# Is the Likelihood-based Deep Generative Modeling appropriate for Representation Learning?

Jakub M. Tomczak

Copenhagen, October 12-13, 2021

VU VRIJE UNIVERSITEIT AMSTERDAM

# Representation learning

# Representation learning

What is data representation [1]:

*"A **good representation** is (...) one that is **useful** **as input to a supervised predictor**."*



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What is data representation [1]:

*"A **good representation** is (...) one that is **useful** **as input to a supervised predictor**."*

$$\mathbf{z} = f_\theta(\mathbf{x})$$
$$y = f_\varphi(\mathbf{z})$$

**z** is useful for the predictor
= low error (?)

what about unsupervised learning?



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What is data representation [1]:

*"A **good representation** is (...) one that is **useful** **as input to a supervised predictor**."*

$$\mathbf{z} = f_\theta(\mathbf{x})$$
$$y = f_\varphi(\mathbf{z})$$

**z** is useful for the predictor
= low error (?)

or, in the context of probabilistic modeling:

what about unsupervised learning?

*"(...) a **good representation** is often one **that captures the posterior distribution of the underlying explanatory factors** for the observed input."*



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What is data representation [1]:

*"A **good representation** is (...) one that is **useful** **as input to a supervised predictor**."*

$$\mathbf{z} = f_\theta(\mathbf{x})$$
$$y = f_\varphi(\mathbf{z})$$

**z** is useful for the predictor
= low error (?)

or, in the context of probabilistic modeling:

what about unsupervised learning?

*"(...) a **good representation** is often one **that captures the posterior distribution of the underlying explanatory factors** for the observed input."*

$$q_\phi(\mathbf{z}|\mathbf{x})$$

**q**(**z**|**x**) is non-trivial (?), e.g., N(0, **1**)



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What makes a representation good (guidelines) [1]:

- smoothness: x ≈ y implies f(x) ≈ f(y)
- multiple explanatory factors
- a hierarchy of explanatory factors
- spatial/temporal coherence
- sparsity
- factors explain x and help to predict *y*
- geometrical properties (manifolds)
- generalizability (shared factors across tasks)
- simplicity of factor dependencies



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What makes a representation good (guidelines) [1]:

| ***Implicit*** (**model** inductive biases) | ***Explicit*** (**objective** inductive biases) |
|---|---|
| • smoothness: x ≈ y implies f(x) ≈ f(y)<br>• multiple explanatory factors<br>• a hierarchy of explanatory factors<br>• geometrical properties (manifolds)<br>• factors explain x and help to predict y<br>• (?) spatial/temporal coherence<br>• (?) simplicity of factor dependencies | • smoothness: x ≈ y implies f(x) ≈ f(y)<br>• sparsity<br>• geometrical properties (manifolds)<br>• (?) spatial/temporal coherence<br>• (?) generalizability (shared factors across tasks) |



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What makes a representation good (guidelines) [1]:

| ***Implicit*** (**model** inductive biases) | ***Explicit*** (**objective** inductive biases) |
|---|---|
| • smoothness: x ≈ y implies f(x) ≈ f(y)<br>• multiple explanatory factors<br>• a hierarchy of explanatory factors<br>• geometrical properties (manifolds)<br>• factors explain x and help to predict y<br>• (?) spatial/temporal coherence<br>• (?) simplicity of factor dependencies | • smoothness: x ≈ y implies f(x) ≈ f(y)<br>• sparsity<br>• geometrical properties (manifolds)<br>• (?) spatial/temporal coherence<br>• (?) generalizability (shared factors across tasks) |

**What about compression?**



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

VU

# Representation learning

What makes a representation good (guidelines) [1]:

| *Implicit* (**model** inductive biases) | *Explicit* (**objective** inductive biases) |
| --- | --- |
| • smoothness: x | plies f(x) ≈ f(y) |
| • multiple expla | |
| • a hierarchy of | es (manifolds) |
| • geometrical pr | coherence |
| • factors explain | nared factors |
| • (?) spatial/tem | across tasks) |
| • (?) simplicity of factor dependencies | |

**Is there a single framework for that?**



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What makes a representation good (guidelines) [1]:

**Is there a single framework for that?**

**Probabilistic modeling +**



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What makes a representation good (guidelines) [1]:

**Is there a single framework for that?**

**Probabilistic modeling + Deep Learning**

**flexible parameterization of distributions**



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

# Representation learning

What makes a representation good (guidelines) [1]:

| *Implicit* (**model** inductive biases) | *Explicit* (**objective** inductive biases) |
| --- | --- |
| • smoothness: x | plies f(x) ≈ f(y) |
| • multiple expla | |
| • a hierarchy of | es (manifolds) |
| • geometrical pr | coherence |
| • factors explain | nared factors |
| • (?) spatial/tem | across tasks) |
| • (?) simplicity of factor dependencies | |

**Is there a single framework for that?**

**Probabilistic modeling + Deep Learning**

**=**

**Deep Generative Modeling**



[1] Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. IEEE TPAMI, 35(8), 1798-1828.

