Deep Generative Modeling is a key to unlocking AI potential

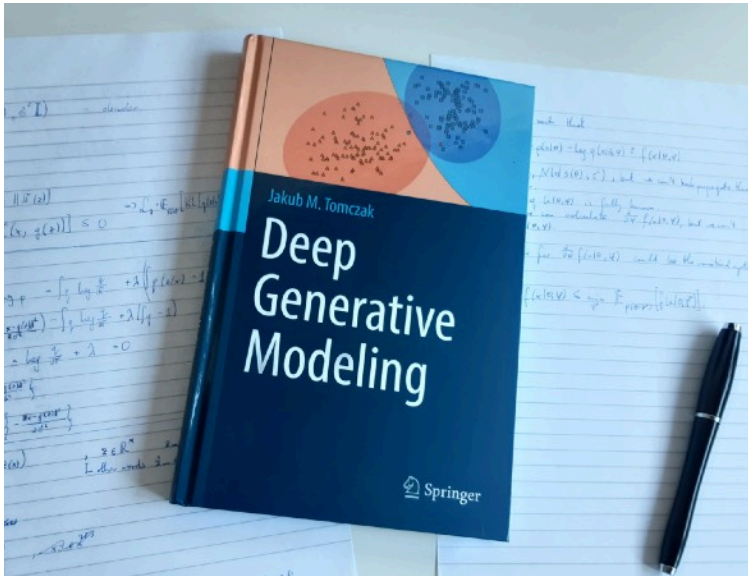
Jakub M. Tomczak

VU VRIJE UNIVERSITEIT AMSTERDAM

MORE ABOUT DEEP GENERATIVE MODELING

Blog: https://jmtomczak.github.io/blog.html

Book: https://link.springer.com/book/10.1007/978-3-030-93158-2



INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **intelligence**?

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **intelligence**?

…

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **intelligence**?

…

What is **artificial intelligence**?

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **intelligence**?

…

What is **artificial intelligence**?

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **intelligence**?

…

What is **artificial intelligence**?

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **intelligence**?

…

What is **artificial intelligence**?

Decision/action

Information

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **intelligence**?

…

What is **artificial intelligence**?

Decision/action

Information

0001101010011…

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **artificial intelligence**?

- **Information** processing
- **Information** storing
- **Information** transmission

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **artificial intelligence**?

- **Information** processing
- **Information** storing
- **Information** transmission
- **Decision** making

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **artificial intelligence**?

- **Information** processing
- **Information** storing
- **Information** transmission
- **Decision** making

} **Learning**, **Knowledge representation**, **Models…**

INFORMATION, INTELLIGENCE AND ARTIFICIAL INTELLIGENCE

What is **artificial intelligence**?

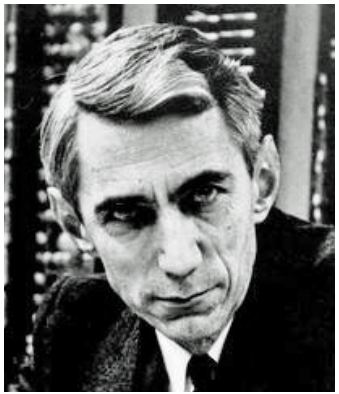
- **Information** processing
- **Information** storing
- **Information** transmission
- **Decision** making

} **Learning**
Knowledge representation
Models…

The question is how to formalize the problem of AI?

VU

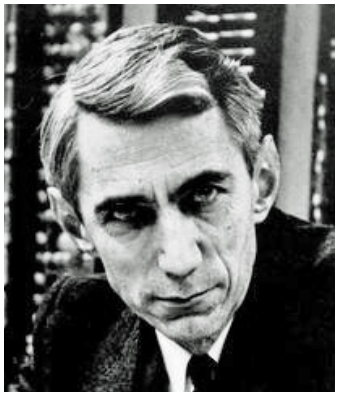
Information (a quick recap)



Claude Shannon

Information (a quick recap)

We have a random source of data x.



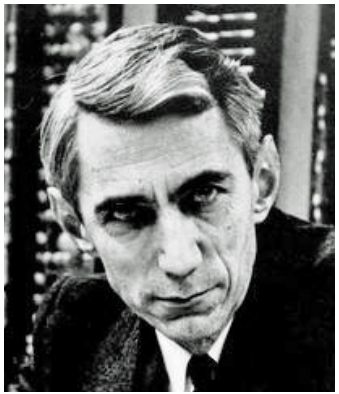
Claude Shannon

Information (a quick recap)

We have a random source of data x.

We can quantify the **uncertainty** of this source by calculating **the entropy**:

$$\mathbb{H}[x] = -\sum_{x} p(x)\log p(x)$$



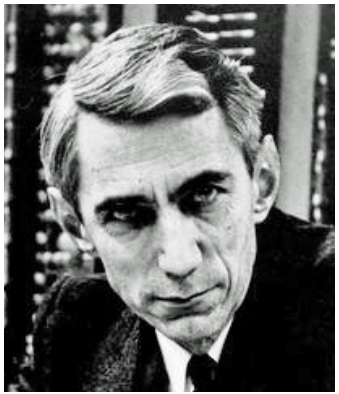
Claude Shannon

Information (a quick recap)

We have a random source of data x.

We can quantify the **uncertainty** of this source by calculating **the entropy**:

$$\mathbb{H}[x] = -\sum_{x} p(x)\log p(x)$$



Claude Shannon

Entropy is max if all x's are equiprobable.

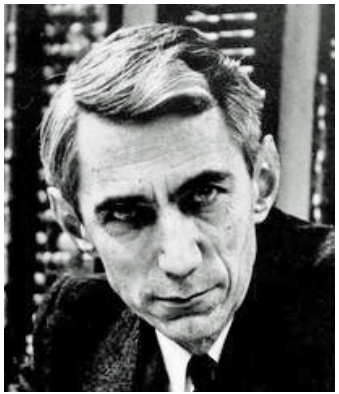
Entropy is min if the probability of one value is 1.

Information (a quick recap)

We have a random source of data x.

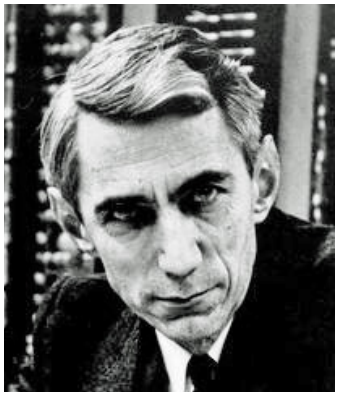
We can quantify the **uncertainty** of this source by calculating **the entropy**:

$$\mathbb{H}[x] = -\sum_{x} p(x)\log p(x)$$



Claude Shannon

Optimal message length $\approx$ the entropy.


Claude Shannon

Information (a quick recap)

We have two random sources: x and y.

We can quantify the **uncertainty** of them by calculating **the joint entropy**:

$$\mathbb{H}[x, y] = -\sum_{x,y} p(x, y) \log p(x, y)$$

or **the conditional entropy**:

$$\mathbb{H}[y \mid x] = -\sum_{x,y} p(x, y) \log p(y \mid x)$$

VU

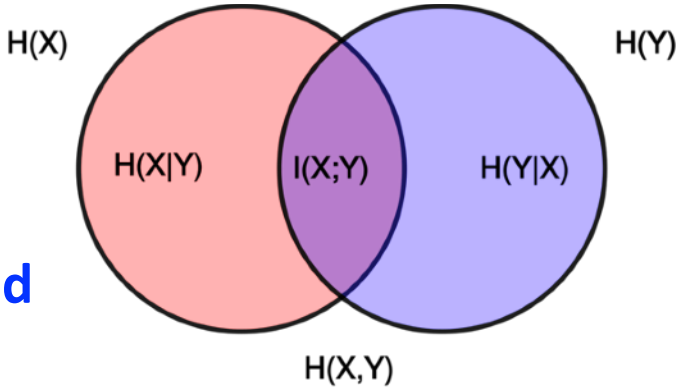
Mutual Information (a quick recap)

We have two random sources: x and y.

Mutual Information (a quick recap)

We have two random sources: x and y.

We can quantify how much **information is shared by the two sources:**

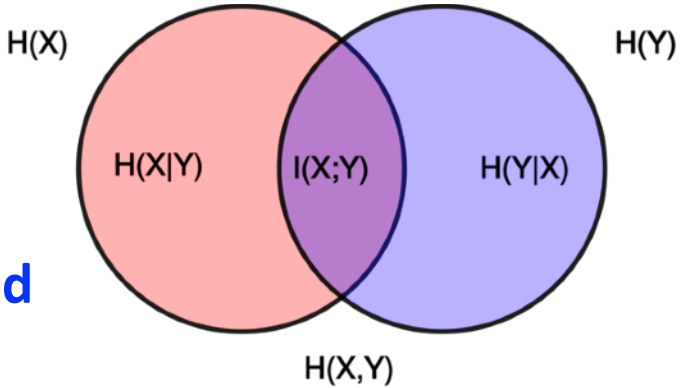


$$\mathbb{I}[x;y] = \mathbb{H}[y] - \mathbb{H}[y\,|\,x]$$

Mutual Information (a quick recap)

We have two random sources: x and y.

We can quantify how much **information is shared by the two sources:**



$$\mathbb{I}[x; y] = \mathbb{H}[y] - \mathbb{H}[y \,|\, x]$$

or **how much knowing one source reduces uncertainty about the other.**

We have two random sources: x (e.g., images) and y (e.g., decisions).

We have two random sources: x (e.g., images) and y (e.g., decisions).

We have also a **model** m (a representation of a world).

We have two random sources: x (e.g., images) and y (e.g., decisions).

We have also a **model** m (a representation of a world).

The **goal** of AI: **maximize** the **mutual information** between (x, y) and m:

$$\mathbb{I}[(x, y); m] = \mathbb{H}[x, y] - \mathbb{H}[x, y \mid m]$$

We have two random sources: x (e.g., images) and y (e.g., decisions).

We have also a **model** m (a representation of a world).

The **goal** of AI: **maximize** the **mutual information** between (x, y) and m:

$$\mathbb{I}[(x, y); m] = \mathbb{H}[x, y] - \mathbb{H}[x, y \mid m]$$

Entropy of the world
(model has no influence on that)

That's the "real" goal!

The **goal** of AI: **maximize** the **mutual information** between (x, y) and m (or minimize $\mathbb{H}[x, y \mid m]$, i.e., minimize uncertainty of the world):

$$\mathbb{H}[x, y \mid m] = \sum_{x,y,m} p(x, y, m) \left[\log p(y \mid x, m) + \log p(x \mid m) \right]$$

The **goal** of AI: **maximize** the **mutual information** between (x, y) and m (or minimize $\mathbb{H}[x, y \mid m]$, i.e., minimize uncertainty of the world):

$$\mathbb{H}[x, y \mid m] = \sum_{x,y,m} p(x, y, m) \left[\log p(y \mid x, m) + \log p(x \mid m) \right]$$

A model for decision making

A model for understanding the world.

The **goal** of AI: **maximize** the **mutual information** between (x, y) and m (or minimize $\mathbb{H}[x, y \mid m]$, i.e., minimize uncertainty of the world).

In order to achieve that, AI should focus on learning **two models**:

- **A model for decision making:** $p(y \mid x, m)$
- **A model for understanding the world:** $p(x \mid m)$

WHAT HAPPENS IF WE LEARN ONLY DECISION MAKING

The bulk of AI is focused **only** on the decision making part!

WHAT HAPPENS IF WE LEARN ONLY DECISION MAKING

The bulk of AI is focused **only** on the decision making part!

Example: Let's say we have a model that is well trained.

$p(y = cat|\mathbf{x}) = 0.90$
$p(y = dog|\mathbf{x}) = 0.05$
$p(y = horse|\mathbf{x}) = 0.05$

WHAT HAPPENS IF WE LEARN ONLY DECISION MAKING

The bulk of AI is focused **only** on the decision making part!

Example: Let's say we have a model that is well trained.

