classifier Guidance: $\nabla \log p(c \mid x)$
- etc.

Four Philosophical Approaches

1. Autoregressive Models

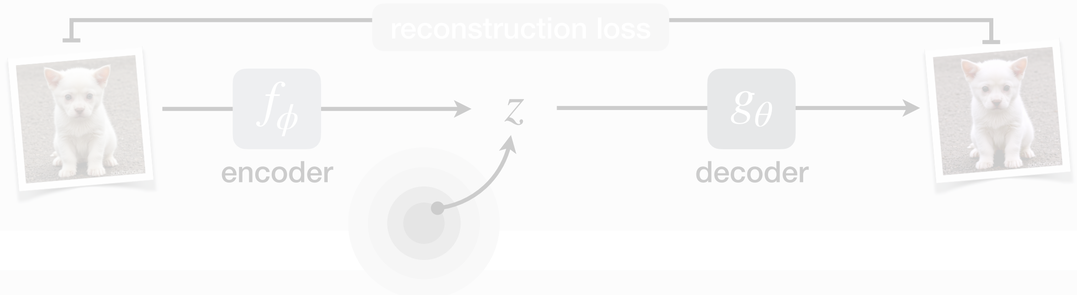
"One step at a time."



2. Autoencoders

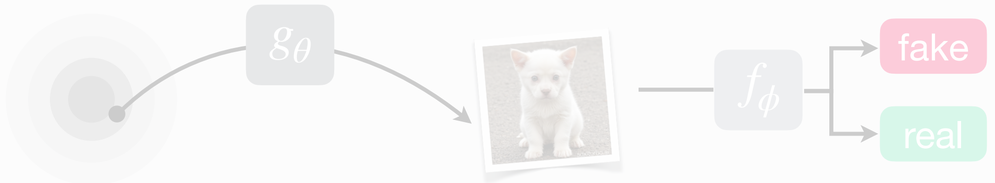
"Reduce to the essence and rebuild."

Variational Autoencoder



3. Generative Adversarial Models

"Fake it till you make it."



4. Diffusion Models

"Mess it up, then `Ctrl+Z`."

Autoregressive Generative Models

Core Principle: At each step, predict the next element given all previous elements
→ sequential prediction.

Key Idea: Model the joint distribution by decomposing it into a product of conditionals using the **chain rule**.

Fully general, works for any distribution, any ordering, no assumptions!

$$p(x) = p(x_1, x_2, \ldots, x_t) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2)\ldots p(x_t|p_{<t})$$

→ reduce a hard problem, i.e., learning $p(x)$, into a sequence of easier problems.

Maximum Likelihood Training

$$\max_\theta \log p_\theta(x) = \max_\theta \log p_\theta\left(x_1, x_2, \ldots, x_t\right)$$
$$= \max_\theta \sum_{i=1}^{t} \log p_\theta\left(x_i \mid x_{<i}\right)$$

Each term is just a standard supervised learning loss!

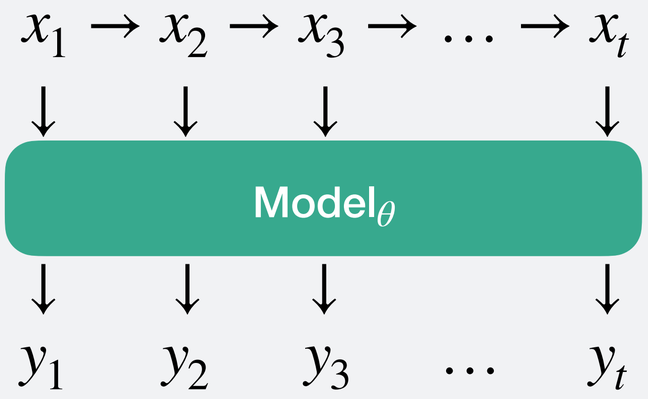
Autoregressive Generative Models: Parameterizations

Efficient!

STRATEGY 1

Shared Parameters
One Model for All Positions

Parameters
$\mathcal{O}(d^2)$

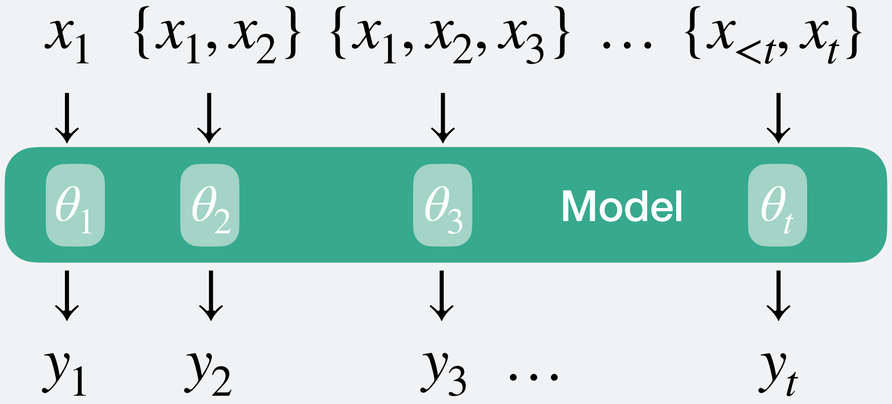


e.g., RNNs (Rumelhart et al., 1985)
GPT (Radford et al., 2018) etc.

STRATEGY 2

Factorized Parameters
Position-Specific Models

Parameters
$\mathcal{O}(t \times d^2)$



e.g., MADE (Germain et al., 2015)
NADE (Uria et al., 2016), etc.

Autoregressive Generative Models: Examples

Example from Language:
Generative Pretrained Transformer (GPT)

Radford et al., (2018)

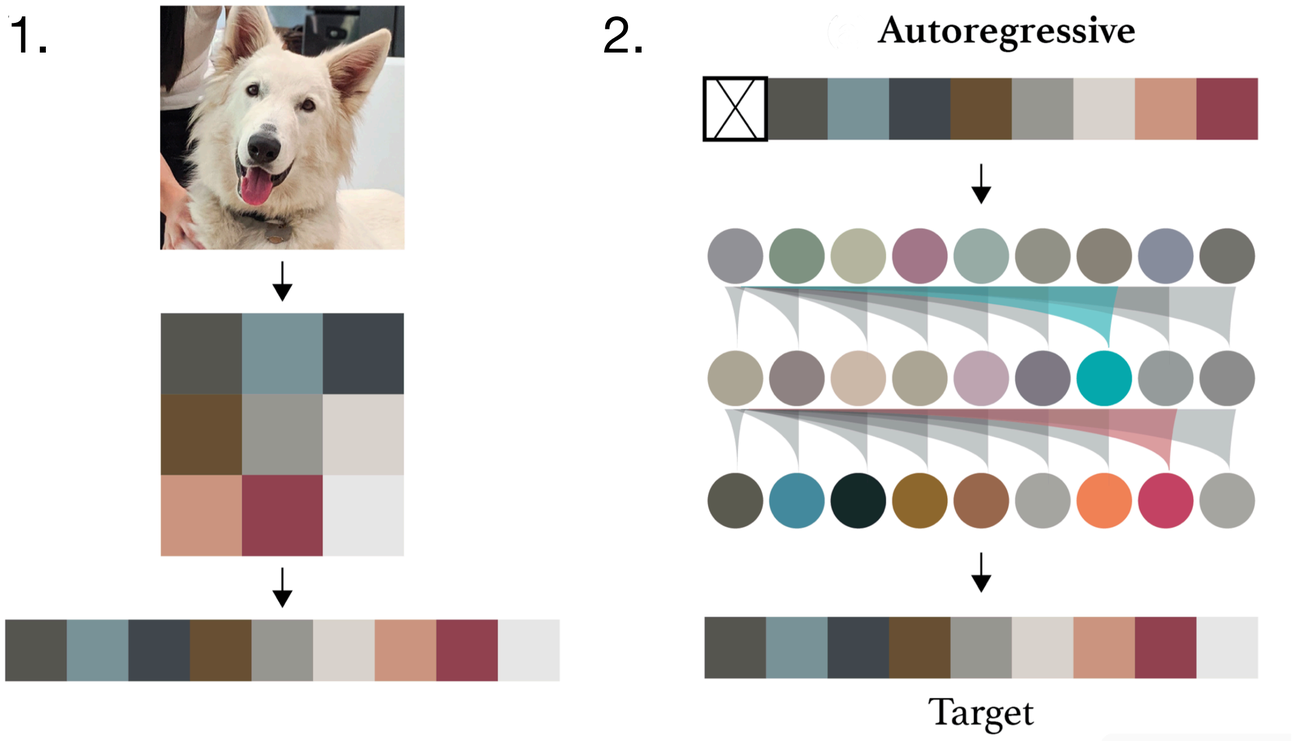
Lecture 6: Language Foundation Models

Example from Vision:
Image GPT (iGPT)

Chen et al., (2020)

1. Pre-process raw images by resizing to a low resolution and reshaping into a 1D sequence.

2. Train via auto-regressive next pixel prediction.



Four Philosophical Approaches

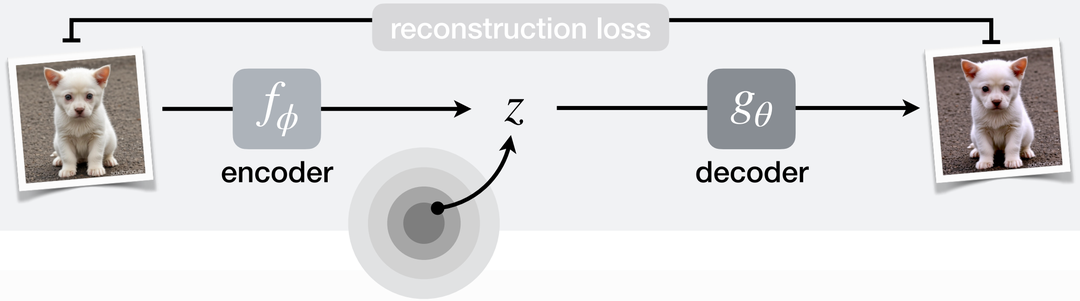
1. Autoregressive Models

"One step at a time."