VU

# Deep Generative Modeling

# Deep Generative Modeling

After [2]:

**Generative modeling**: formulating a model of the joint distribution, $p_\vartheta(\mathbf{x}, y)$

**Discriminative modeling**: formulating a model of the conditional, $p_\vartheta(y|\mathbf{x})$


[2] Jebara, T. (2012). Machine learning: discriminative and generative (Vol. 755). Springer Science & Business Media.

# Deep Generative Modeling

After [2]:

**Generative modeling**: formulating a model of the joint distribution, $p_\vartheta(\mathbf{x}, y)$

**Discriminative modeling**: formulating a model of the conditional, $p_\vartheta(y|\mathbf{x})$

Why do we need **generative modeling**?

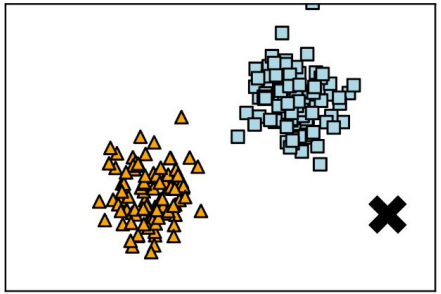

[2] Jebara, T. (2012). Machine learning: discriminative and generative (Vol. 755). Springer Science & Business Media.

# Deep Generative Modeling

After [2]:

**Generative modeling**: formulating a model of the joint distribution, $p_\vartheta(\mathbf{x}, y)$

**Discriminative modeling**: formulating a model of the conditional, $p_\vartheta(y|\mathbf{x})$



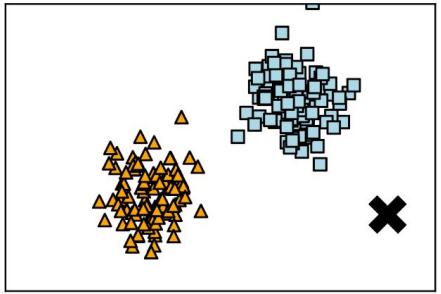
Data

# Deep Generative Modeling

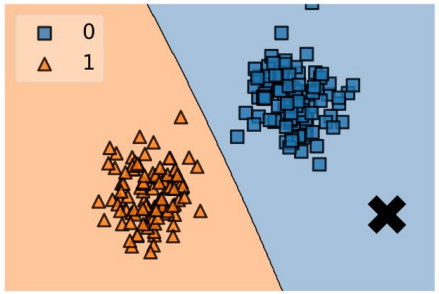
After [2]:

**Generative modeling**: formulating a model of the joint distribution, $p_\vartheta(\mathbf{x}, y)$

**Discriminative modeling**: formulating a model of the conditional, $p_\vartheta(y|\mathbf{x})$



Data




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

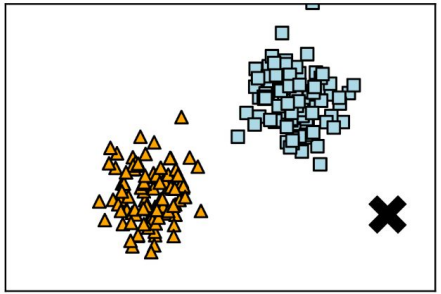
$p(blue|\mathbf{x})$ is high
= certain decision!

# Deep Generative Modeling

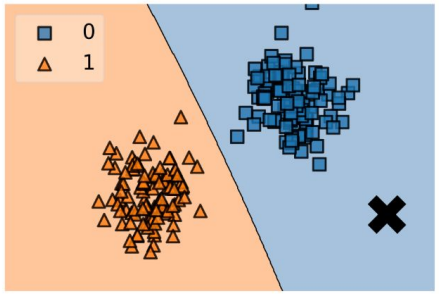
After [2]:

**Generative modeling**: formulating a model of the joint distribution, $p_\vartheta(\mathbf{x}, y)$

**Discriminative modeling**: formulating a model of the conditional, $p_\vartheta(y|\mathbf{x})$

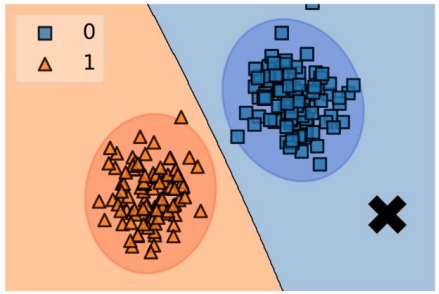


Data




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

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




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

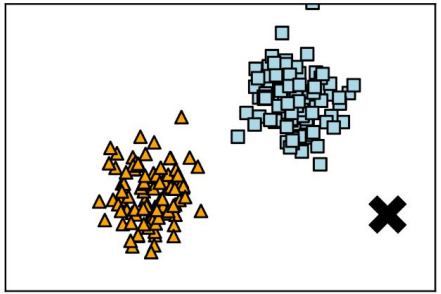
$p(blue|\mathbf{x})$ is high
and $p(\mathbf{x})$ is low
= uncertain decision!