$p(y = cat|\mathbf{x}) = 0.90$
$p(y = dog|\mathbf{x}) = 0.05$
$p(y = horse|\mathbf{x}) = 0.05$

+

noise

=

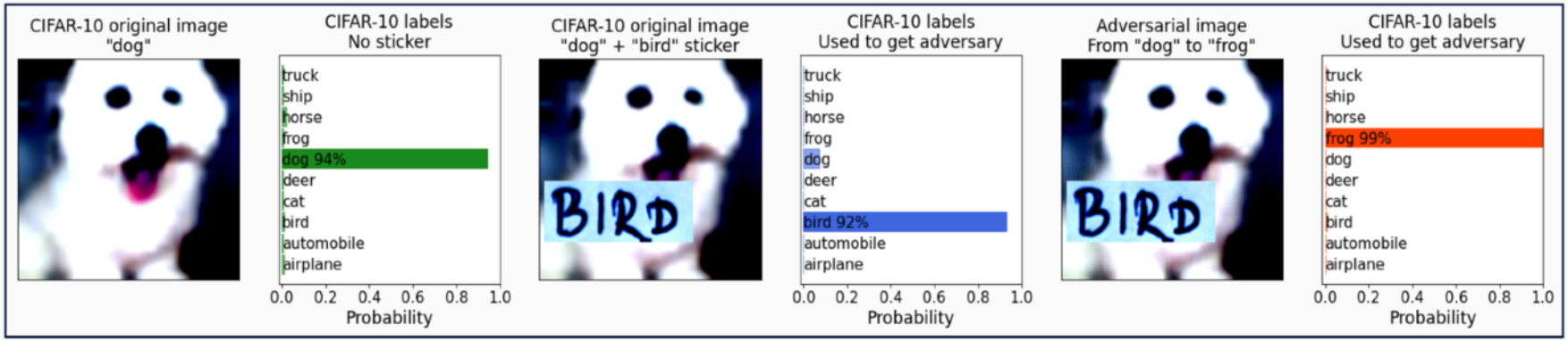
$p(y = cat|\mathbf{x}) = 0.05$
$p(y = dog|\mathbf{x}) = 0.05$
$p(y = horse|\mathbf{x}) = 0.90$

But after adding a little noise it could fail completely…

IS LEARNING CLASSIFIERS ENOUGH?

Let’s assume we have a perfectly trained neural net.

What happens if we add (adversarial) noise to an image?

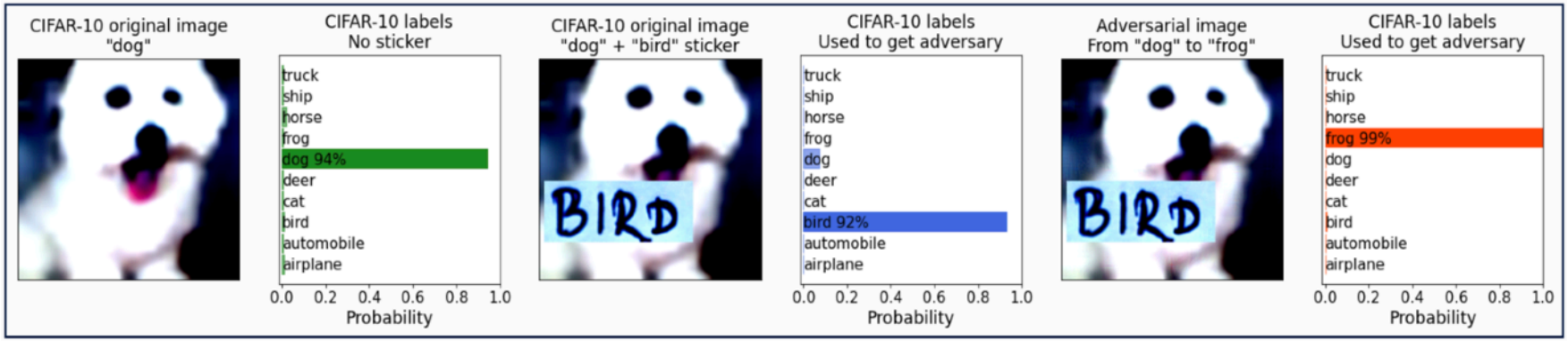


S. Fort, “Pixels still beat text: Attacking the OpenAI CLIP model with text patches and adversarial pixel perturbations”, [Link]

IS LEARNING CLASSIFIERS ENOUGH?

Let's assume we have a perfectly trained neural net.

What happens if we add (adversarial) noise to an image?

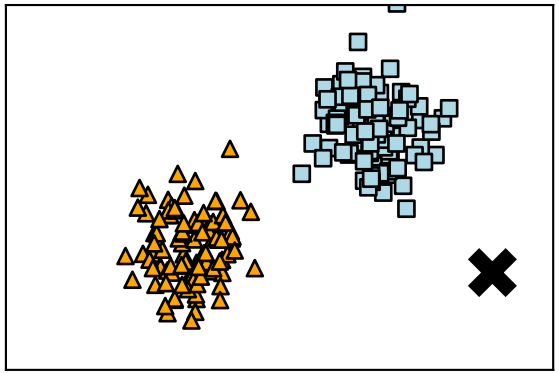


It fails completely…

S. Fort, "Pixels still beat text: Attacking the OpenAI CLIP model with text patches and adversarial pixel perturbations", [Link]

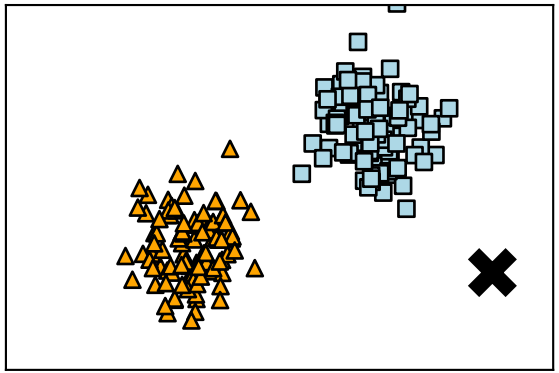
DEEP GENERATIVE MODELING: WHY DO WE NEED IT?

DEEP GENERATIVE MODELING: WHY DO WE NEED IT?

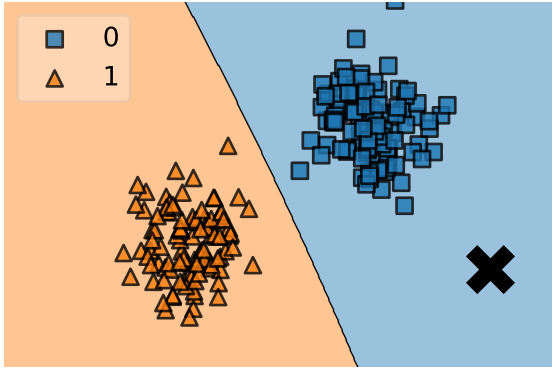


Data

DEEP GENERATIVE MODELING: WHY DO WE NEED IT?



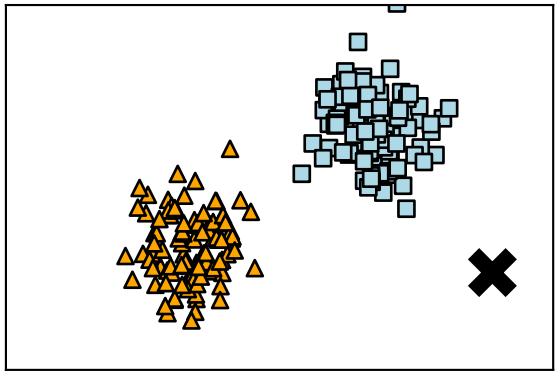
Data



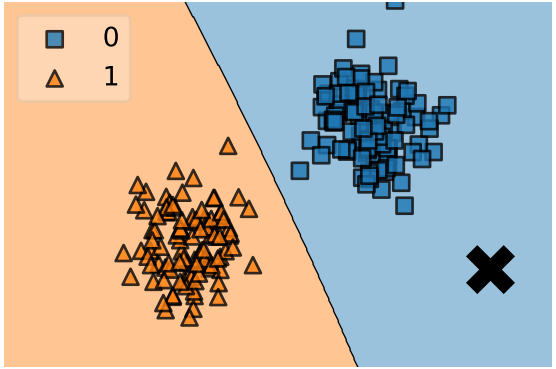
$p(y|\mathbf{x})$

$p(blue|\mathbf{x})$ is high
= certain decision!

DEEP GENERATIVE MODELING: WHY DO WE NEED IT?

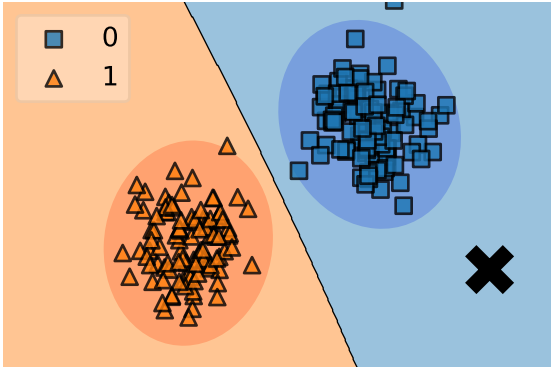


Data



$p(y|\mathbf{x})$

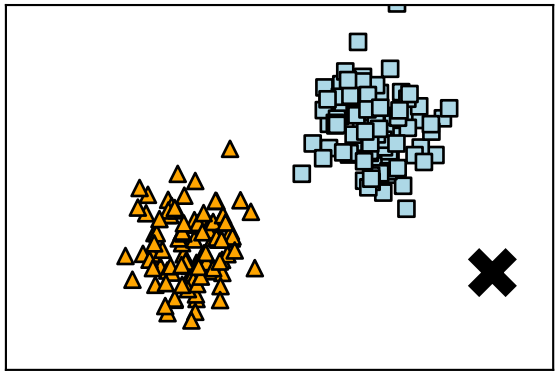
$p(blue|\mathbf{x})$ is high
= certain decision!



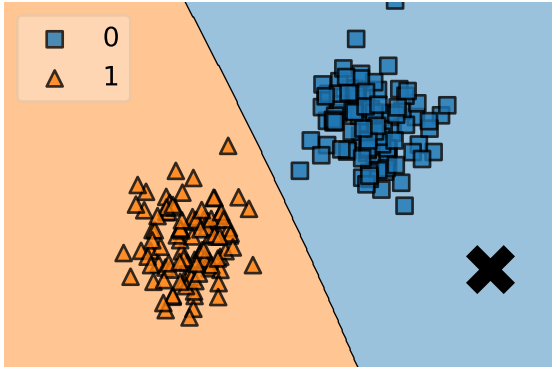
$p(\mathbf{x}, y) = p(y|\mathbf{x})\ p(\mathbf{x})$

$p(blue|\mathbf{x})$ is high
and $p(\mathbf{x})$ is low
= uncertain decision!

DEEP GENERATIVE MODELING: WHY DO WE NEED IT?

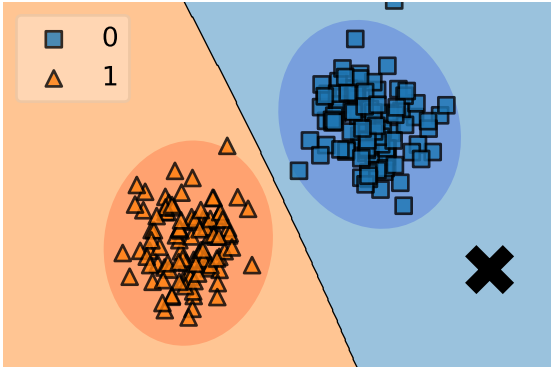


Data



$p(y|\mathbf{x})$

$p(blue|\mathbf{x})$ is high
= certain decision!

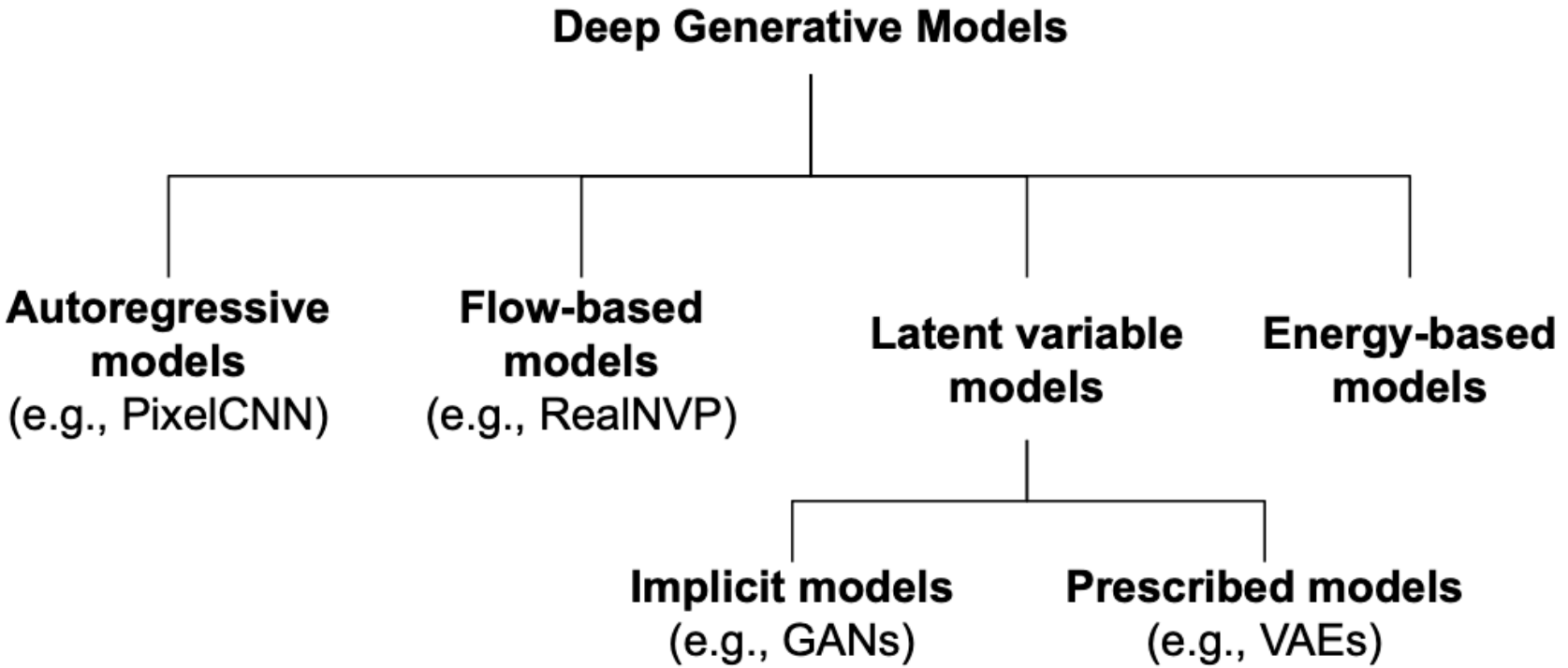


$p(\mathbf{x}, y) = p(y|\mathbf{x})\ p(\mathbf{x})$

$p(blue|\mathbf{x})$ is high
and $p(\mathbf{x})$ is low
= uncertain decision!