2. Autoencoders

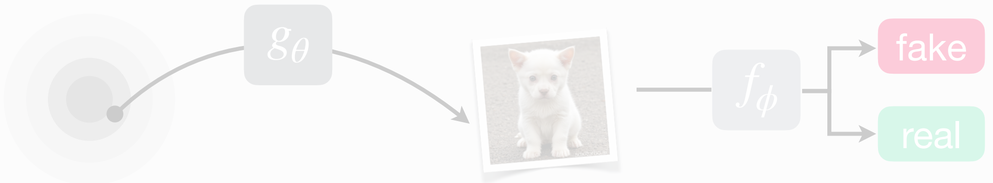
"Reduce to the essence and rebuild."

Variational Autoencoder



3. Generative Adversarial Models

"Fake it till you make it."



4. Diffusion Models

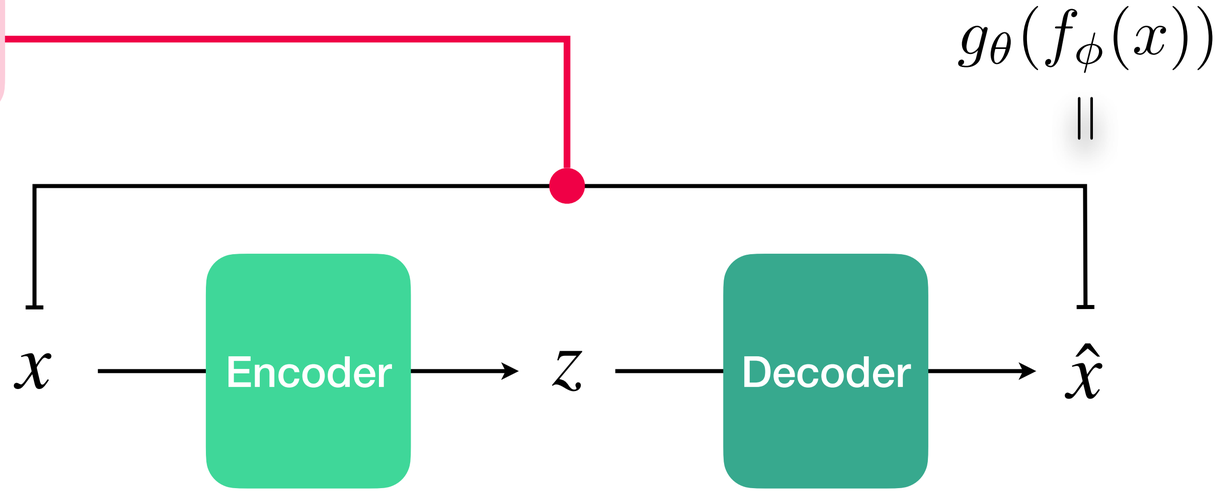
"Mess it up, then `Ctrl+Z`."

Autoencoders

Key idea: An autoencoder learns to compress data into a lower-dimensional representation and reconstruct it.

Reconstruction Loss

$$\mathcal{L}(\theta, \phi) = \|x - g_\theta(f_\phi(x))\|^2$$



Architecture:

Encoder: $f_\phi : \mathcal{X} \to \mathcal{Z}$

i.e., f_ϕ maps data to latent code.

Decoder: $g_\theta : \mathcal{Z} \to \mathcal{X}$

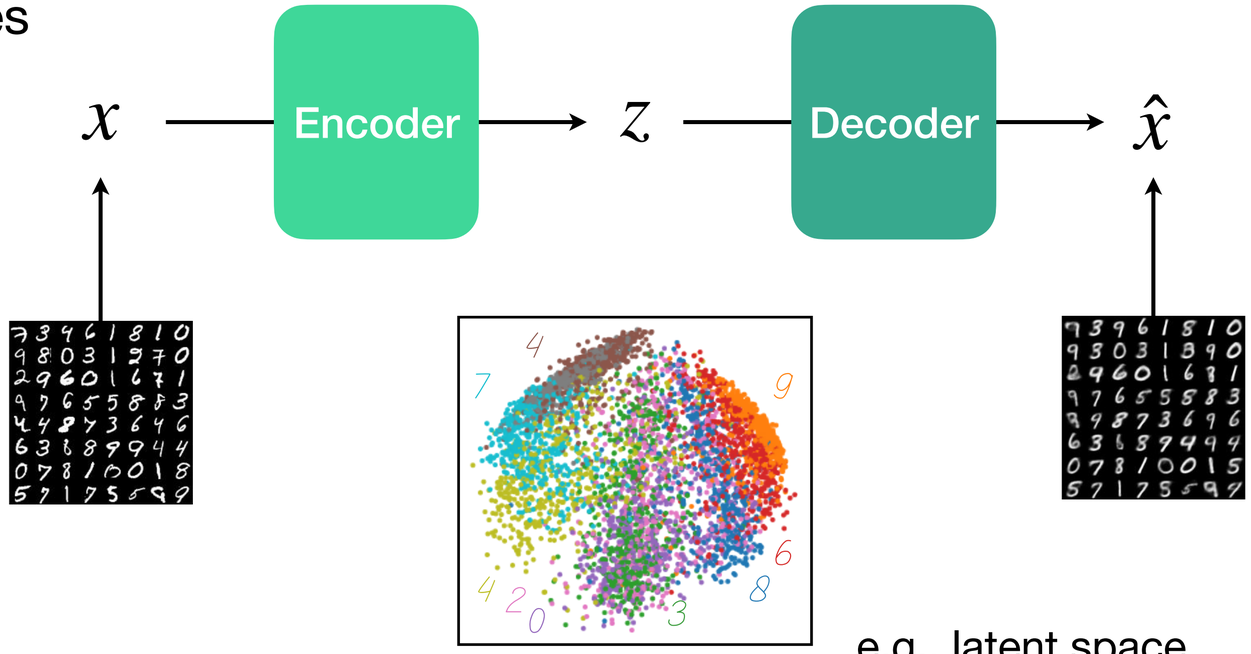
i.e., g_θ reconstructs from latent code.

Latent space: $\mathcal{Z}$

i.e., compressed representation of input space $\mathcal{X}$.

What is a Latent Space?

A lower-dimensional representation that captures the essential factors of variation in data.



e.g., latent space of a linear autoencoder

Properties:

- **Compression**, i.e., $\dim(\mathcal{Z}) \ll \dim(\mathcal{X})$ → dimensionality reduction!
- **Abstraction**: Latent codes capture semantic meaning → discover meaningful representations!
- **Manifold Hypothesis**: High-dimensional data lies on lower-dimensional manifold.

Variants of Autoencoders

1. **Denoising Autoencoders**

   Idea: Learn robust features by reconstructing clean data from corrupted input.

Lecture 6: Vision Foundation Models

2. **Masked Autoencoder (MAE)**

   Idea: Reconstruct randomly masked patches/tokens for efficient self-supervised pretraining.