# Deep Generative Modeling

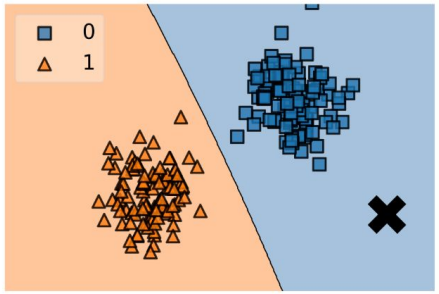
After [2]:

**Generative modeling**: formulating a model of the joint distribution, $p_\vartheta(\mathbf{x}, y)$

**Discriminative modeling**: formulating a model of the conditional, $p_\vartheta(y|\mathbf{x})$

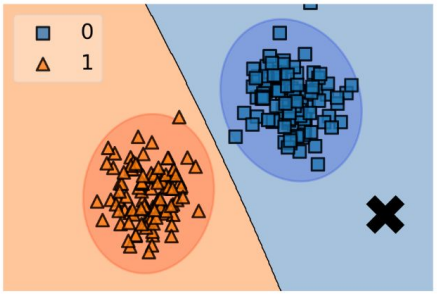


Data




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

$p(blue|\mathbf{x})$ 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!

**Considering the joint could be crucial for decision making!**

# Deep Generative Modeling

We consider the joint distribution factorized as follows:

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

Then the log-**likelihood** function (assuming *iid*) takes the following form:

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

# Deep Generative Modeling

We consider the joint distribution factorized as follows:

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

Then the log-**likelihood** function (assuming *iid*) takes the following form:

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

**Parameterized by deep neural nets**

**Various models:**
**→ Deep Latent Variable Models (e.g., VAEs, diffusion models)**
**→ Autoregressive Models (ARMs)**
**→ Energy-based Models (EBMs)**

# Deep Generative Modeling

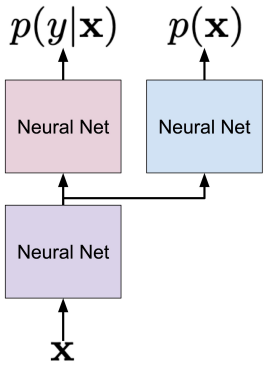
We consider the joint distribution factorized as follows:

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

Then the log-**likelihood** function (assuming *iid*) takes the following form:

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

**We can use a shared parameterization.**



# Deep Generative Modeling

We consider the joint distribution factorized as follows:

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

Then the log-**likelihood** function (assuming *iid*) takes the following form:

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

**1. Are there any problems with this objective?**

**2. Is this objective appropriate for representation learning?**

# The likelihood function

# Problems with the likelihood function

Let us take a look at the objective again (the log-likelihood function):

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

# Problems with the likelihood function

Let us take a look at the objective again (the log-likelihood function):

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

The **conditional** log-likelihood function is typically **fine**!

# Problems with the likelihood function

Let us take a look at the objective again (the log-likelihood function):

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

The conditional log-likelihood function is typically **fine**!

A potential danger with the marginal log-likelihood function:

# Problems with the likelihood function

Let us take a look at the objective again (the log-likelihood function):

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

The conditional log-likelihood function is typically **fine**!

A potential danger with the marginal log-likelihood function:

$\longrightarrow$ Consider a latent variable model: $p_\vartheta(\mathbf{x}) = \int p_\vartheta(\mathbf{x}|\mathbf{z})\ p_\vartheta(\mathbf{z})\ \mathrm{d}\mathbf{z}$

**infinite mixture model**

# Problems with the likelihood function

Let us take a look at the objective again (the log-likelihood function):

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

The conditional log-likelihood function is typically **fine**!

A potential danger with the marginal log-likelihood function:

→ Consider a latent variable model: $p_\vartheta(\mathbf{x}) = \int p_\vartheta(\mathbf{x}|\mathbf{z})\ p_\vartheta(\mathbf{z})\ \mathrm{d}\mathbf{z}$

→ If we take Gaussians, then we get an **unbounded likelihood** [3]!

**Solution: add a constant diagonal matrix to the decoder's covariance matrix.**


[3] Mattei, P. A., & Frellsen, J. (2018). Leveraging the exact likelihood of deep latent variable models. NeurIPS (pp. 3859-3870).

# Problems with the likelihood function

Let us take a look at the objective again (the log-likelihood function):

$$\ln p_\vartheta(\mathcal{D}) = \sum_n \ln p_\vartheta(y_n|\mathbf{x}_n) + \sum_n \ln p_\vartheta(\mathbf{x}_n)$$

The conditional log-likelihood function is typically **fine**!

A potential danger with the marginal log-likelihood function:

→ Consider a latent variable model: $p_\vartheta(\mathbf{x}) = \int p_\vartheta(\mathbf{x}|\mathbf{z})\ p_\vartheta(\mathbf{z})\ \mathrm{d}\mathbf{z}$

→ If we take Gaussians, then we get an **unbounded likelihood** [3]!

→ If we take Bernoullis (or Categoricals), it's **fine** [3]!


[3] Mattei, P. A., & Frellsen, J. (2018). Leveraging the exact likelihood of deep latent variable models. NeurIPS (pp. 3859-3870).

# The likelihood function for representation learning

Is then the likelihood function well-suited for representation learning?