Thus, learning the conditional is only a part of the story!
How can we learn p(x)?

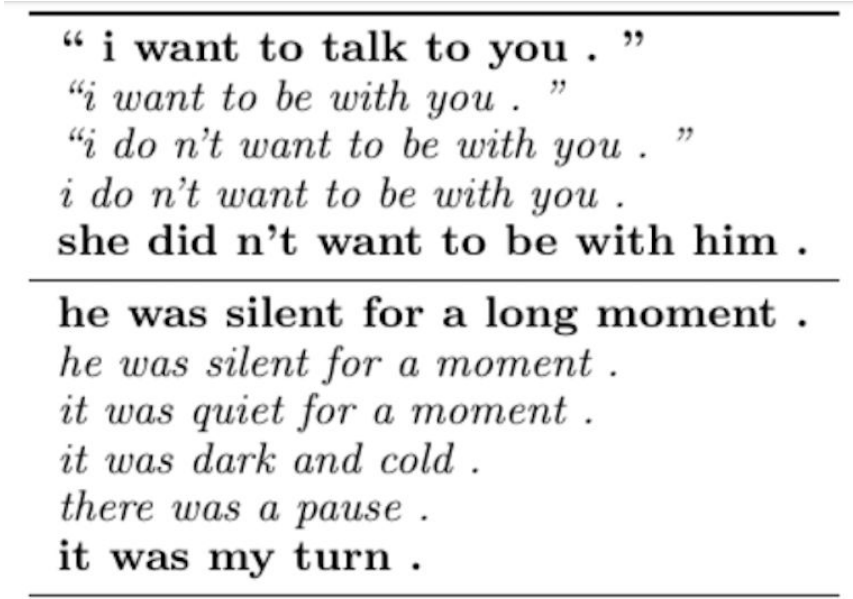
DEEP GENERATIVE MODELING: HOW WE CAN FORMULATE IT?



DEEP GENERATIVE MODELING: HOW WE CAN FORMULATE IT?

Generative models	Training	Likelihood	Sampling	Compression	Representation
Autoregressive models	stable	exact	slow	lossless	no
Flow-based models	stable	exact	fast/slow	lossless	yes
Implicit models	unstable	no	fast	no	no
Prescribed models	stable	approximate	fast	lossy	yes
Energy-based models	stable	unnormalized	slow	rather not	yes

DEEP GENERATIVE MODELING: WHERE CAN WE USE IT?

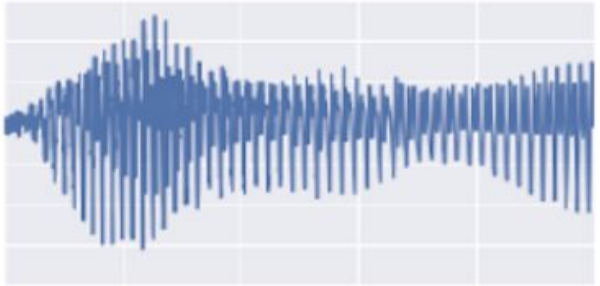


Text

Text synthesis (e.g., chatbots)

DEEP GENERATIVE MODELING: WHERE CAN WE USE IT?

Images

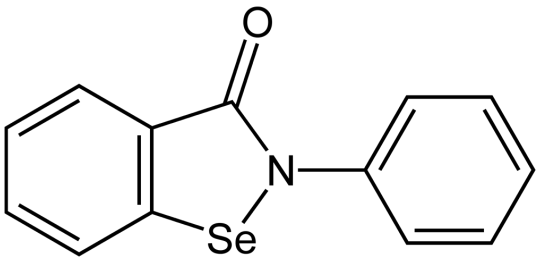


Audio

Speech synthesis (e.g., chatbots)

Deep fakes

DEEP GENERATIVE MODELING: WHERE CAN WE USE IT?



Graphs

Drug discovery

3D structure discovery

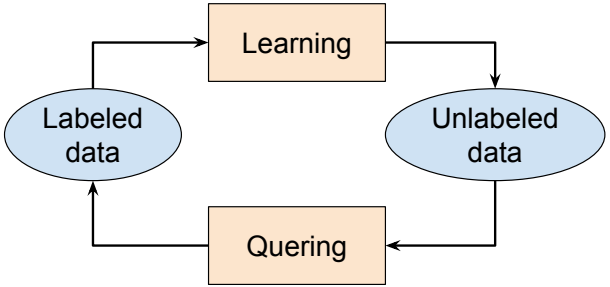
Molecular dynamics

DEEP GENERATIVE MODELING: WHERE CAN WE USE IT?

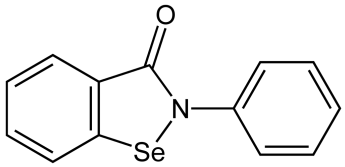
“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Text

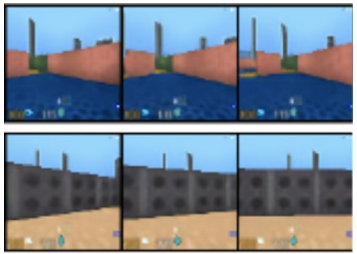
Images



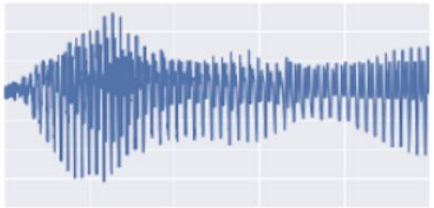
Active learning



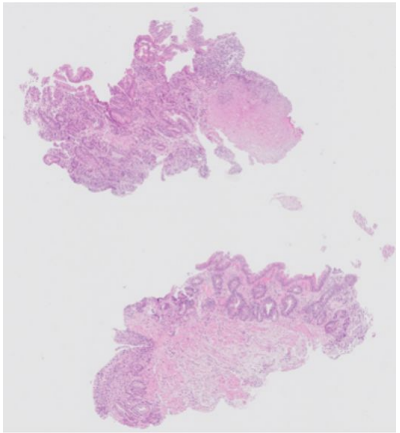
Graphs



Reinforcement learning



Audio



Medical imaging

BASIC RULES FOR DEEP GENERATIVE MODELING

BASIC RULES FOR DEEP GENERATIVE MODELING

Two rules of probability theory:

- **Sum rule**:

$$p(x) = \sum_{y} p(x, y)$$

- **Product rule**:

$$p(x, y) = p(x \mid y)p(y)$$

or

$$p(x, y) = p(y \mid x)p(x)$$

BASIC RULES FOR DEEP GENERATIVE MODELING

The objective (typically): **the log-likelihood function**

Given ***iid* data**: $\mathscr{D} = \{x_1, x_2, \ldots, x_N\}$

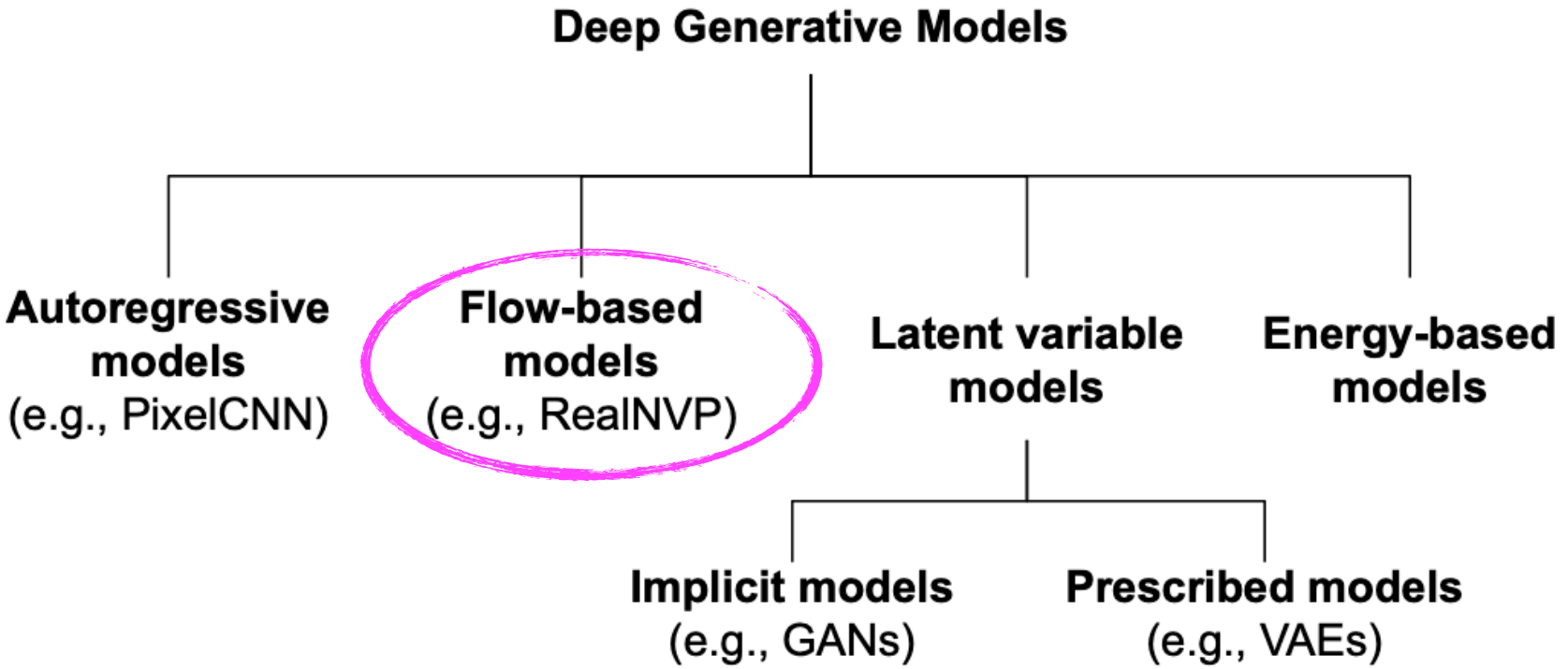
A **model** $p(x \mid \theta)$.

The log-likelihood function is:

$$\ln p(\mathscr{D} \mid \theta) = \ln \prod_{n=1}^{N} p(x_n \mid \theta)$$

$$= \sum_{n=1}^{N} \ln p(x_n \mid \theta)$$

DEEP GENERATIVE MODELING: HOW WE CAN FORMULATE IT?