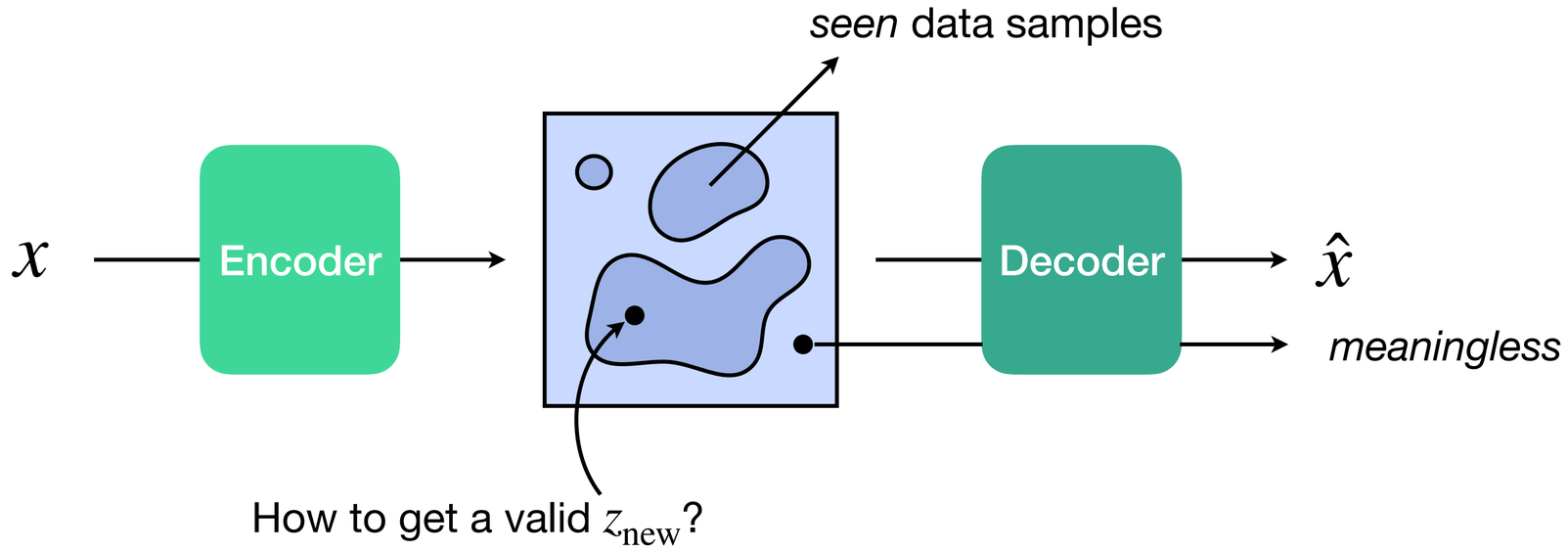
3. **Sparse Autoencoders**

   Idea: Encourage sparse latent representations for interpretability.

Autoencoders: The Generation Problem

Exercise 3 · Task 1

Standard autoencoders fail at generation because



1. **No probabilistic framework**: Everything is deterministic.
2. **Irregularities in latent space**, e.g., holes. Only encodes training points.
3. **No sampling mechanism**: What distribution to sample z from?

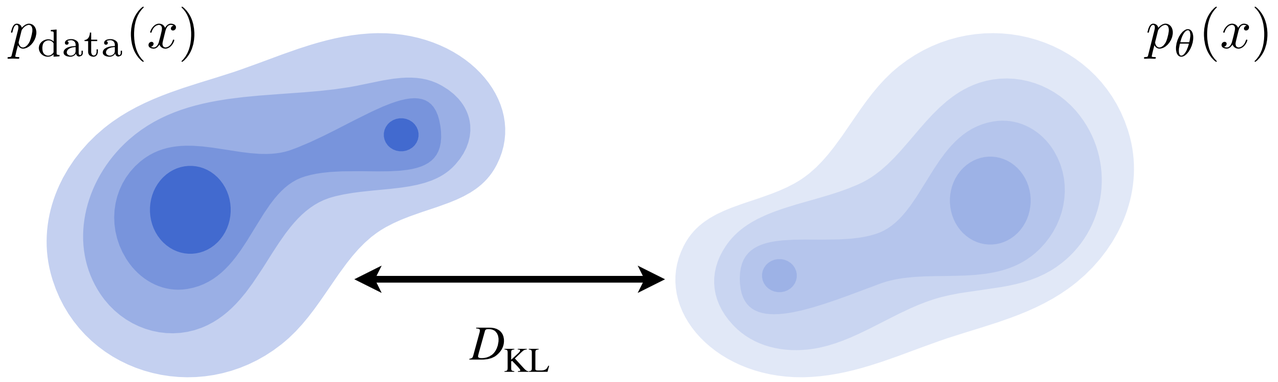
We need a principled probabilistic approach!

Excursus: **Maximum Likelihood Learning**

Before introducing probabilistic versions of autoencoders, we need to understand the theoretical foundation for learning distributions.

Goal: Given samples from unknown distribution p_{data}, learn model parameters θ such that $p_\theta \approx p_{\text{data}}$.

→ We need a **measure of distance** between p_θ and p_{data}!



Excursus: **Kullback-Leibler Divergence**

Definition: Measures how much one distribution differs from another.

Kullback-Leibler Divergence

$$D_{\mathrm{KL}}(p\|q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(x)}{q(x)}\right]$$

or KL Divergence

Properties:

Always non-negative: $D_{\mathrm{KL}}(p\|q) \geq 0$.

Zero if and only if $p = q$ almost everywhere.

Not symmetric: $D_{\mathrm{KL}}(p\|q) \neq D_{\mathrm{KL}}(q\|p)$.

Interpretation:

Information theory: Extra bits needed to encode samples from p using optimal code for q.

Statistical: Expected log ratio of probabilities.

Geometric: Not a true distance (not symmetric), but useful measure.

Excursus: **Maximum Likelihood Learning**

Goal: Given samples from unknown distribution p_{data}, learn model parameters θ such that $p_\theta \approx p_{\text{data}}$.

We want to minimize the distance between true data distribution and our model, i.e.,

$$\min_\theta D_{\text{KL}}\left(p_{\text{data}} \| p_\theta\right)$$

Expanding the KL divergence

$$D_{\text{KL}}\left(p_{\text{data}} \| p_\theta\right) = \mathbb{E}_{x \sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}\left(x\right)}{p_\theta(x)}\right]$$

$$= \mathbb{E}_{x \sim p_{\text{data}}}\left[\log p_{\text{data}}\left(x\right)\right] - \mathbb{E}_{x \sim p_{\text{data}}}\left[\log p_\theta(x)\right]$$

independent of θ

Excursus: **Maximum Likelihood Learning**

Goal: Given samples from unknown distribution p_{data}, learn model parameters θ such that $p_\theta \approx p_{\text{data}}$.

We want to minimize the distance between true data distribution and our model, i.e.,

$$\min_\theta D_{\text{KL}}\left(p_{\text{data}} \| p_\theta\right)$$

Key Insight: Minimizing KL divergence is equivalent to maximizing expected log-likelihood!

Maximum Likelihood Estimation (MLE) ←

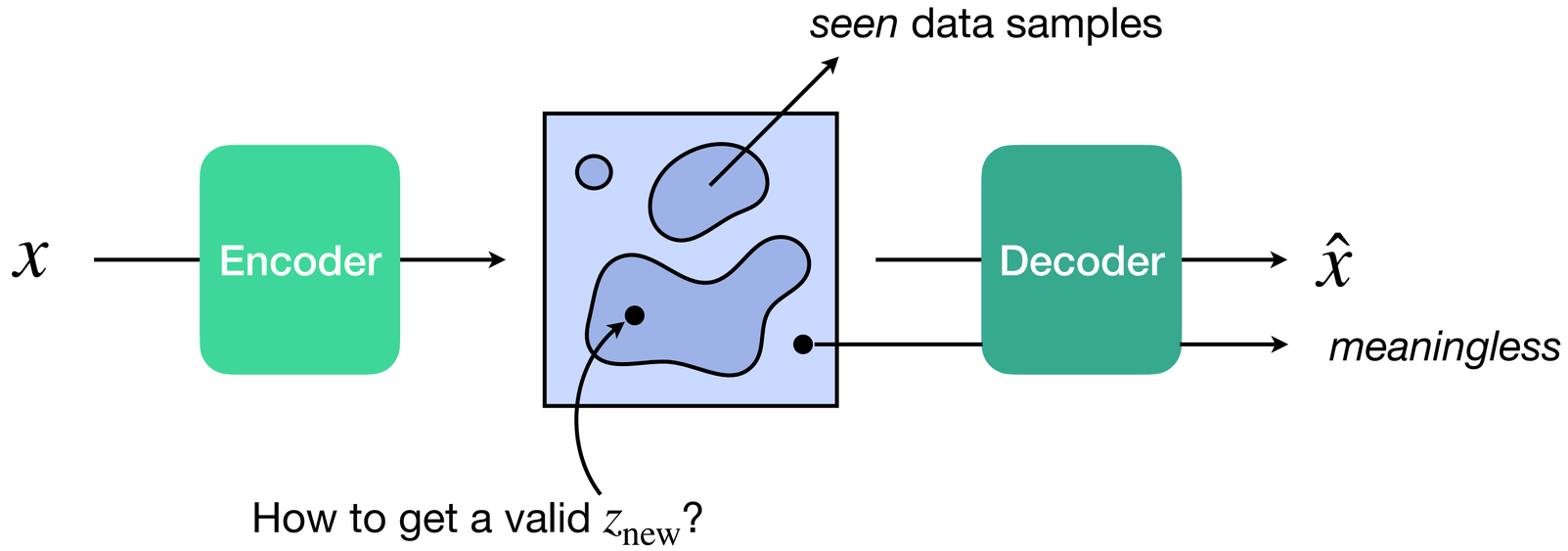
$$\arg\min_\theta D_{\text{KL}}\left(p_{\text{data}} \| p_\theta\right) = \arg\max_\theta \mathbb{E}_{x \sim p_{\text{data}}}\left[\log p_\theta(x)\right]$$

In practice, we approximate the expectation with our finite dataset.

$$\arg\max_\theta \frac{1}{N} \sum_{i=1}^{N} \log p_\theta\left(x_i\right)$$

Autoencoders: The Generation Problem

Standard autoencoders fail at generation because



1. **No probabilistic framework**: Everything is deterministic.
2. **Irregularities in latent space**, e.g., holes. Only encodes training points.
3. **No sampling mechanism**: What distribution to sample z from?