# The likelihood function for representation learning

Is then the likelihood function well-suited for representation learning?

$$
\begin{aligned}
KL[p_{data}(\mathbf{x})||p_\theta(\mathbf{x})] &= -\mathbb{H}[p_{data}(\mathbf{x})] + \mathbb{CE}[p_{data}(\mathbf{x})||p_\theta(\mathbf{x})] \\
&= const + \mathbb{CE}[p_{data}(\mathbf{x})||p_\theta(\mathbf{x})] \\
&= const - \int p_{data}(\mathbf{x}) \ln p_\theta(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\
&= const - \frac{1}{N} \sum_{n=1}^{N} \ln p_\theta(\mathbf{x}_n)
\end{aligned}
$$

**The likelihood-based approach is equivalent to calculating a match between a model and the empirical distribution according to the KL-divergence.**

# The likelihood function for representation learning

Is then the likelihood function well-suited for representation learning?

| *Implicit* (**model** inductive biases) | ***Explicit*** (**objective** inductive biases) |
|---|---|
| • smoothness: x ≈ y implies f(x) ≈ f(y)<br>• multiple explanatory factors<br>• a hierarchy of explanatory factors<br>• geometrical properties (manifolds)<br>• factors explain x and help to predict y<br>• (?) spatial/temporal coherence<br>• (?) simplicity of factor dependencies | • smoothness: x ≈ y implies f(x) ≈ f(y)<br>• sparsity<br>• geometrical properties (manifolds)<br>• (?) spatial/temporal coherence<br>• (?) generalizability (shared factors across tasks) |

**The likelihood function alone does <u>not</u> help representation learning *explicitly*.**

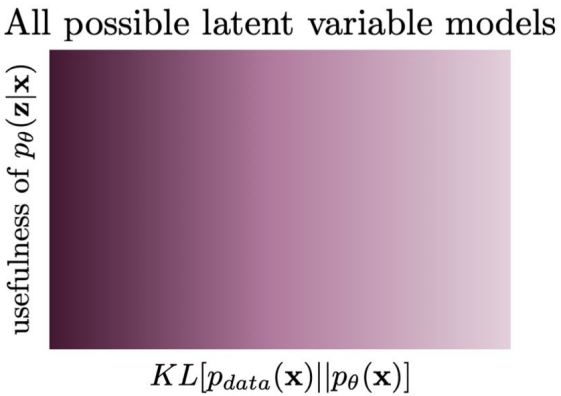


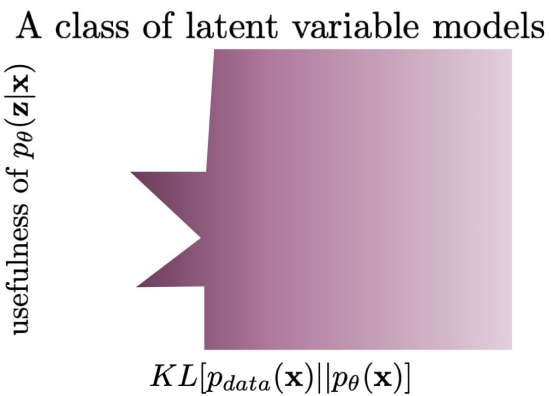
VU

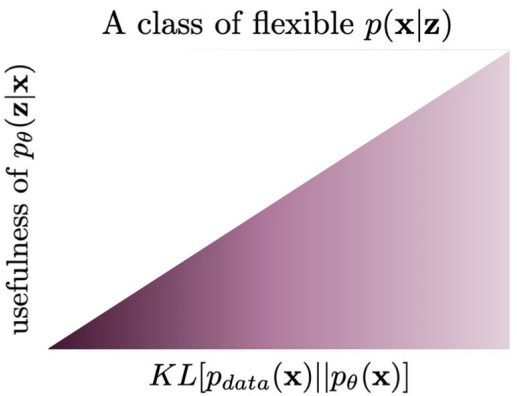
# The likelihood function for representation learning

Is then the likelihood function well-suited for representation learning?

Three scenarios:



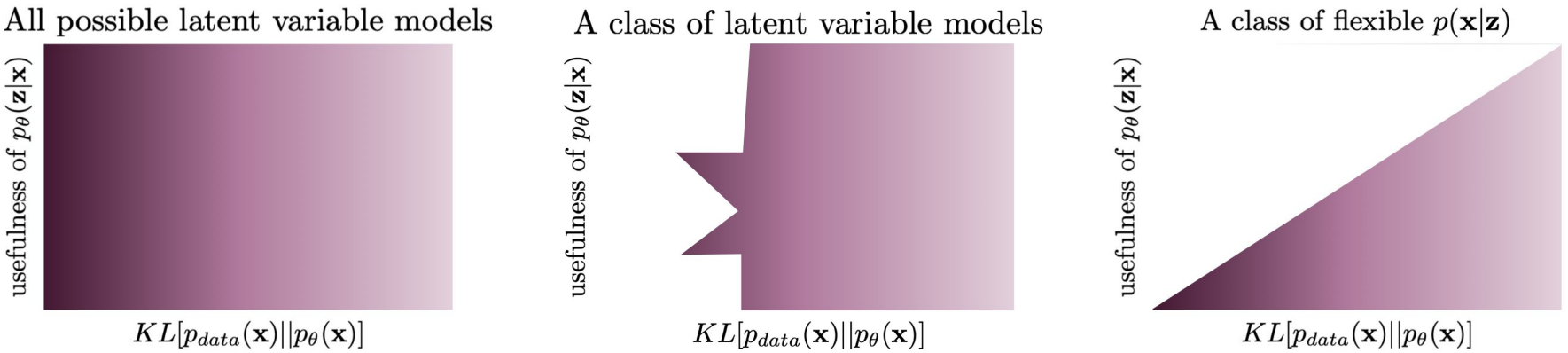


[4] Huszár, F. (2017). Is maximum likelihood useful for representation learning? https://www.inference.vc/maximum-likelihood-for-representation-learning-2/

# The likelihood function for representation learning

Is then the likelihood function well-suited for representation learning?

Three scenarios:



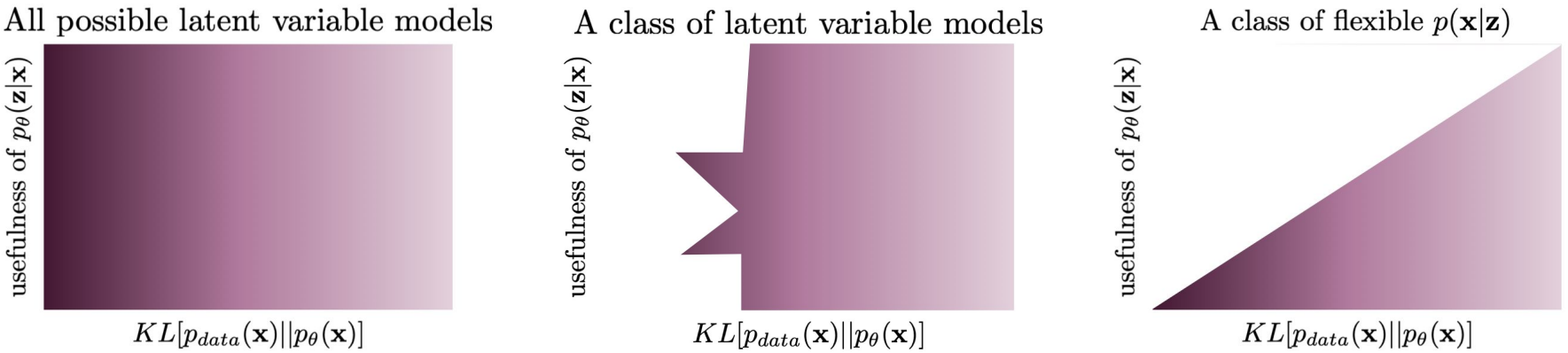

**The likelihood function has <u>no</u> incentive to learn *useful* representations.**


[4] Huszár, F. (2017). Is maximum likelihood useful for representation learning? https://www.inference.vc/maximum-likelihood-for-representation-learning-2/

# The likelihood function for representation learning

Is then the likelihood function well-suited for representation learning?

Three scenarios:




**The likelihood function has no incentive to learn *useful* representations.**

**But is it bad? Maybe we should focus on the parameterization instead!**


[4] Huszár, F. (2017). Is maximum likelihood useful for representation learning? https://www.inference.vc/maximum-likelihood-for-representation-learning-2/

# The likelihood function for representation learning

Is then the likelihood function well-suited for representation learning?

| ***Implicit*** (**model** inductive biases) | ***Explicit*** (**objective** inductive biases) |
|---|---|
| • smoothness: x ≈ y implies f(x) ≈ f(y)<br>• multiple explanatory factors<br>• a hierarchy of explanatory factors<br>• geometrical properties (manifolds)<br>• factors explain x and help to predict y<br>• (?) spatial/temporal coherence<br>• (?) simplicity of factor dependencies | • smoothness: x ≈ y implies f(x) ≈ f(y)<br>• sparsity<br>• geometrical properties (manifolds)<br>• (?) spatial/temporal coherence<br>• (?) generalizability (shared factors across tasks) |

**The likelihood function is a *general* objective and the parameterization is key?**

# How to model the marginal distribution? What parameterization?

# Classes of DGMs

Before we delve into the parameterizations, let us recap the DGMs:

- Latent Variables Models: $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\ p(\mathbf{z})\ \mathrm{d}\mathbf{z}$
- Flow-based models: $p(\mathbf{x}) = p\big(\mathbf{z} = f(\mathbf{x})\big)|\mathbf{J}_{f(\mathbf{x})}|$
- Autoregressive models: $p(\mathbf{x}) = p(x_0)\prod_{i=1}^{D} p(x_i|\mathbf{x}_{<i})$
- Energy-based models: $p(\mathbf{x}) = \dfrac{\exp\{-E(\mathbf{x})\}}{Z}$

# Classes of DGMs

Before we delve into the parameterizations, let us recap the DGMs:

- Latent Variables Models: $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})\ p(\mathbf{z})\ \mathrm{d}\mathbf{z}$
- Flow-based models: $p(\mathbf{x}) = p\big(\mathbf{z} = f(\mathbf{x})\big)|\mathbf{J}_{f(\mathbf{x})}|$
- Autoregressive models: $p(\mathbf{x}) = p(x_0)\prod_{i=1}^{D} p(x_i|\mathbf{x}_{<i})$
- Energy-based models: $p(\mathbf{x}) = \dfrac{\exp\{-E(\mathbf{x})\}}{Z}$

**Latent variables = representation**

**What is a *representation* here?**

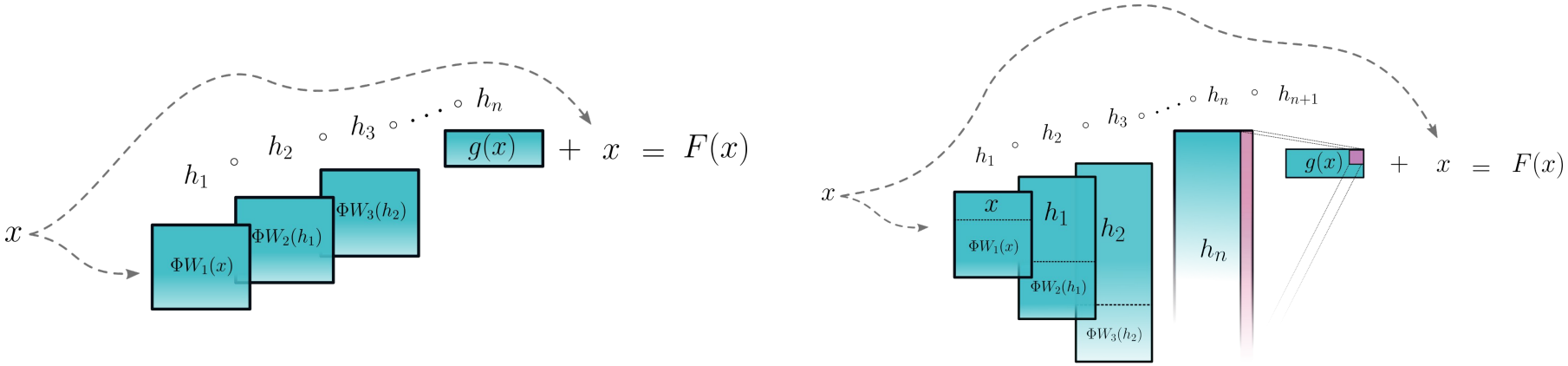
# Importance of parameterization in Flows

In flows, we need either powerful **invertible** neural networks, e.g.:

# Importance of parameterization in Flows

In flows, we need either powerful **invertible** neural networks, e.g.:

→ ResNets [5] or DenseNets [6] with constraints (less than 1-Lipschitz).





[5] Chen, R. T., Behrmann, J., Duvenaud, D. K., & Jacobsen, J. H. (2019). Residual Flows for Invertible Generative Modeling. NeurIPS 2019

[6] Perugachi-Diaz, Y., Tomczak, J. M., & Bhulai, S. (2021). Invertible densenets with concatenated lipswish. NeurIPS 2021

# Importance of parameterization in Flows

In flows, we need either powerful **invertible** neural networks, e.g.:

→ ResNets [5] or DenseNets [6] with constraints (less than 1-Lipschitz).

or/and **invertible structure** with **easily computable Jacobian**, e.g.:

→ Householder flows [7], Sylvester flows [8]

→ Convolution Exponential + Sylvester flows [9]

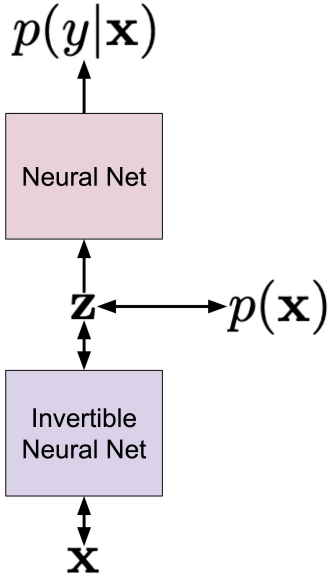

[5] Chen, R. T., Behrmann, J., Duvenaud, D. K., & Jacobsen, J. H. (2019). Residual Flows for Invertible Generative Modeling. NeurIPS 2019
[6] Perugachi-Diaz, Y., Tomczak, J. M., & Bhulai, S. (2021). Invertible densenets with concatenated lipswish. NeurIPS 2021
[7] Tomczak, J. M., & Welling, M. (2016). Improving variational auto-encoders using householder flow. Bayesian Deep Learing @ NeurIPS 2016
[8] van den Berg, R., Hasenclever, L., Tomczak, J. M., & Welling, M. Sylvester Normalizing Flows for Variational Inference. UAI 2018
[9] Hoogeboom, E., Garcia Satorras, V., Tomczak, J., & Welling, M. (2020). The Convolution Exponential and Generalized Sylvester Flows. NeurIPS 2020

# Flows for joint modeling

We can use flows to formulate a joint distribution [10]:

- The marginal distribution is a flow-based model.
- The mapping from **x** to **z** is shared with a predictor.
- We need to change the objective function:

$$\ell_\lambda(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \lambda \ln p(\mathbf{x})$$





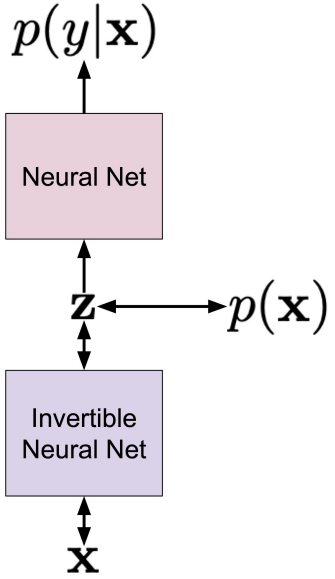
[10] Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., & Lakshminarayanan, B. (2019). Hybrid models with deep and invertible features. ICML 2019

# Flows for joint modeling

We can use flows to formulate a joint distribution [10]:

- The marginal distribution is a flow-based model.
- The mapping from **x** to **z** is shared with a predictor.
- We need to change the objective function:

$$\ell_\lambda(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \lambda \ln p(\mathbf{x})$$




**The gradient flow is uneven from the conditional and the marginal.**

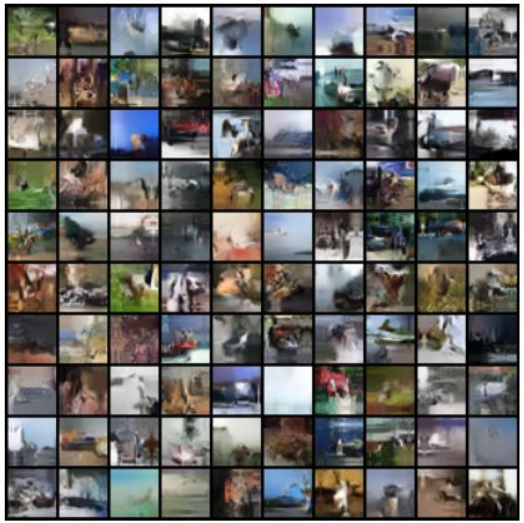

[10] Nalisnick, E., Matsukawa, A., Teh, Y. W., Gorur, D., & Lakshminarayanan, B. (2019). Hybrid models with deep and invertible features. ICML 2019

# Flows for joint modeling

We see that $\lambda$ is crucial [6]:

| | $\lambda = 0$ | $\lambda = \frac{1}{D}$ | | $\lambda = 1$ | |
|---|---|---|---|---|---|
| Model \Evaluation | Acc ↑ | Acc ↑ | bpd ↓ | Acc ↑ | bpd ↓ |
| Coupling | 89.77% | 87.58% | 4.30 | 67.62% | 3.54 |
| + $1 \times 1$ conv | 90.82% | 87.96% | 4.09 | 67.38% | 3.47 |
| Residual Blocks (full) | 91.78% | 90.47% | 3.62 | 70.32% | 3.39 |
| Dense Blocks (full) | **92.40%** | **90.79%** | **3.49** | **75.67%** | **3.31** |

(a) Hybrid model trained with $\lambda = \frac{1}{D}$



(b) Hybrid model trained with $\lambda = 1$


[6] Perugachi-Diaz, Y., Tomczak, J. M., & Bhulai, S. (2021). Invertible densenets with concatenated lipswish. NeurIPS 2021

# Flows for joint modeling

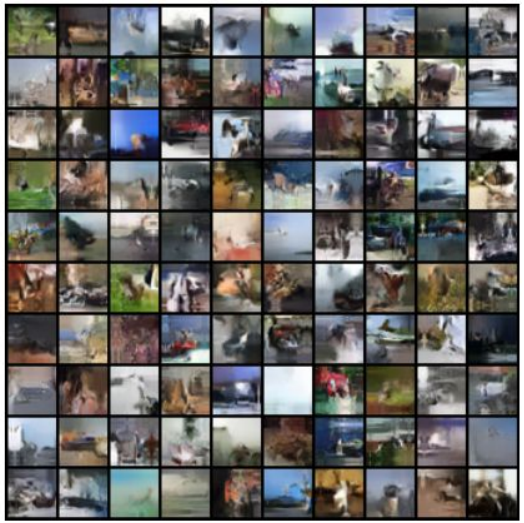
We see that $\lambda$ is crucial [6]:

| Model \Evaluation | $\lambda = 0$ Acc ↑ | $\lambda = \frac{1}{D}$ Acc ↑ | $\lambda = \frac{1}{D}$ bpd ↓ | $\lambda = 1$ Acc ↑ | $\lambda = 1$ bpd ↓ |
|---|---|---|---|---|---|
| Coupling | 89.77% | 87.58% | 4.30 | 67.62% | 3.54 |
| + $1 \times 1$ conv | 90.82% | 87.96% | 4.09 | 67.38% | 3.47 |
| Residual Blocks (full) | 91.78% | 90.47% | 3.62 | 70.32% | 3.39 |
| Dense Blocks (full) | **92.40%** | **90.79%** | **3.49** | **75.67%** | **3.31** |

$$\ell_\lambda(\mathbf{x}, y) = \ln p(y|\mathbf{x}) + \lambda \ln p(\mathbf{x})$$