FLOWS (FLOW-BASED MODELS)

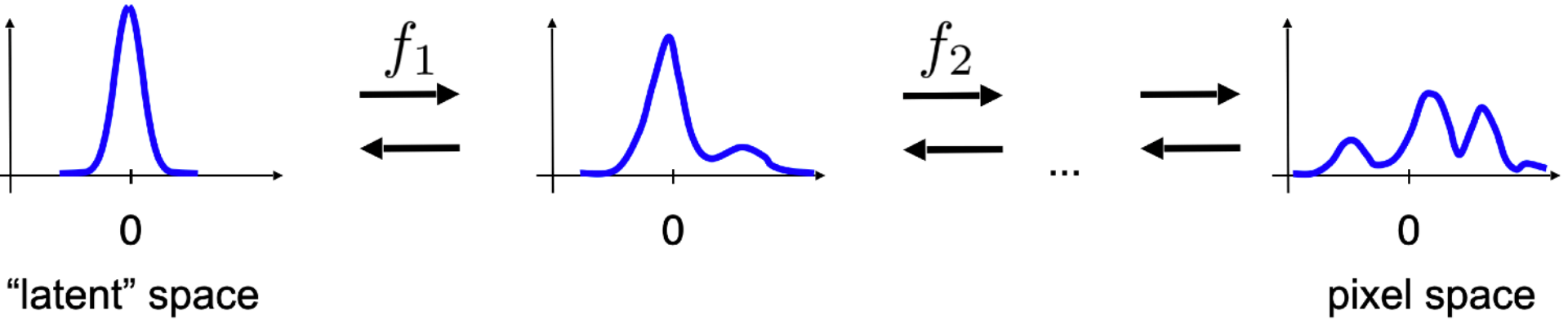
We change a random variable **x** to another random variable **z** using **invertible** transformations, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$:

$$p(\mathbf{x}) = \pi\left(\mathbf{z}_0 = f^{-1}(\mathbf{x})\right) \prod_{i=1}^{K} \left| \mathbf{J}_{f_i}(z_{i-1}) \right|^{-1}$$

FLOWS (FLOW-BASED MODELS)

We change a random variable **x** to another random variable **z** using **invertible** transformations, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$:

$$p(\mathbf{x}) = \pi\left(\mathbf{z}_0 = f^{-1}(\mathbf{x})\right) \prod_{i=1}^{K} \left| \mathbf{J}_{f_i}(z_{i-1}) \right|^{-1}$$



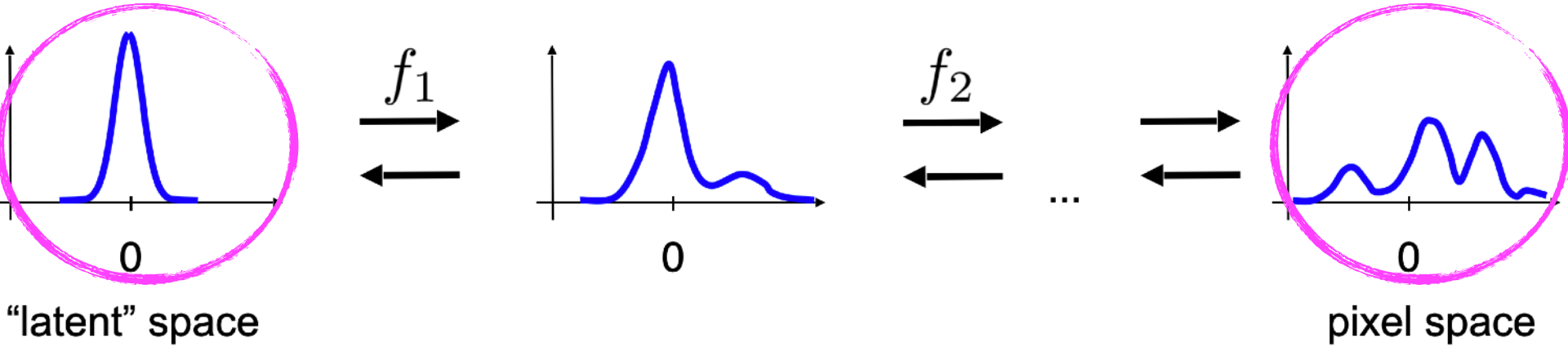
FLOWS (FLOW-BASED MODELS)

We change a random variable **x** to another random variable **z** using **invertible** transformations, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$:

$$p(\mathbf{x}) = \pi\left(\mathbf{z}_0 = f^{-1}(\mathbf{x})\right) \prod_{i=1}^{K} \left| \mathbf{J}_{f_i}(z_{i-1}) \right|^{-1}$$

Simple distribution

Complex distribution

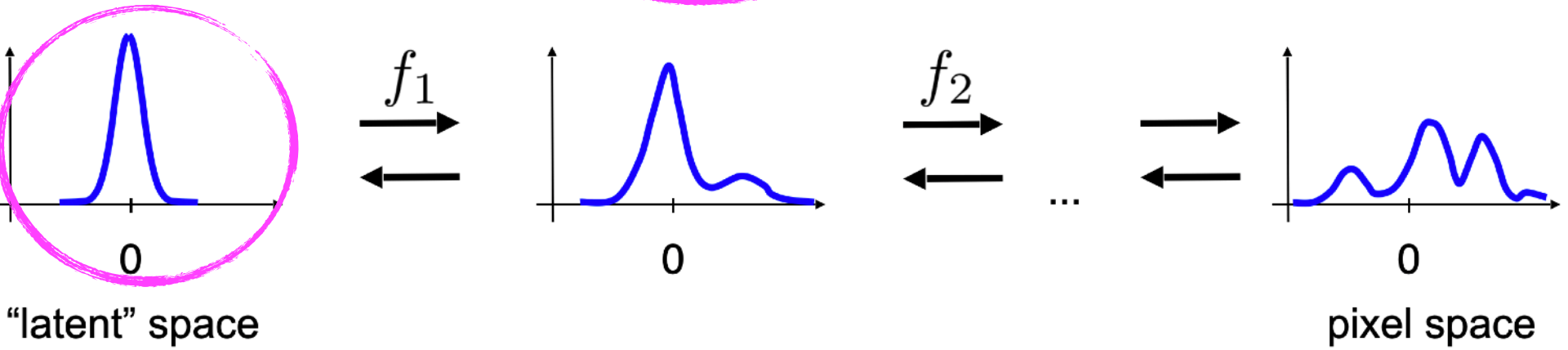


FLOWS (FLOW-BASED MODELS)

We change a random variable **x** to another random variable **z** using **invertible** transformations, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$:

Known, e.g., Gaussian

$$p(\mathbf{x}) = \pi\left(\mathbf{z}_0 = f^{-1}(\mathbf{x})\right) \prod_{i=1}^{K} \left| \mathbf{J}_{f_i}(z_{i-1}) \right|^{-1}$$

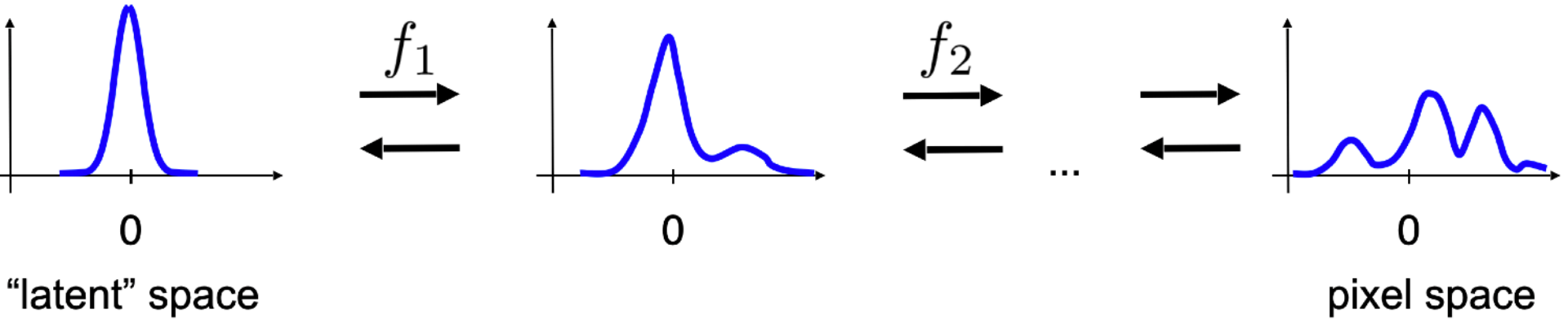


FLOWS (FLOW-BASED MODELS)

We change a random variable **x** to another random variable **z** using **invertible** transformations, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$:

Jacobian must be tractable

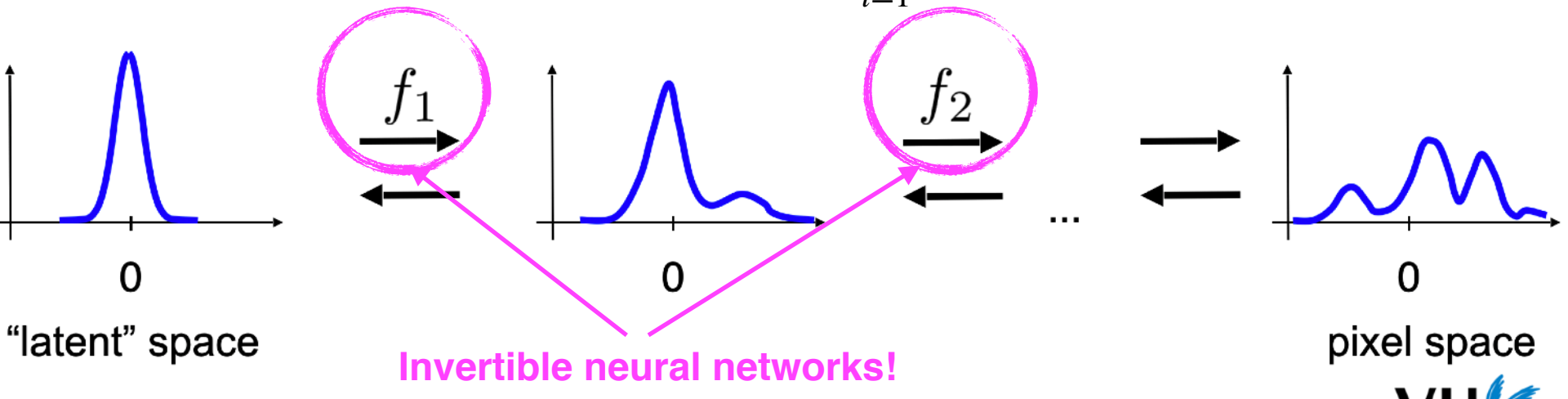
$$p(\mathbf{x}) = \pi\left(\mathbf{z}_0 = f^{-1}(\mathbf{x})\right) \prod_{i=1}^{K} \left| \mathbf{J}_{f_i}(z_{i-1}) \right|^{-1}$$



FLOWS (FLOW-BASED MODELS)

We change a random variable **x** to another random variable **z** using **invertible** transformations, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$:

$$p(\mathbf{x}) = \pi\left(\mathbf{z}_0 = f^{-1}(\mathbf{x})\right) \prod_{i=1}^{K} \left| \mathbf{J}_{f_i}(z_{i-1}) \right|^{-1}$$



We change a random variable **x** to another random variable **z** using **invertible** transformations, $\mathbf{x}, \mathbf{z} \in \mathbb{R}^D$:

$$p(\mathbf{x}) = \pi\left(\mathbf{z}_0 = f^{-1}(\mathbf{x})\right) \prod_{i=1}^{K} \left| \mathbf{J}_{f_i}(z_{i-1}) \right|^{-1}$$

Training objective:

$$\ln p(\mathbf{x}) = \ln \pi\left(\mathbf{z}_0 = f^{-1}(\mathbf{x})\right) - \sum_{i=1}^{K} \ln \left| \mathbf{J}_{f_i}(z_{i-1}) \right|$$

FLOWS (FLOW-BASED MODELS): INVERTIBLE LAYERS

Two main components

1) **Coupling layer**:

$$\mathbf{y}_a = \mathbf{x}_a$$
$$\mathbf{y}_b = \exp\left(s\left(\mathbf{x}_a\right)\right) \odot \mathbf{x}_b + t\left(\mathbf{x}_a\right)$$

is invertible by design:

$$\mathbf{x}_b = \left(\mathbf{y}_b - t(\mathbf{y}_a)\right) \odot \exp\left(-s(\mathbf{y}_a)\right)$$
$$\mathbf{x}_a = \mathbf{y}_a$$

2) **Permutation layer**

FLOWS (FLOW-BASED MODELS): INVERTIBLE LAYERS

Two main components

1) **Coupling layer**:

$$\mathbf{y}_a = \mathbf{x}_a$$
$$\mathbf{y}_b = \exp\left(s\left(\mathbf{x}_a\right)\right) \odot \mathbf{x}_b + t\left(\mathbf{x}_a\right)$$

is invertible by design:

$$\mathbf{x}_b = \left(\mathbf{y}_b - t(\mathbf{y}_a)\right) \odot \exp\left(-s(\mathbf{y}_a)\right)$$
$$\mathbf{x}_a = \mathbf{y}_a$$

Jacobian is tractable!