We need a principled probabilistic approach!

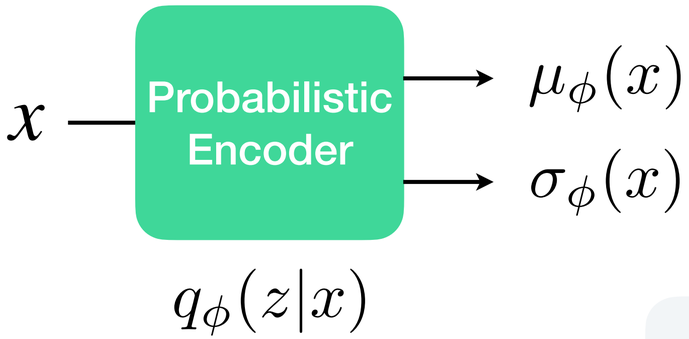
Variational Autoencoders

Kingma et al., (2014)

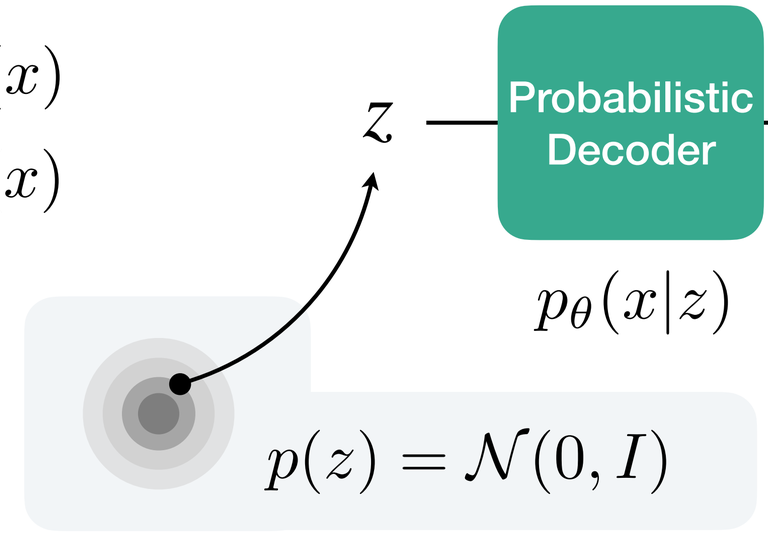
Variational autoencoders solve the generation problem of autoencoders by combining two ideas:

1. **Probabilistic Encoding**: Instead of mapping each input to a single point, map it to a distribution.
2. **Regularized Latent Space**: Force the latent space to have a *known structure we can sample from*.

1. **Encoder** outputs distribution parameters:



2. **Duing training** sample $z \sim \mathcal{N}\left(\mu(x), \sigma^2(x)\right)$

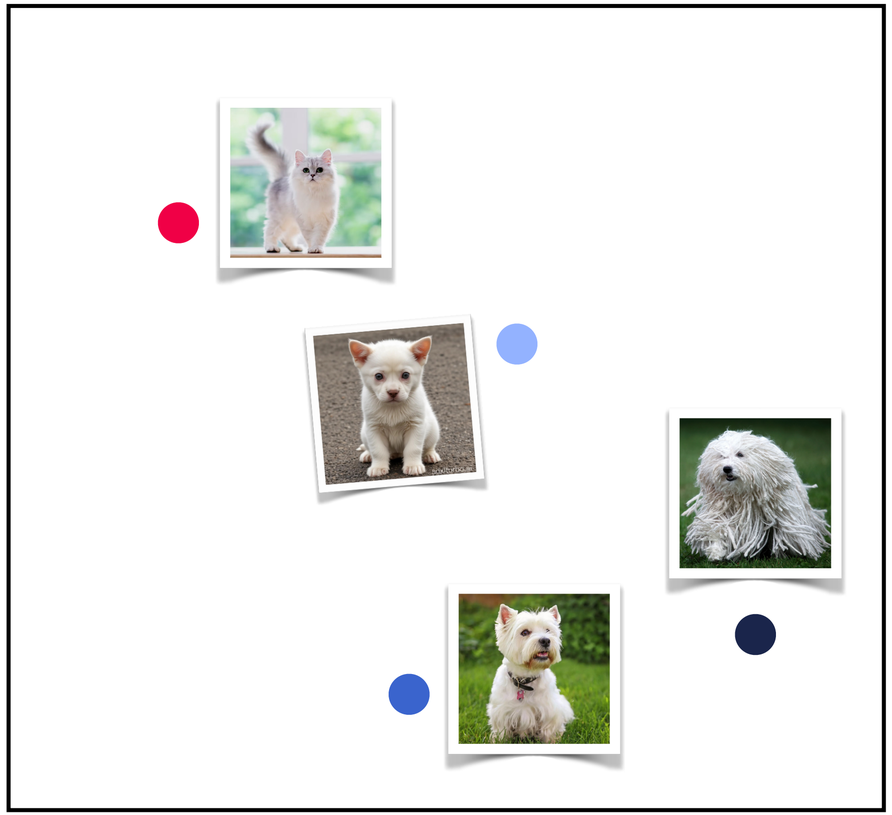


3. Generate by sampling $z \sim \mathcal{N}(0, I)$ and **decode**

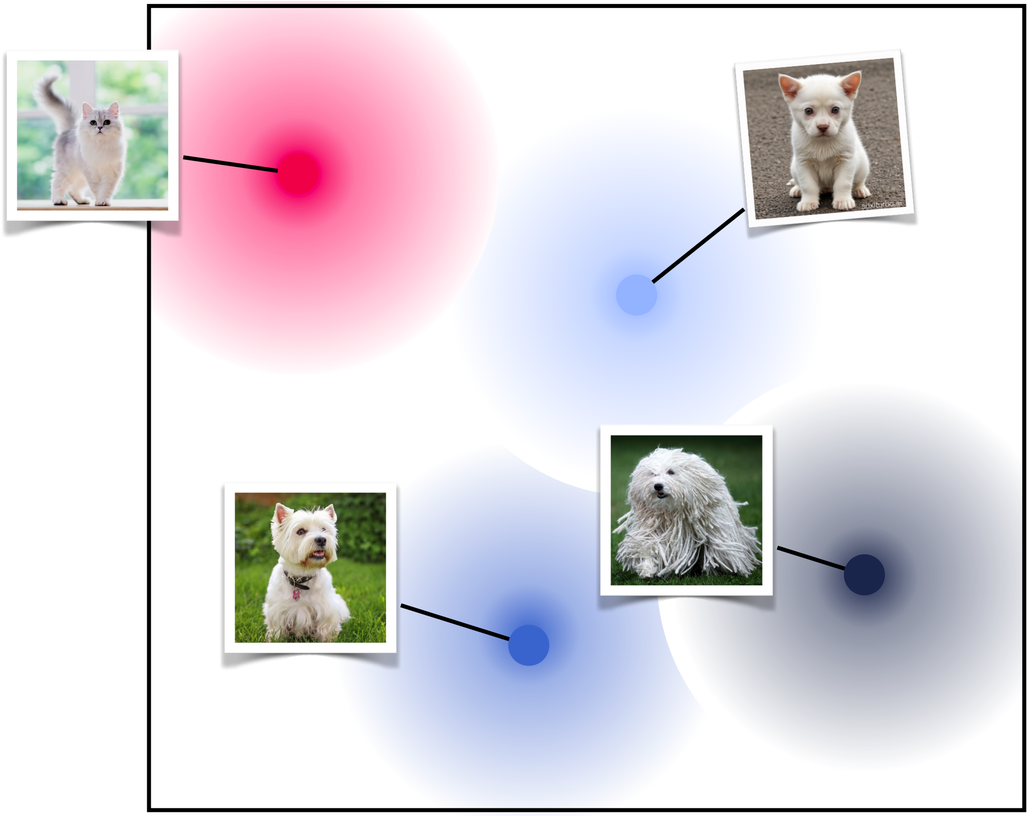
4. **Regularize** encoded distributions to be close to prior $\mathcal{N}(0, I)$

Variational Autoencoders

Latent Space in an Autoencoder



Latent Space in a Variational Autoencoder



→ continuous latent space

Variational Autoencoders

We want to **maximize the log-likelihood** of our data $\log p_\theta(x)$.