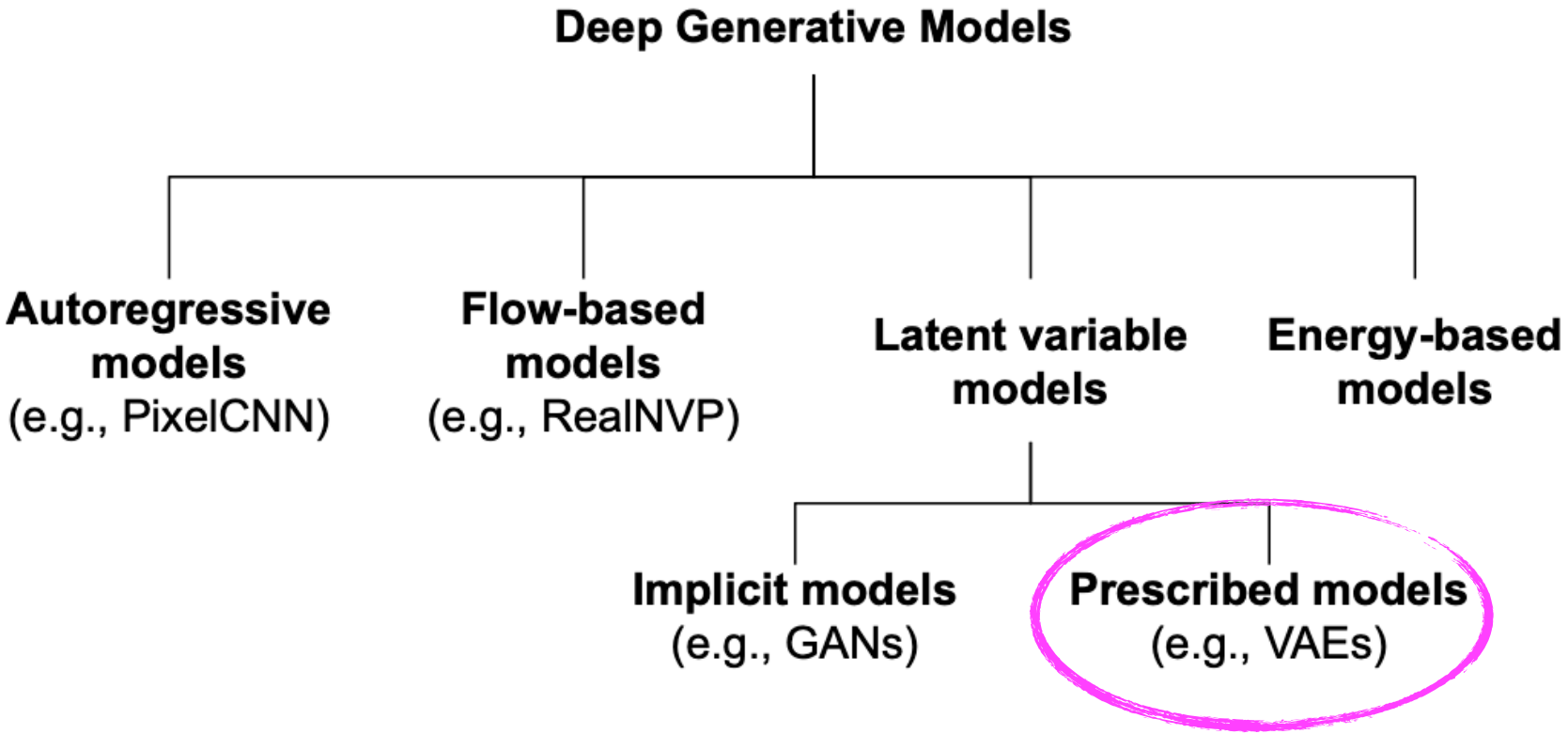
$$\det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp\left(s\left(\mathbf{x}_a\right)\right)_j = \exp\left(\sum_{j=1}^{D-d} s\left(\mathbf{x}_a\right)_j\right)$$

2) **Permutation layer** $\det(\mathbf{J}) = 1$

FLOWS (FLOW-BASED MODELS)

Kingma, D.P., and Prafulla D. "Glow: Generative flow with invertible 1x1 convolutions." *NeurIPS, 2018*

DEEP GENERATIVE MODELING: HOW WE CAN FORMULATE IT?

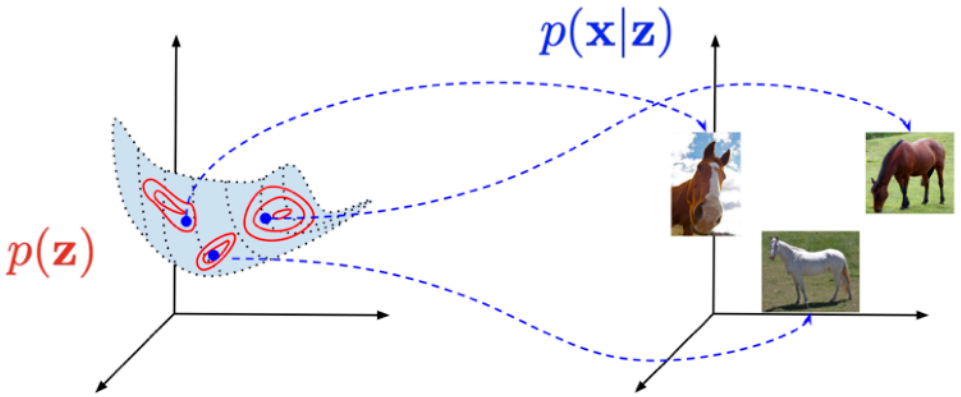


VARIATIONAL AUTO-ENCODERS

Let's consider a **latent variable model** where we distinguish:

- **latent variables** $\mathbf{z} \in \mathcal{Z}^M$
- **observable variables** $\mathbf{x} \in \mathcal{X}^D$

Latent variables lie on a **low-dimensional manifold**.



Generative process:

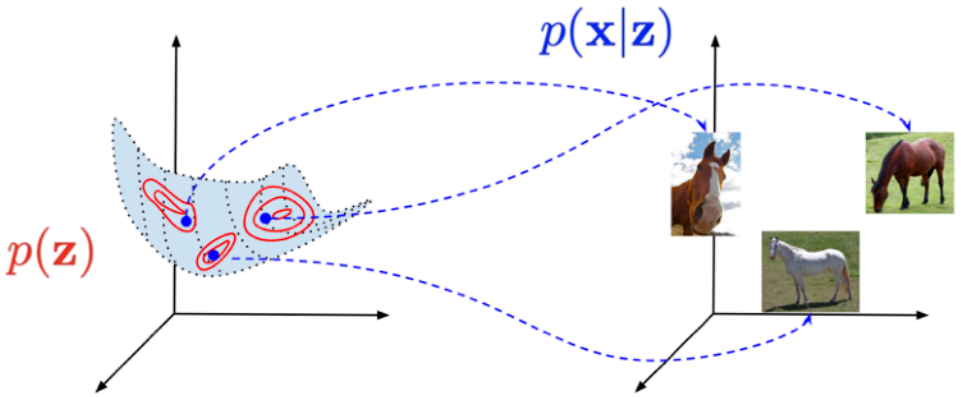
1. $\mathbf{z} \sim p(\mathbf{z})$
2. $\mathbf{x} \sim p(\mathbf{x}|\mathbf{z})$

VARIATIONAL AUTO-ENCODERS

Let's consider a **latent variable model** where we distinguish:

- **latent variables** $\mathbf{z} \in \mathcal{Z}^M$
- **observable variables** $\mathbf{x} \in \mathcal{X}^D$

Latent variables lie on a **low-dimensional manifold**.



Generative process:

1. $\mathbf{z} \sim p(\mathbf{z})$
2. $\mathbf{x} \sim p(\mathbf{x}\,|\,\mathbf{z})$

The objective function:

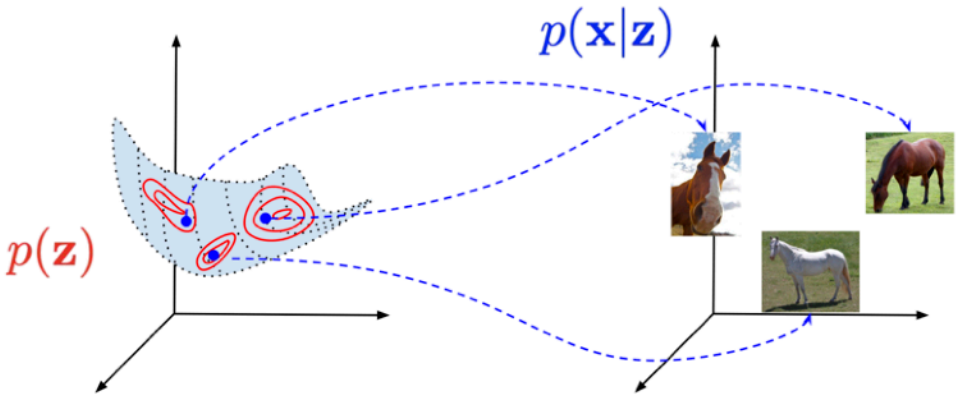
$$\ln p(\mathbf{x}) = \ln \int p(\mathbf{x}\,|\,\mathbf{z}) p(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

VARIATIONAL AUTO-ENCODERS

Let's consider a **latent variable model** where we distinguish:

- **latent variables** $\mathbf{z} \in \mathcal{Z}^M$
- **observable variables** $\mathbf{x} \in \mathcal{X}^D$

Latent variables lie on a **low-dimensional manifold**.



Generative process:

1. $\mathbf{z} \sim p(\mathbf{z})$
2. $\mathbf{x} \sim p(\mathbf{x}\,|\,\mathbf{z})$

The objective function:

$$\ln p(\mathbf{x}) = \ln \int p(\mathbf{x}\,|\,\mathbf{z}) p(\mathbf{z})\ \mathrm{d}\mathbf{z}$$

The integral is intractable…

VARIATIONAL AUTO-ENCODERS

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p(\mathbf{x}\,|\,\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln q_\phi(\mathbf{z}\,|\,\mathbf{x}) - \ln p(\mathbf{z})\right]$$

ELBO: Evidence Lower Bound

 Kingma, D.P., and Welling, M.. "Auto-encoding variational bayes." *ICLR 2014*

VARIATIONAL AUTO-ENCODERS

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p(\mathbf{x}\,|\,\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln q_\phi(\mathbf{z}\,|\,\mathbf{x}) - \ln p(\mathbf{z})\right]$$

We consider **amortized inference**: $q_\phi(\mathbf{z}\,|\,\mathbf{x})$

In other words, a single parameterization for each new input **x**.

Kingma, D.P., and Welling, M.. "Auto-encoding variational bayes." *ICLR 2014*

VARIATIONAL AUTO-ENCODERS

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p(\mathbf{x}\,|\,\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln q_\phi(\mathbf{z}\,|\,\mathbf{x}) - \ln p(\mathbf{z})\right]$$

We consider **amortized inference**: $q_\phi(\mathbf{z}\,|\,\mathbf{x})$

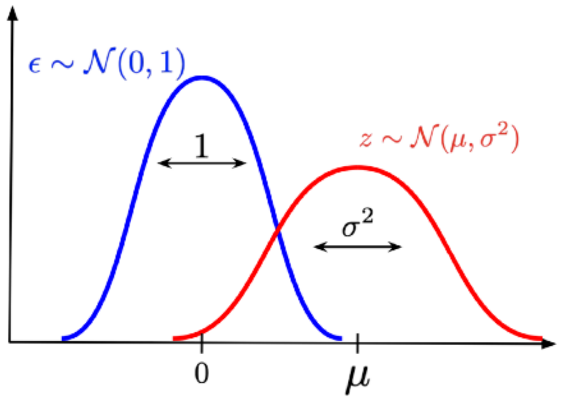
In other words, a single parameterization for each new input **x**.

Moreover, we use **reparameterization trick**:

Every Gaussian variable could be defined as:

$z = \mu + \sigma \cdot \varepsilon$

where $\varepsilon \sim \mathcal{N}(0,1)$



 Kingma, D.P., and Welling, M.. "Auto-encoding variational bayes." *ICLR 2014*

VARIATIONAL AUTO-ENCODERS

$$\ln p(\mathbf{x}) \geq \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln p(\mathbf{x}\,|\,\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\ln q_\phi(\mathbf{z}\,|\,\mathbf{x}) - \ln p(\mathbf{z})\right]$$

We consider **amortized inference**: $q_\phi(\mathbf{z}\,|\,\mathbf{x})$

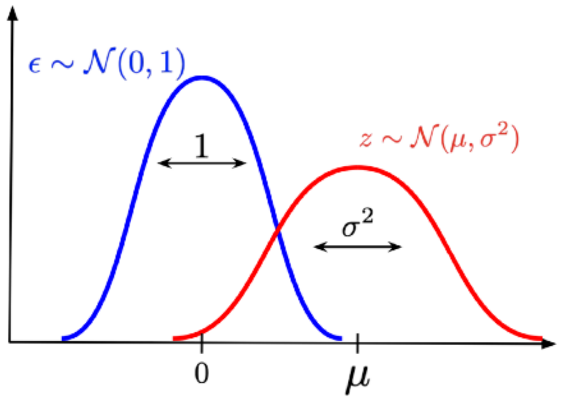
In other words, a single parameterization for each new input **x**.

Moreover, we use **reparameterization trick**:

It reduces the variance of the gradients.

It allows to get randomness outside **z**.

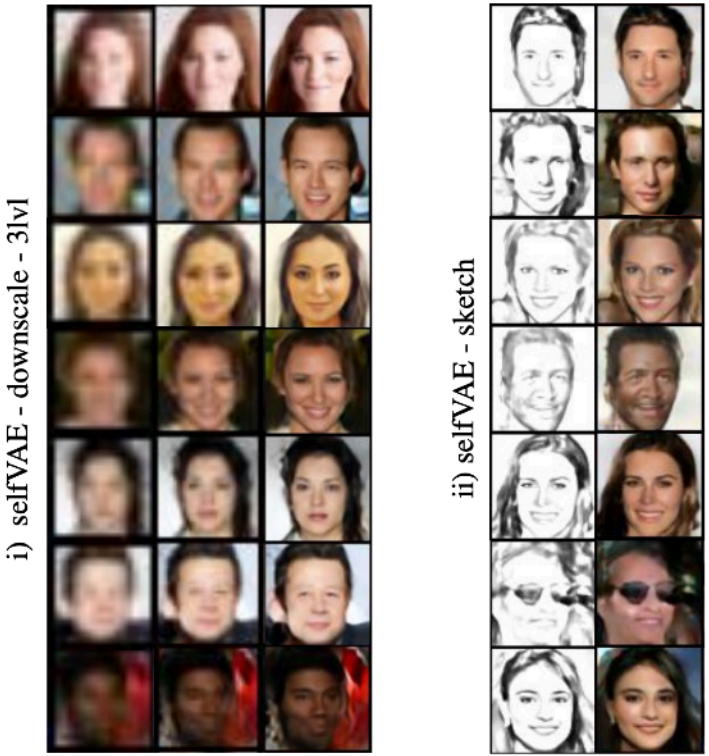
$z = \mu + \sigma \cdot \varepsilon$



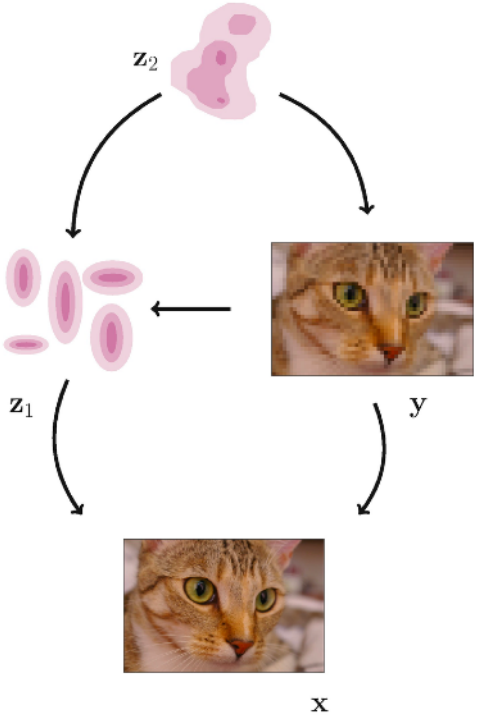
 Kingma, D.P., and Welling, M.. "Auto-encoding variational bayes." *ICLR 2014*

VU

HIERARCHICAL VARIATIONAL AUTO-ENCODERS



Generations

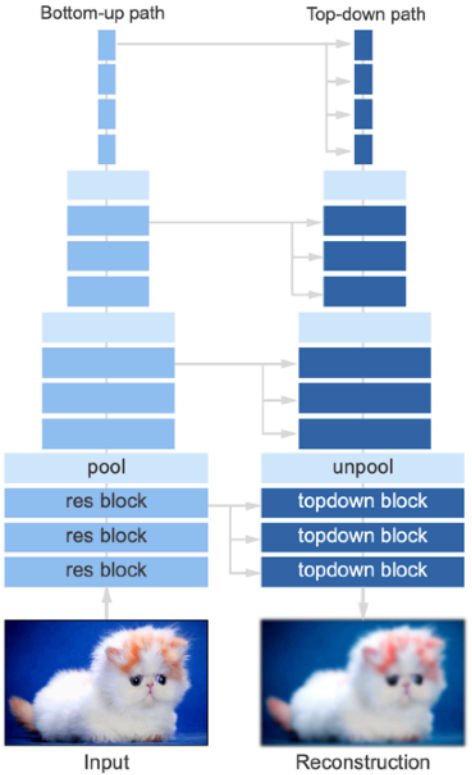


Hierarchical VAE

 Gatopoulos, I., and Tomczak, J.M., "Self-Supervised Variational Auto-Encoders." *Entropy* (2021).

TOP-DOWN VARIATIONAL AUTO-ENCODERS

Generations



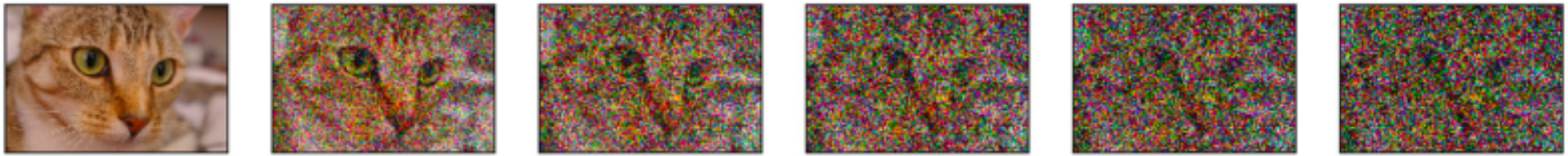
Very Deep VAE

 Child, R. "Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images." *ICLR 2021*

DIFFUSION-BASED DEEP GENERATIVE MODELS

$$q_\phi(\mathbf{z}_i \mid \mathbf{z}_{i-1}) = \mathcal{N}\left(\mathbf{z}_i \mid \sqrt{1-\beta_i}\mathbf{z}_{i-1}, \beta_i \mathbf{I}\right)$$

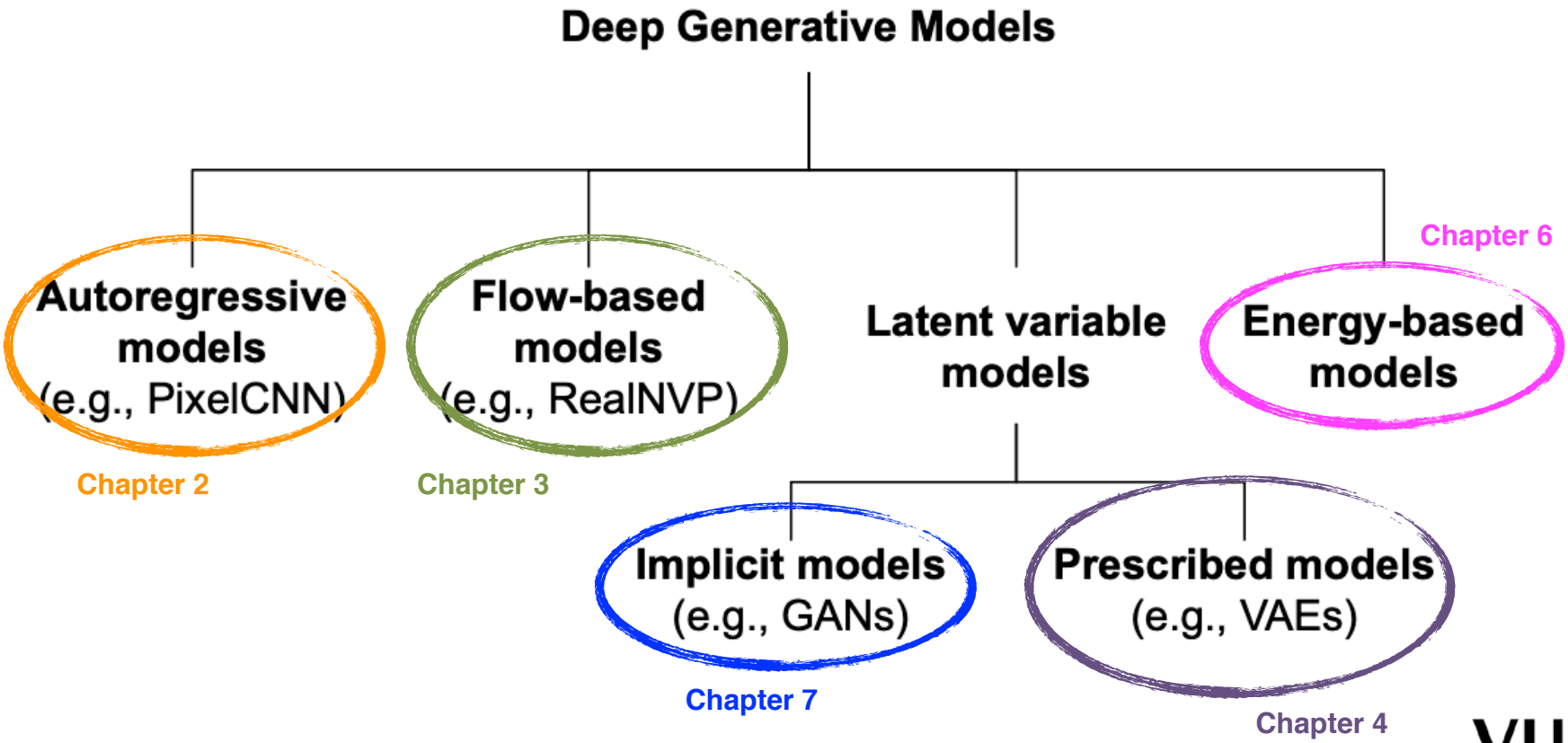
Forward diffusion (FIXED!)



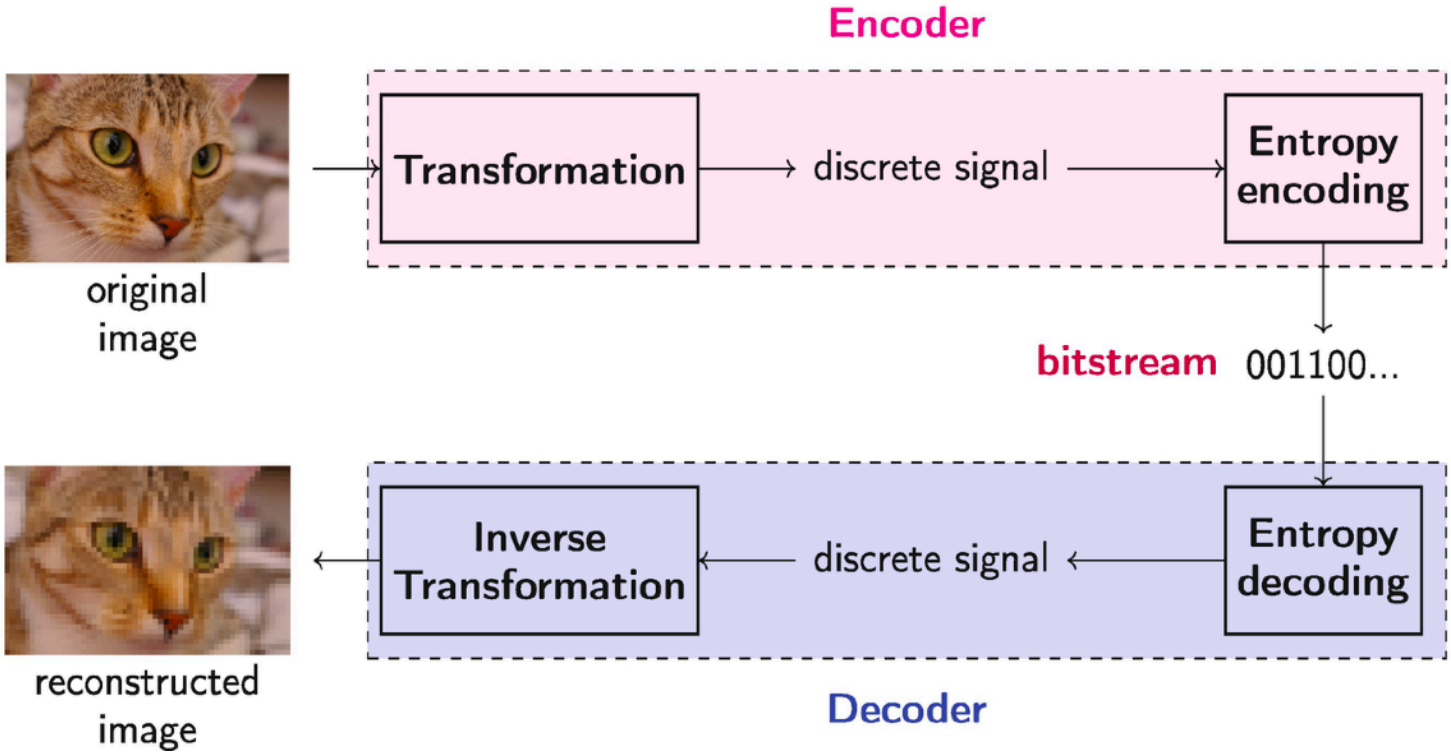
Backward diffusion (learnable)

Sohl-Dickstein J., Weiss E., Maheswaranathan N., & Ganguli S.. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning 2015*, pages 2256–2265

DEEP GENERATIVE MODELING: HOW WE CAN FORMULATE IT?