But with latent variables, this requires marginalizing

$$\log p_\theta(x) = \log \int p_\theta(x \mid z) p(z) dz$$

What we have:

- $p(z) = \mathcal{N}(0, I)$, i.e., a simple known prior.
- $p_\theta(x \mid z) =$ Decoder$_\theta$, i.e., a complex non-linear neural network.

$$\int \underbrace{\text{Decoder}_\theta}_{\text{complex decoder}} \cdot \underbrace{\mathcal{N}(0, I)}_{\text{simple prior}} \, dz = \; ???$$

→ No analytical solution exists!

Approximation through sampling not differentiable.

etc.

Problem!

Derivation: **Evidence Lower Bound (ELBO)**

We want to **maximize the log-likelihood** of our data $\log p_\theta(x)$.

But with latent variables, this requires marginalizing

Trick!

$$\log p_\theta(x) = \log \sum_z p_\theta(x, z) = \log \sum_z p_\theta(x|z)p(z)$$

"Which latent codes actually generated this data?"

Key Idea:
Introduce a distribution $q_\phi(z\,|\,x)$ that tries to guess which z values are likely for a given x.

$q_\phi(z|x)$

"Our best guess which latent codes generated this data"

Derivation: **Evidence Lower Bound (ELBO)**

Given the marginal likelihood $p_\theta(x) = \int p_\theta(x, z) dz$

For any distribution $q_\phi(z|x)$ (any distribution with support covering that of $p_\theta(z \mid x)$, we can write

$$= \int p_\theta(x, z) dz = \int q_\phi(z \mid x) \frac{p_\theta(x, z)}{q_\phi(z \mid x)} dz$$

Now take the log $\log p_\theta(x) = \log \int q_\phi(z \mid x) \frac{p_\theta(x, z)}{q_\phi(z \mid x)} dz$

Using the factorization $p_\theta(x, z) = p_\theta(x \mid z) p(z)$, we get

$$= \log \int q_\phi(z \mid x) \frac{p_\theta(x \mid z) p(z)}{q_\phi(z \mid x)} dz$$

Finally, recognize this as an expectation

$$= \log \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x \mid z) p(z)}{q_\phi(z \mid x)} \right]$$

Derivation: **Evidence Lower Bound (ELBO)**

$$\log p_\theta(x) = \log \mathbb{E}_{q_\phi(z|x)} \left[\frac{p_\theta(x \mid z) p(z)}{q_\phi(z \mid x)} \right]$$

Since log is concave, Jensen's inequality gives us $\log \mathbb{E}[X] \geq \mathbb{E}[\log X]$

Therefore

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x \mid z) p(z)}{q_\phi(z \mid x)} \right]$$

Separate the terms

$$\geq \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) \right] + \underbrace{\mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p(z)}{q_\phi(z \mid x)} \right]}_{-D_{\mathrm{KL}}(q_\phi(z|x) \| p(z))}$$

ELBO

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z) \right] - D_{\mathrm{KL}} \left(q_\phi(z \mid x) \| p(z) \right)$$

Variational Autoencoders

Exercise 3 · Task 2

Since we cannot compute $\log p_\theta(x)$ directly, we optimize the ELBO instead.

ELBO

$$\mathcal{L}(\phi, \theta) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - D_{\mathrm{KL}} \left(q_\phi(z \mid x) \| p(z)\right)$$

$\underbrace{\qquad\qquad}_{\text{Reconstruction term}}$ $\underbrace{\qquad\qquad}_{\text{Regularization term}}$

"How well can we reconstruct x from z?"

"How close is our encoding distribution to the prior?"

Why Is This Useful?

1. **It's tractable!**
   - Reconstruction: Sample z from q_ϕ, evaluate $\log p_\theta(x \mid z)$.
   - KL term: Often has closed form (e.g., for Gaussians).
2. **It is a lower bound:**
   - Maximizing ELBO pushes up the likelihood $\log p_\theta(x) \geq \mathcal{L}(\theta, \phi)$.
   - The gap equals $D_{\mathrm{KL}}(q_\phi(z \mid x) \| p_\theta(z \mid x))$.

Variational Autoencoders

To train our VAE, we need to compute gradients of the ELBO with respect to ϕ, i.e., parameters of the encoder.

$$\nabla_\phi \mathcal{L} = \nabla_\phi \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x \mid z)\right] - \underbrace{\nabla_\phi D_{\mathrm{KL}} \left(q_\phi(z \mid x) \| p(z)\right)}_{\text{has a closed form}}$$

Problem!

→ the distribution we are sampling from depends on ϕ and sampling is *not differentiable*

```python
mu, sigma = encoder(x)          # can backprop through this
z = random.normal(mu, sigma)    # cannot backprop through random sampling
x_recon = decoder(z)            # could backprop through this if we had gradients for z
```

Variational Autoencoders: Reparameterization Trick

Key Insight: We can express the same random variable in a different way!

Trick!

Instead of $z \sim \mathcal{N}\left(\mu, \sigma^2\right)$

we write $z = \mu + \sigma \odot \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(0, I)$

Why does this help?

- The randomness is now in ϵ, which does not depend on parameters.
- z is now a *deterministic* function of μ, σ, and ϵ.
- We can compute gradients with respect to μ and σ!

Variational Autoencoders: Putting Everything Together

```python
# encode
mu, log_var = encoder(x)
std = exp(0.5 * log_var)

# reparameterization
epsilon = randn_like(std)
z = mu + std * epsilon

# decode
x_recon = decoder(z)

# ELBO components
recon_loss = -log p(x|z)
kl_loss = 0.5 * sum(mu^2 + exp(log_var) - log_var - 1)

# loss is minimizing the negative ELBO
loss = -(recon_loss + beta*kl_loss)
```

Strengths:

- Principled probabilistic framework.
- Tractable training via ELBO.
- Smooth, interpolatable latent space.
- Fast generation after training.

Limitations:

- Blurry reconstructions as Gaussian likelihood assumption leads to averaging behavior.
- Posterior collapse: Encoder ignores input and outputs prior.
- Reconstruction ($\beta < 1$) vs. generation trade-off ($\beta > 1$).

Variants of Variational Autoencoders

1. **VQ-VAE**

Lecture 6: Vision Foundation Models

2. **VAE-Diffusion Hybrids**

Lecture 4: Generative Models II

and more …

Four Philosophical Approaches

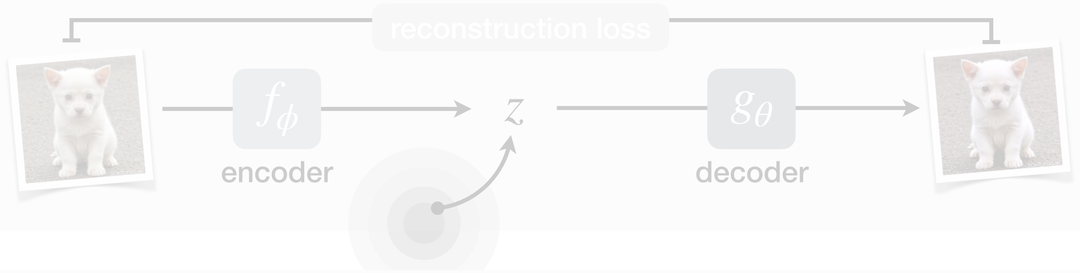
1. Autoregressive Models

"One step at a time."

2. Autoencoders

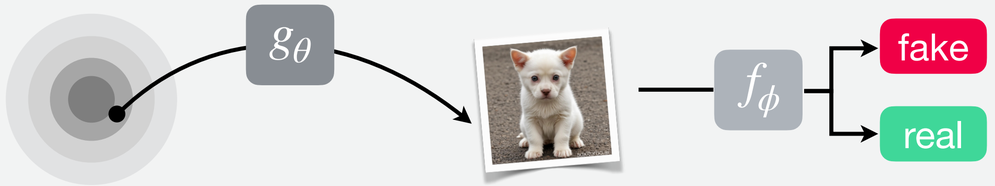
"Reduce to the essence and rebuild."

Variational Autoencoder



3. Generative Adversarial Models

"Fake it till you make it."



4. Diffusion Models

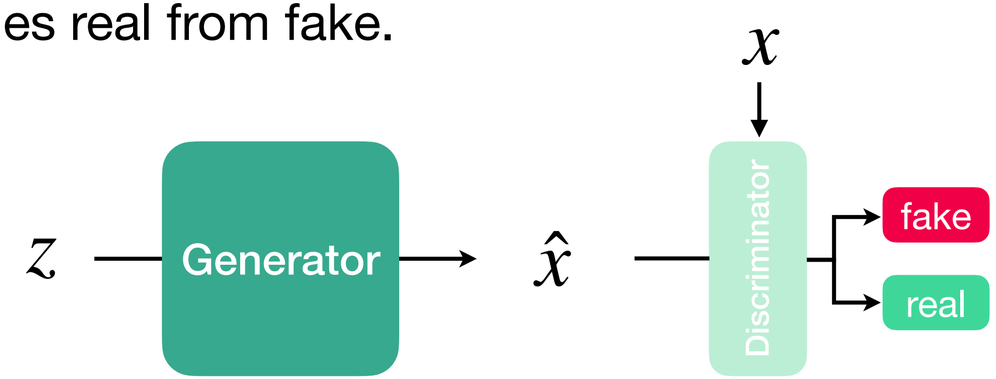
"Mess it up, then `Ctrl+Z`."