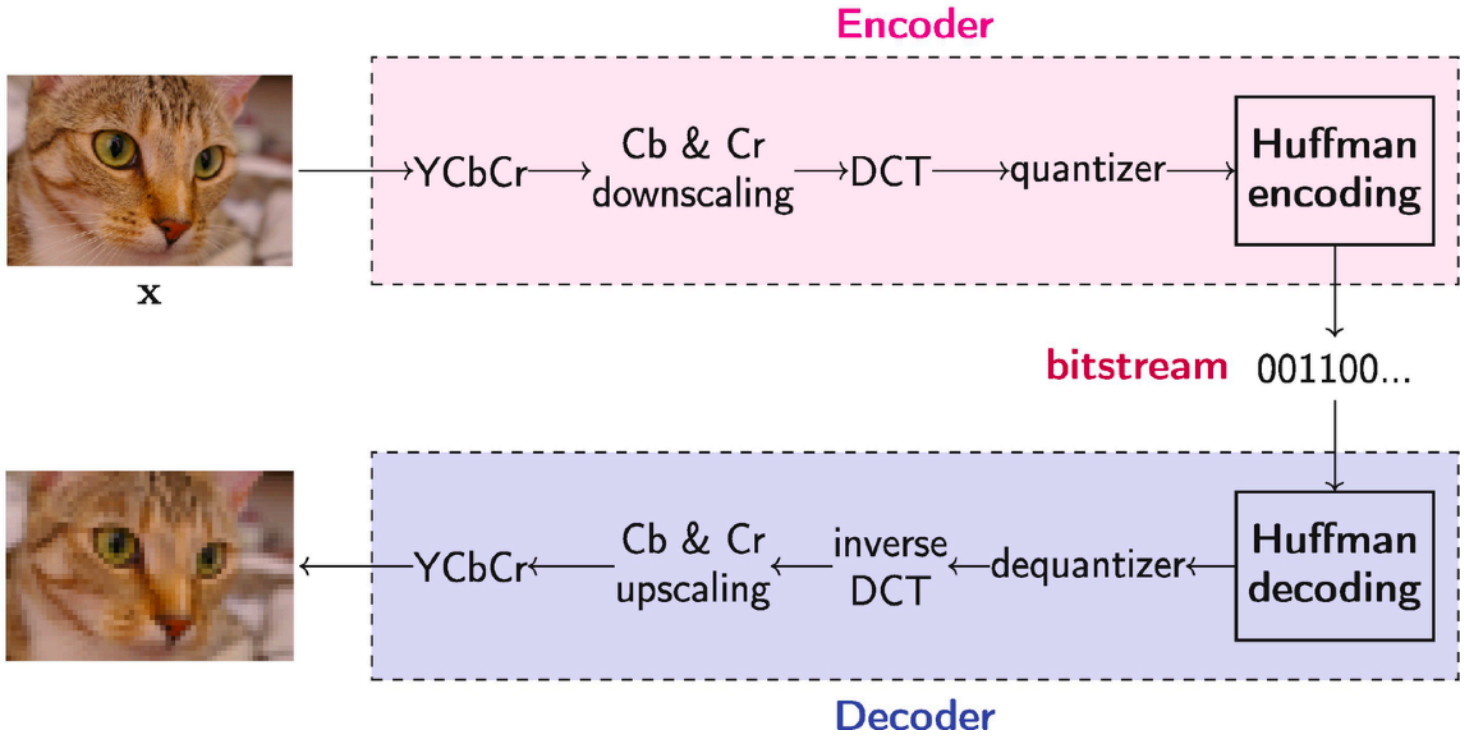


APPLICATIONS BEYOND DATA SYNTHESIS: NEURAL COMPRESSION



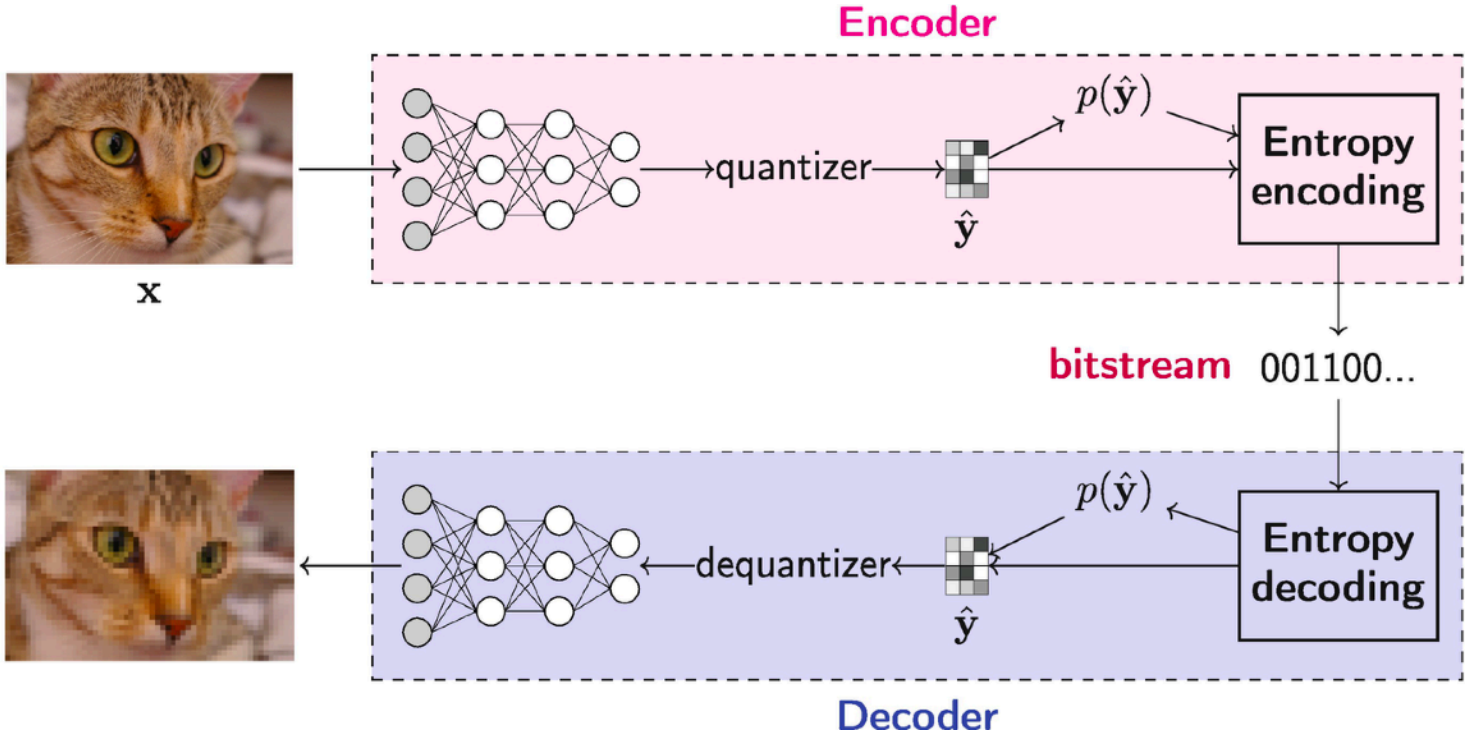
Standard compression scheme

APPLICATIONS BEYOND DATA SYNTHESIS: NEURAL COMPRESSION



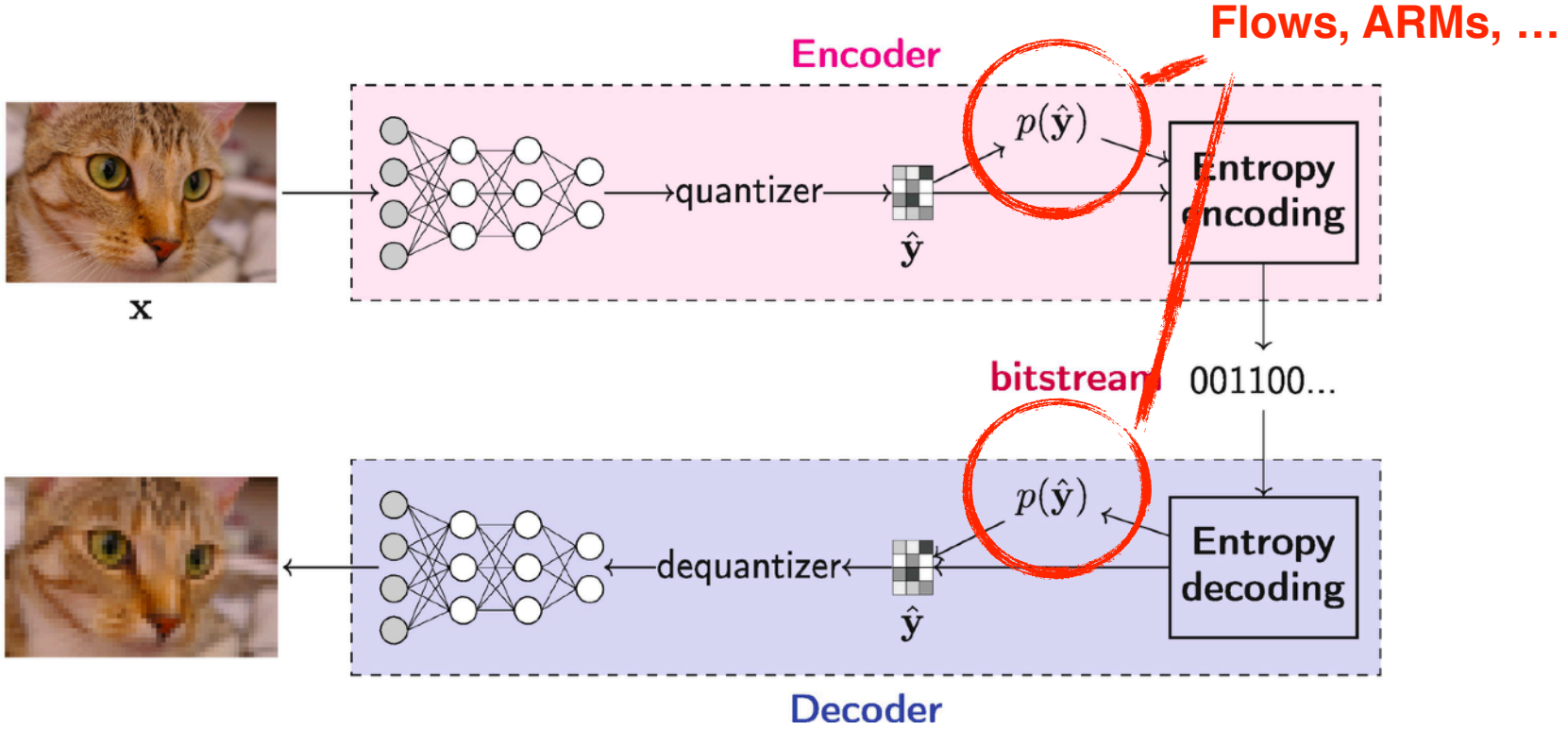
JPEG compression scheme

APPLICATIONS BEYOND DATA SYNTHESIS: NEURAL COMPRESSION



Neural compression scheme

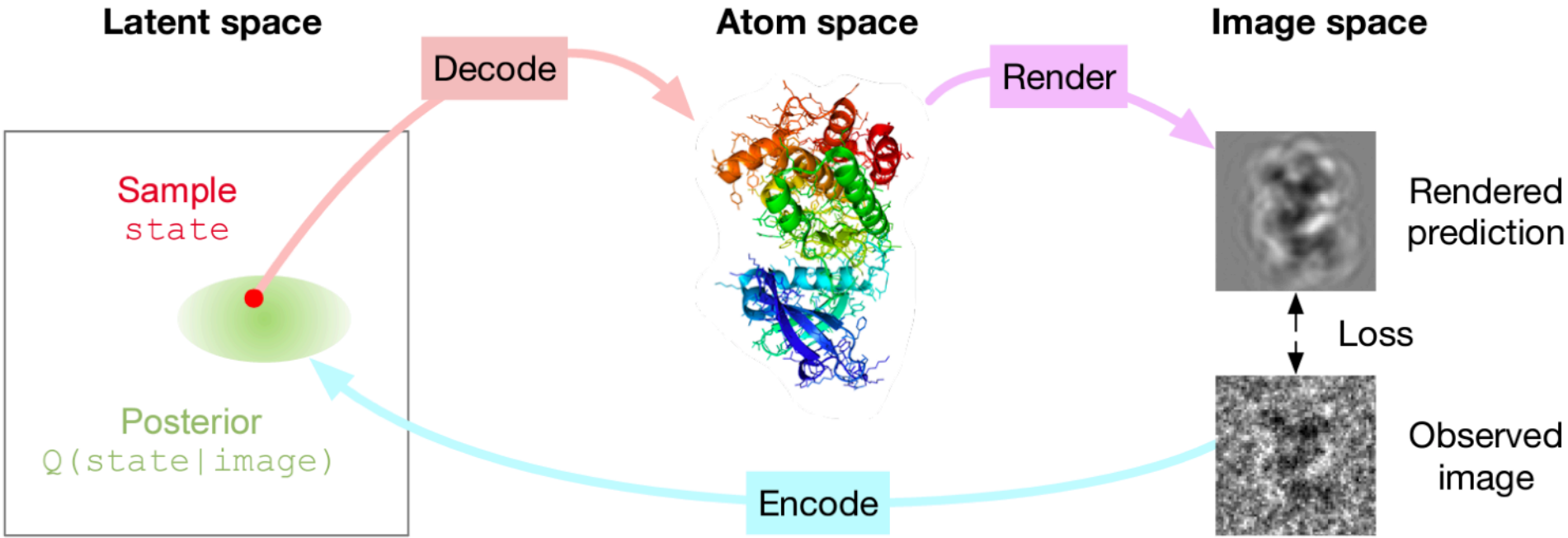
APPLICATIONS BEYOND DATA SYNTHESIS: NEURAL COMPRESSION



Neural compression scheme

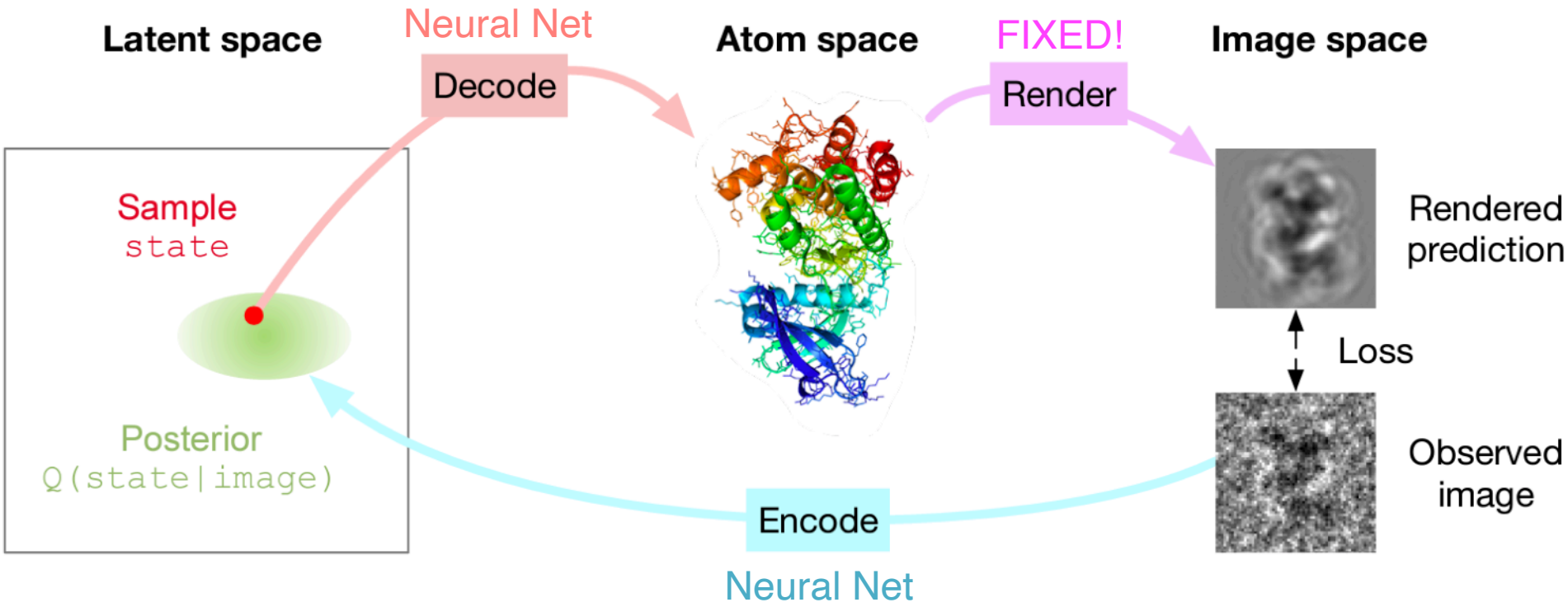
Tomczak J.M., Deep Generative Modeling, *Chapter 8,* Springer, 2022

APPLICATIONS BEYOND DATA SYNTHESIS: MOLECULAR MODELING



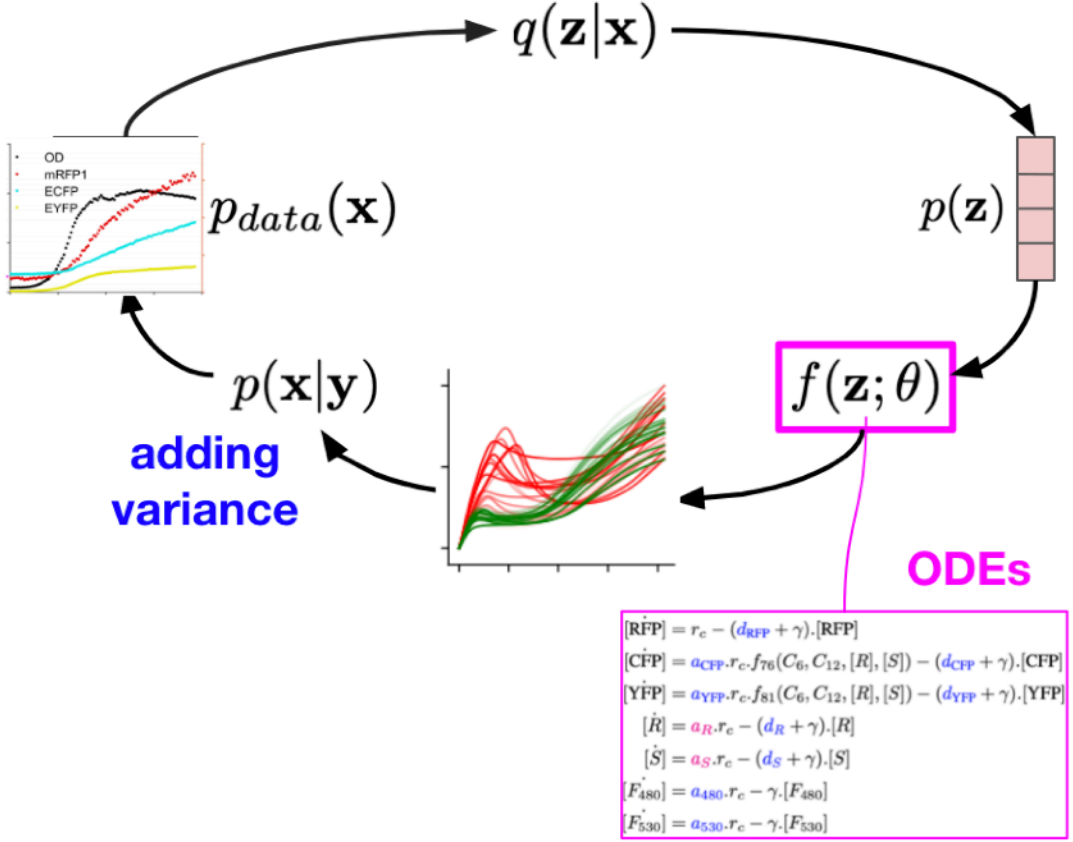
Rosenbaum, Dan, et al. Inferring a Continuous Distribution of Atom Coordinates from Cryo-EM Images using VAEs. arXiv preprint arXiv:2106.14108, June 26, 2021

APPLICATIONS BEYOND DATA SYNTHESIS: MOLECULAR MODELING



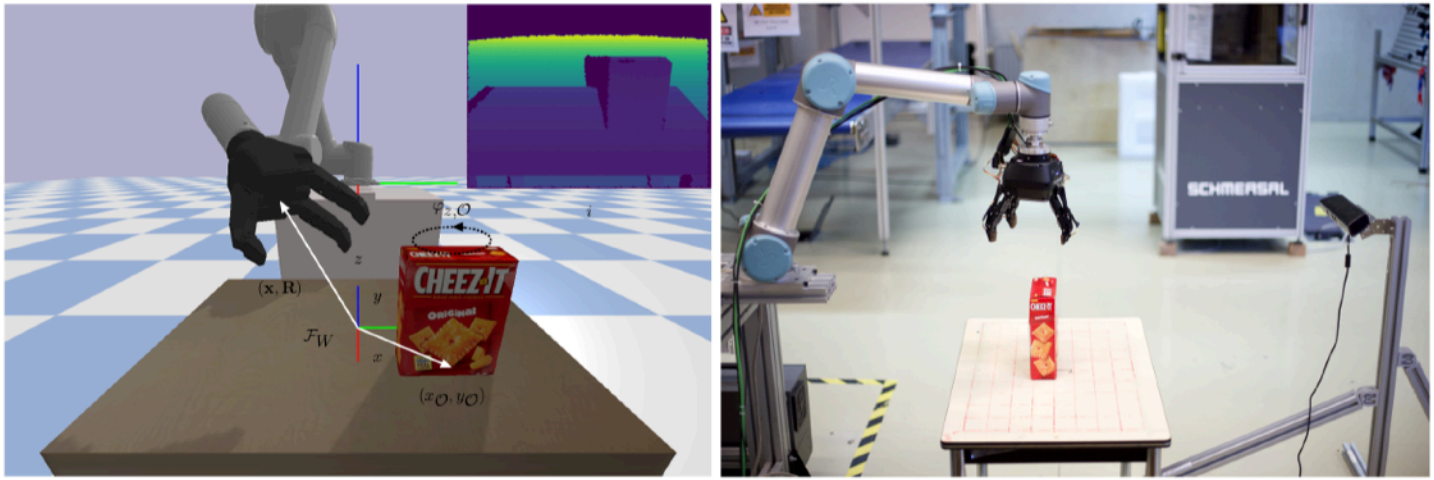
Rosenbaum, Dan, et al. Inferring a Continuous Distribution of Atom Coordinates from Cryo-EM Images using VAEs. arXiv preprint arXiv:2106.14108, June 26, 2021

APPLICATIONS BEYOND DATA SYNTHESIS: GENETICS



Meeds, T., Roeder, G., Grant, P., Phillips, A., & Dalchau, N. Efficient amortised bayesian inference for hierarchical and nonlinear dynamical systems. *ICML 2019*

APPLICATIONS BEYOND DATA SYNTHESIS: ROBOT LEARNING



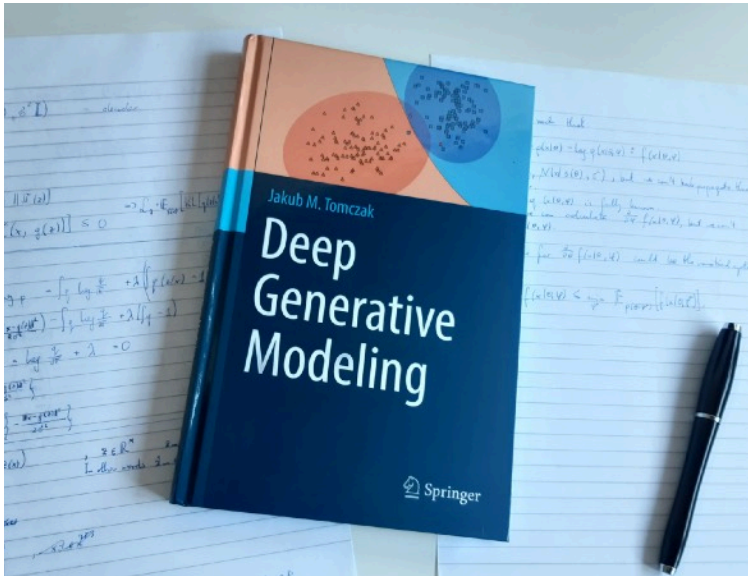
Generative model for generating trajectories.

Marlier, N., Brüls, O., & Louppe, G. (2021). Simulation-based Bayesian inference for multi-fingered robotic grasping. *arXiv preprint arXiv:2109.14275*.

MORE ABOUT DEEP GENERATIVE MODELING

Blog: https://jmtomczak.github.io/blog.html

Book: https://link.springer.com/book/10.1007/978-3-030-93158-2



CONCLUSION

- Why generative modeling?

$$p(\mathbf{x}, y) = p(y \mid \mathbf{x})\, p(\mathbf{x})$$

CONCLUSION

- Why generative modeling?

$$p(\mathbf{x}, y) = p(y \mid \mathbf{x})\, p(\mathbf{x})$$

- How to model the joint?

  ➡ Hybrid modeling (Chapter 5)

CONCLUSION

- Why generative modeling?

$$p(\mathbf{x}, y) = p(y \mid \mathbf{x})\, p(\mathbf{x})$$

- How to model the joint?
  - ➡ Hybrid modeling (Chapter 5)
- Important directions:
  - ➡ Better uncertainty quantification
  - ➡ New parameterization (new neural networks)
  - ➡ Out-of-Distribution
  - ➡ Continual learning

THANK YOU FOR YOUR ATTENTION

Jakub M. Tomczak
Computational Intelligence group
Vrije Universiteit Amsterdam

Webpage: https://jmtomczak.github.io/

Github: https://github.com/jmtomczak

Twitter: https://twitter.com/jmtomczak