Generative Adversarial Networks

GANs take a radically different approach: instead of explicitly modeling $p(x)$,
learn to generate samples through a competitive game.

Two Players:

- **Generator** g_θ: Creates fake samples from random noise.
- **Discriminator** f_ϕ: Binary classifier that distinguishes real from fake.



The Game:

- g_θ tries to fool f_ϕ by generating realistic samples.
- f_ϕ tries to correctly classify real vs. generated.
- **At equilibrium**: g_θ generates perfect samples, f_ϕ outputs 0.5 everywhere.

Goodfellow et al., (2014)

Generative Adversarial Networks: Objective

The GAN objective is a two-player minimax game

Minimax Objective

$$\min_{g_\theta} \max_{f_\phi} V(f_\phi, g_\theta) = \mathbb{E}_{x \sim p_{\text{data}}}[\log f_\phi(x)] + \mathbb{E}_{z \sim p(z)}[\log(1 - f_\phi(g_\theta(z)))]$$

Discriminator f_ϕ's Objective:

- $\mathbb{E}\left[\log f_\phi(x)\right]$ → Reward for correctly identifying real data as real.
- $\mathbb{E}\left[\log\left(1 - f_\phi\left(g_\theta(z)\right)\right)\right]$ → Reward for correctly identifying generated data as fake.
- Optimal strategy: Output $f_\theta(\text{real}) = 1$ and $f_\theta(\text{fake}) = 0$.

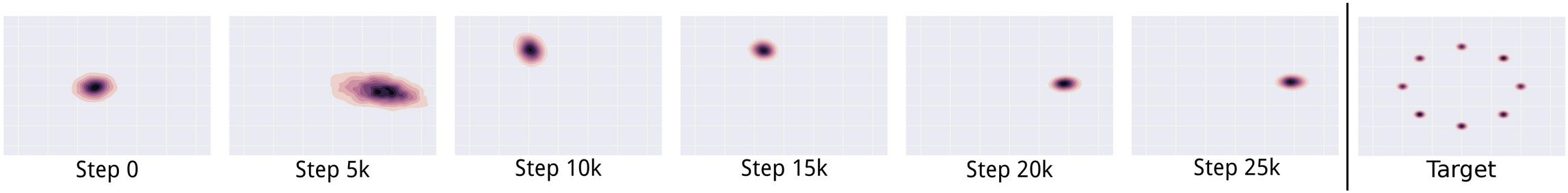
Generator g_θ's Objective:

- Can only influence the second term through $g_\theta(z)$.
- Wants $f_\phi\left(g_\theta(z)\right) \to 1$, i.e., generated samples classified as real.
- This drives the second term negative, reducing the discriminator's score.

At Equilibrium:

- $p_\theta = p_{\text{data}}$
- Discriminator outputs 0.5 everywhere.

Generative Adversarial Networks: Mode Collaps



Mode collapse: Generator produces limited diversity, mapping all noise to a few outputs.

Why it happens?

1. **Local optimization**: Generator finds it easier to perfect one mode than cover all modes.
2. **Discriminator Overfitting**: If f_ϕ is too good, gradient for g_θ vanishes except at specific points.
3. **No Explicit Diversity Term**: Unlike VAEs, no KL regularization forcing coverage.

Metz et al., (2017)

Excursion: **Jensen-Shannon Divergence**

Definition: Symmetric version of KL divergence.

Jensen-Shannon Divergence

$$D_{\mathrm{JS}}(p\|q) = \frac{1}{2}D_{\mathrm{KL}}(p\|m) + \frac{1}{2}D_{\mathrm{KL}}(q\|m)$$

where $m = \frac{1}{2}(p+q)$.

Properties:

Symmetric, i.e., $D_{\mathrm{JS}}(p\|q) = D_{\mathrm{JS}}(q\|p)$.

Bounded: $0 \leq D_{\mathrm{JS}}(p\|q) \leq \log 2$

$D_{\mathrm{JS}}(p\|q) = 0$ if and only if $p = q$.

Why not use directly $D_{\mathrm{KL}}(p\|q)$ or $D_{\mathrm{KL}}(q\|p)$?

→ D_{KL} is assymetric and not defined if p puts mass where $q = 0$.

→ Common ground distribution where both p and q are absolutely continuous is needed, i.e., mixture m.

Generative Adversarial Networks: Theoretical Analysis

The Optimal Discriminator: Solving $\max_{f_\phi} V(f_\phi, g_\theta)$ for a fixed generator g_θ,

Exercise 3 · Task 3

the optimal discriminator is defined as $f_\phi^*(x) = \dfrac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{g_\theta}(x)}$

Substituting into $V(f_\phi, g_\theta)$:

$$V\left(f_\phi^*, g_\theta\right) = \mathbb{E}_{x\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}\left(x\right)}{\underbrace{p_{\text{data}}\left(x\right) + p_{g_\theta}(x)}_{2m(x)}}\right] + \mathbb{E}_{x\sim p_{g_\theta}}\left[\log \frac{p_{g_\theta}(x)}{\underbrace{p_{\text{data}}\left(x\right) + p_{g_\theta}(x)}_{2m(x)}}\right]$$

Let's rewrite this as $m(x) = \frac{1}{2}\left(p_{\text{data}}\left(x\right) + p_{g_\theta}(x)\right)$

and expand the logarithms

$$= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log p_{\text{data}}\left(x\right)\right] - \mathbb{E}_{x\sim p_{\text{data}}}\left[\log 2m(x)\right] + \mathbb{E}_{x\sim p_{g_\theta}}\left[\log p_{g_\theta}(x)\right] - \mathbb{E}_{x\sim p_{g_\theta}}\left[\log 2m(x)\right]$$

Generative Adversarial Networks: Theoretical Analysis

$$= \mathbb{E}_{x\sim p_{\text{data}}}\left[\log p_{\text{data}}\left(x\right)\right] - \mathbb{E}_{x\sim p_{\text{data}}}\left[\log 2m(x)\right] + \mathbb{E}_{x\sim p_{g_\theta}}\left[\log p_{g_\theta}(x)\right] - \mathbb{E}_{x\sim p_{g_\theta}}\left[\log 2m(x)\right]$$

Recognizing KL divergences:

- $\mathbb{E}_{x\sim p_{\text{data}}}\left[\log p_{\text{data}}\left(x\right)\right] - \mathbb{E}_{x\sim p_{\text{data}}}\left[\log m(x)\right] = D_{\text{KL}}\left(p_{\text{data}}\,\|m\right)$
- $\mathbb{E}_{x\sim p_{g_\theta}}\left[\log p_{g_\theta}(x)\right] - \mathbb{E}_{x\sim p_{g_\theta}}\left[\log m(x)\right] = D_{\text{KL}}\left(p_{g_\theta}\|m\right)$

The log 2 terms:

- $-\mathbb{E}_{x\sim p_{\text{data}}}\left[\log 2\right] - \mathbb{E}_{x\sim p_{g_\theta}}\left[\log 2\right] = -\log 2 - \log 2 = -2\log 2 = -\log 4$

Key Result:

Training GANs is equivalent to minimizing JS divergence!

$$V\left(f_\phi^*, g_\theta\right) = -\log 4 + D_{\text{KL}}\left(p_{\text{data}}\|m\right) + D_{\text{KL}}\left(p_{g_\theta}\|m\right)$$

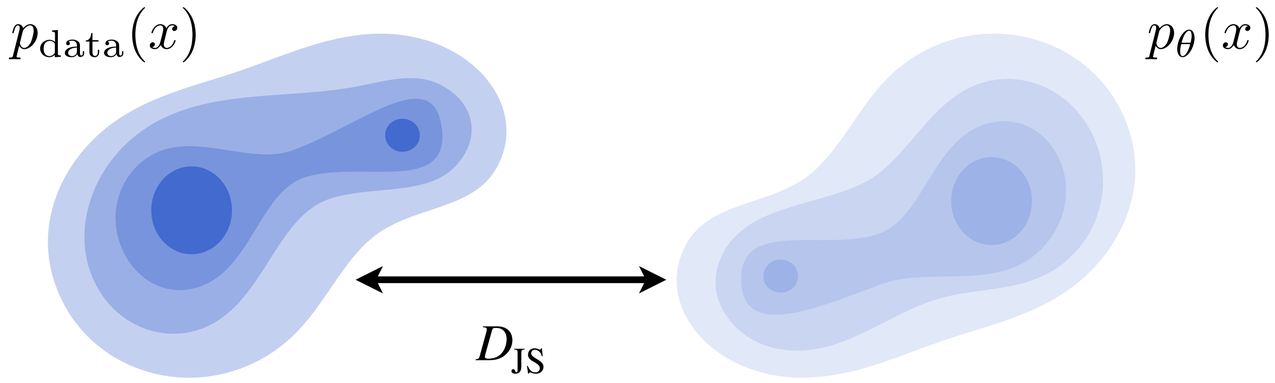
$$= -\log 4 + 2\cdot D_{\text{JS}}\left(p_{\text{data}}\,\|p_{g_\theta}\right)$$

A Distribution-Based Perspective on Generative Adversarial Networks

Key Result:

Training GANs is equivalent to minimizing JS divergence!

$$\min_{g_\theta} D_{\mathrm{JS}}\left(p_{\mathrm{data}} \,\|p_{g_\theta}\right)$$



Core Issue: When the real and generated distributions do not overlap, JS divergence becomes constant, i.e., $\log 2$ (its maximum value).

Gradient with respect to generator parameters $\nabla_\theta D_{\mathrm{JS}} = 0$.

No learning signal!

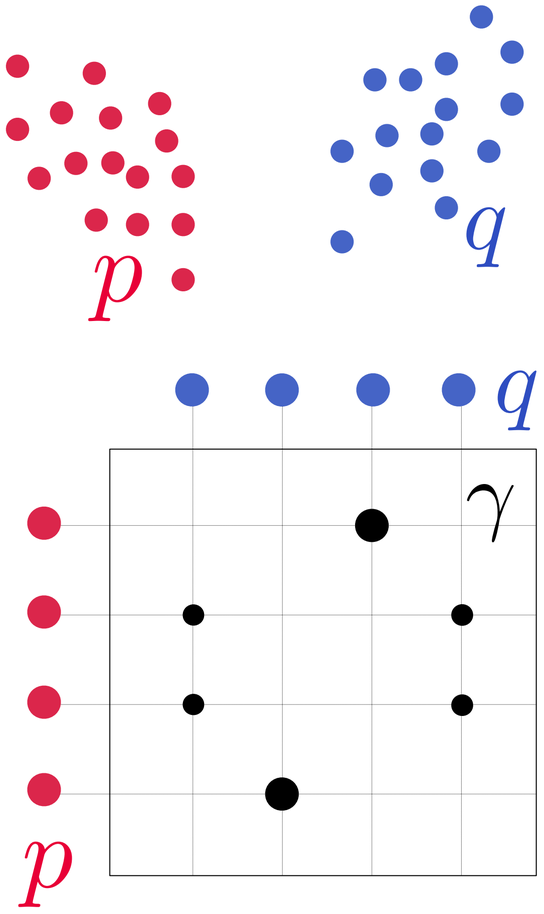
Excursion: **Wasserstein Distance**

Definition: Minimum expected cost to transport mass from one distribution to another.

Wasserstein Distence

$$W_1(p, q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

ℓ_1 distance as cost function



γ: joint distribution that describes how to transport mass from p to q.

$\Pi(p, q)$: set of all valid joint distributions whose marginals match p and q.

$\|x - y\|$: cost of transporting a unit mass from location x to location y.

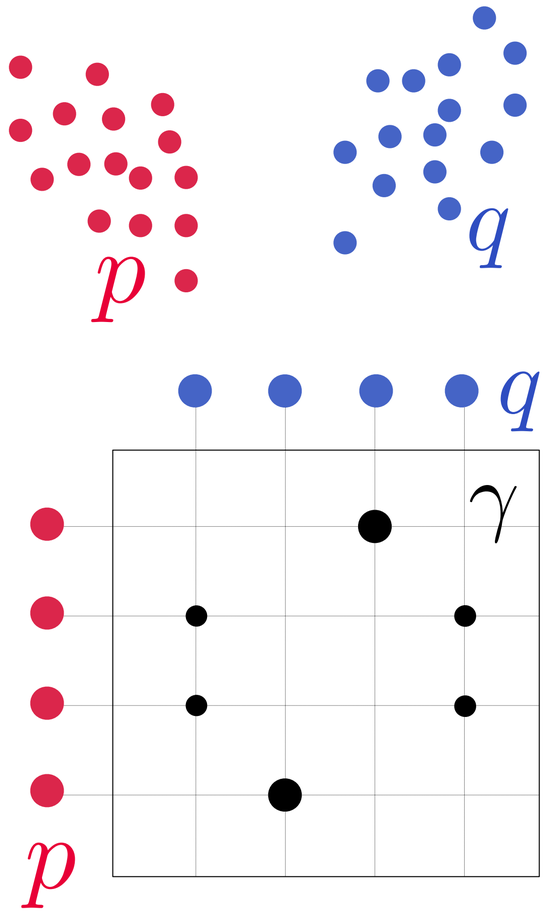
$\gamma(x, y)$ represents the amount of probability mass moved from point x to point y.

Excursion: **Wasserstein Distance**

Definition: Minimum expected cost to transport mass from one distribution to another.

Wasserstein Distence

$$W_1(p, q) = \inf_{\gamma \in \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$



γ must satisfy marginal constraints:

- $\int \gamma(x, y)dy = p(x)$, i.e., integrating out y gives back p.
- $\int \gamma(x, y)dx = q(y)$, i.e., integrating out x gives back q.

Key advantage:

Provides meaningful gradients even when distributions do not overlap.

Excursion: **Wasserstein Distance**

Dual of the Wasserstein Distance:
Lagrange duality transforms the constrained optimization over γ into an unconstrained maximization over 1-Lipschitz functions.

Kantorovich-Rubinstein Duality

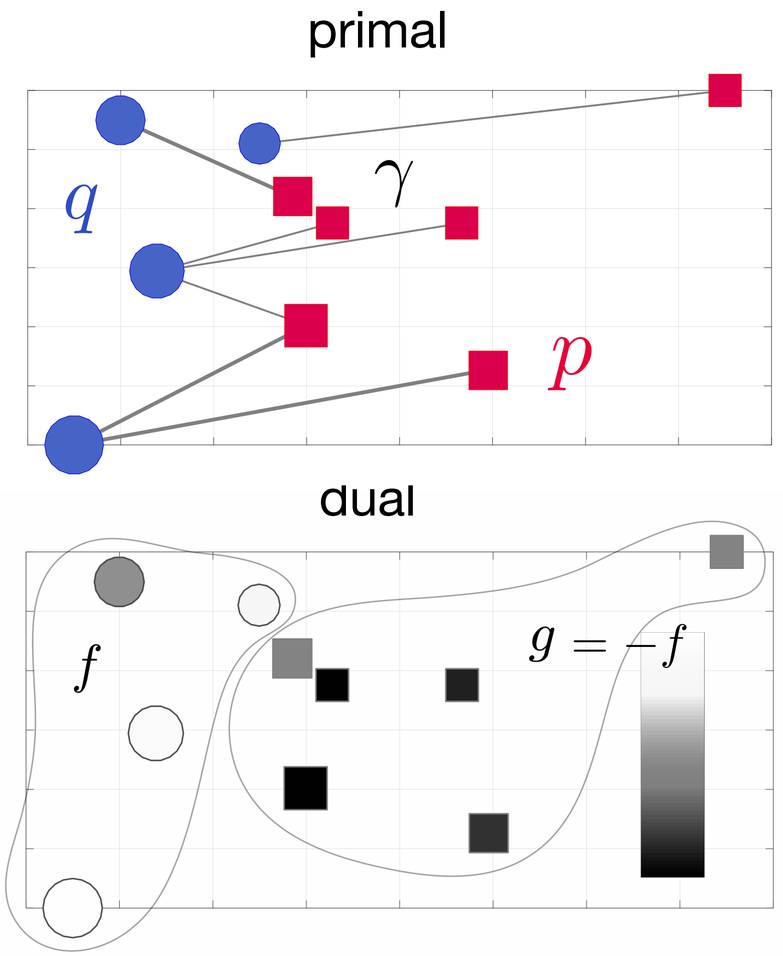
$$W_1(p,q) = \sup_{\|f\|_L \leq 1} \left(\mathbb{E}_{x\sim p}[f(x)] - \mathbb{E}_{y\sim q}[f(y)]\right)$$

where $\|f\|_L \leq 1$ means f is 1-Lipschitz.

f: dual potential function.

1-Lipschitz: $|f(x) - f(y)| \leq \|x - y\|$,
i.e., function canot change too rapidly.

sup: Find the f that maximizes the difference.



Wasserstein Generative Adversarial Networks

Exercise 3 · Task 4

Classic GAN

$$\min_{g_\theta} \max_{f_\phi} \mathbb{E}_{x \sim p_{\text{data}}} \left[\log f_\phi(x)\right] + \mathbb{E}_{z \sim p(z)} \left[\log\left(1 - f_\phi\left(g_\theta(z)\right)\right)\right]$$

GAN

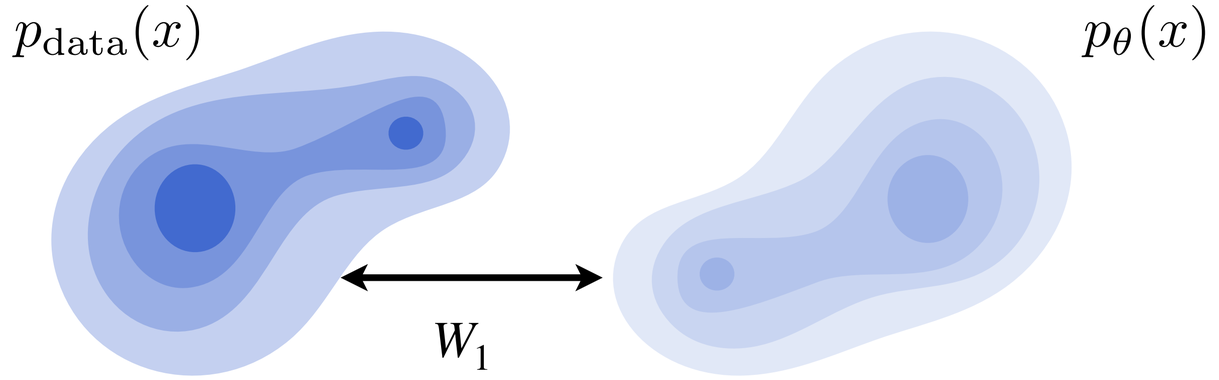
Wasserstein GAN (WGAN)

$$\min_{g_\theta} \max_{\|f_\phi\|_L \leq 1} \mathbb{E}_{x \sim p_{\text{data}}} \left[f_\phi(x)\right] - \mathbb{E}_{z \sim p(z)} \left[f_\phi\left(g_\theta(z)\right)\right]$$

WGAN

Interpretation:

- **Discriminator → Critic**:
  Outputs real values, not probabilities.
- Must enforce: 1-Lipschitz constraint,
  e.g., via weight clipping, gradient penalty, etc.



Arjovsky et al., (2017)

Wasserstein Generative Adversarial Networks

Generalize beyond W_1 to W_2...

There's a deep theoretical connection between W_2 GANs and diffusion models through **optimal transport theory**.

Next week!

Beyond the scope of this course.

→ Tutorial at **ICML 2023**

Optimal Transport in Learning, Control, and Dynamical Systems

Charlotte Bunne

ICML Tutorial 2023

Optimal transport (OT) theory (Santambrogio, 2015; Villani, 2003, 2009) is a core element of the machine learning toolbox and has become within a few years the go-to framework to analyze, model, and solve an ever-increasing variety of tasks involving probability measures. This is best exemplified by its increasing importance to fitting generative models, where the goal is to learn a map (Arjovsky et al., 2017; Genevay et al., 2018; Salimans et al., 2018), or more generally a diffusion (Song et al., 2021; De Bortoli et al., 2021) to morph a simple measure (e.g., Gaussian) onto a data distribution of interest (e.g., images). This is also apparent in the many applications that use OT to align probability measures that have since arisen, e.g., to transfer label knowledge between datasets (Flamary et al., 2016; Singh and Jaggi, 2020), to analyze sampling schemes (Dalalyan, 2017), or study population trajectories (Schiebinger et al., 2019; Bunne et al., 2023b).

In this tutorial, we primarily cast light on the static and dynamic formulations of optimal transport, and simultaneously establish their theoretical nexus by recalling its mathematical history from Monge and Kantorovich to modern Fields Medal winners Villani, Figalli, and Abel and Wolf Prize recipient Caffarelli in order to provide a solid foundation for the discussion ahead.

Contents

Strengths of GANs:

- Sharp, high-quality samples.
- No explicit density required.
- Flexible architecture design.
- Fast generation.

Limitations of GANs:

- Training instability.
- Mode collapse.
- No explicit likelihood.
- Difficult hyperparameter tuning.

Code Notebook 3 · Demo

Four Philosophical Approaches

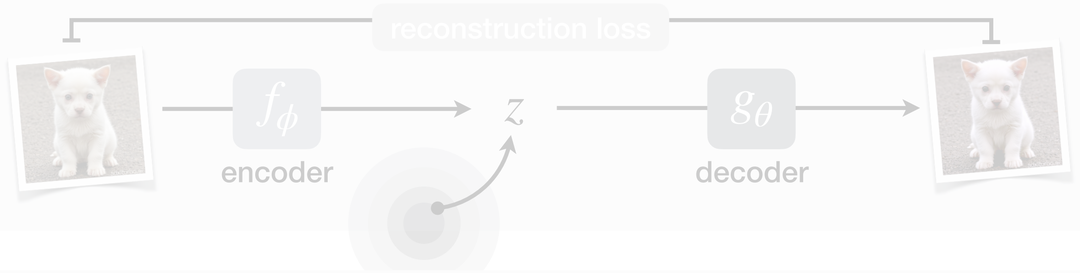
1. Autoregressive Models

"One step at a time."

2. Autoencoders

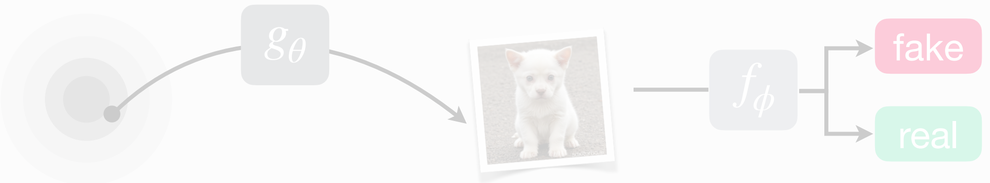
"Reduce to the essence and rebuild."

Variational Autoencoder



3. Generative Adversarial Models

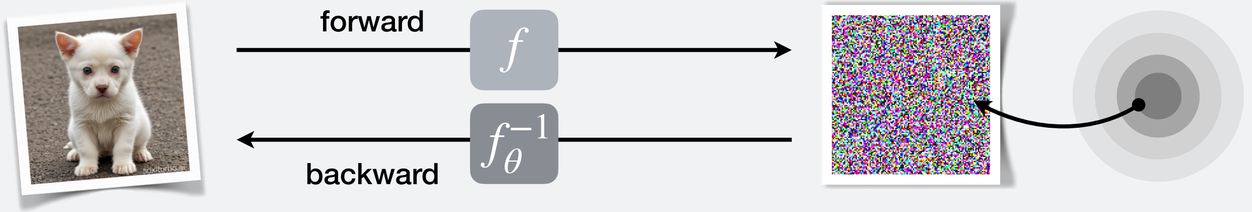
"Fake it till you make it."



Next week!

4. Diffusion Models

"Mess it up, then `Ctrl+Z`*."*



Role of Generative AI in Foundation Models

Generative Modeling as Self-Supervision

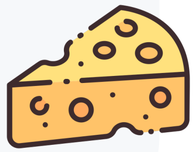
Lecture 2: Learning at Scale

Principle:
Predicting unobserved from observed provides *unlimited* training signal.

Key Insight:
Generation forces models to learn structure, not just correlations.

"Teaching a model to generate teaches it to understand."

"If you want to master something, teach it." – Richard Feynman



Generative Models as Architectural DNA

Generative AI spans the entire FM building block stack!

i.e.,

tokenization,
core architectures,
learning principles,

and more.

Lecture 5-9: Architectures

From Representation to Simulation

Traditional models:
Learn representations
→ *what is!*

Generative models:
Enable simulation
→ *what if!*

Lecture 13: Reasoning

Lecture 12: World Models

Why Generative Training Produces "Foundation" Capabilities

1. **Compression Requires Understanding**
   To predict/generate accurately, must learn underlying structure.

2. **Self-Supervision Scales**
   Generation provides unlimited training signal from raw data.

3. **Emergence from Scale**
   Generative objectives + scale → capabilities not explicitly trained.

4. **Conditional Generation = Universal Interface**
   $p(\text{output} \mid \text{prompt})$ handles *any* task.

Paradigm Shift!

We don't train models to do tasks anymore: we train them to model data distributions, and tasks emerge as conditional generation.

This Week's Papers

No readings this week!

This Week's Exercise Sheet

Only code exercises this week!

This Week's Code Demonstration

Code Notebook 2 · Task 1

Exploring Contrastive Learning with SimCLR (follow-up from last week): Hands-on implementation and exploration of the SimCLR framework.
→ Jupyter notebook exercise

Code Notebook 2 · Task 2

Exploring Self-Distillation with BYOL: Hands-on implementation and exploration of the BYOL framework.
→ Jupyter notebook exercise

Code Notebook 2 · Task 3

Exploring Redundancy Reduction-based Learning with Barlow Twins: Hands-on implementation and exploration of the Barlow Twins framework.
→ Jupyter notebook exercise

CS-461

Foundation Models and Generative AI

Have a great week!