**This is not the likelihood function anymore...**

(a) Hybrid model trained with $\lambda = \frac{1}{D}$



(b) Hybrid model trained with $\lambda = 1$


[6] Perugachi-Diaz, Y., Tomczak, J. M., & Bhulai, S. (2021). Invertible densenets with concatenated lipswish. NeurIPS 2021

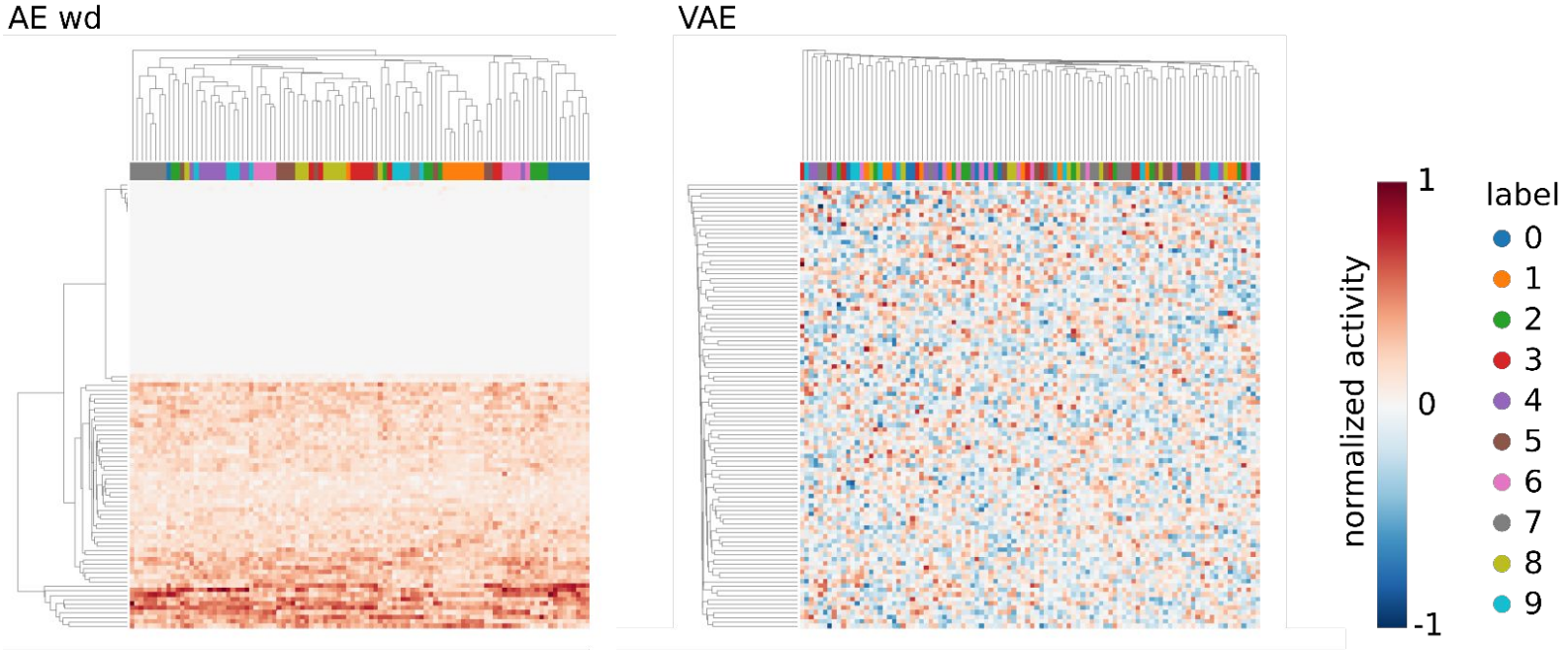
# Variational Auto-Encoders

- Variational Auto-Encoders (VAEs) constitute a broad class of models.

# Variational Auto-Encoders

- Variational Auto-Encoders (VAEs) constitute a broad class of models.

- But they fail to learn a *representation* [11].




[11] Hübotter, J. F., Lanillos, P., & Tomczak, J. M. (2021). Training Deep Spiking Auto-encoders without Bursting or Dying Neurons through Regularization. arXiv preprint arXiv:2109.11045.

# Variational Auto-Encoders

- Variational Auto-Encoders (VAEs) constitute a broad class of models.
- But they fail to learn a *representation* [11].
- It is possible to introduce geometrical perspective to VAEs:

→ Hyperspherical VAEs [12], Riemaniann manifolds [13], Lie groups [14]

- Hierarchical latent structure is crucial for learning powerful VAEs.

[11] Hübotter, J. F., Lanillos, P., & Tomczak, J. M. (2021). Training Deep Spiking Auto-encoders without Bursting or Dying Neurons through Regularization. arXiv preprint arXiv:2109.11045.
[12] Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., & Tomczak, J. M. (2018). Hyperspherical variational auto-encoders. UAI 2018
[13] Arvanitidis, G., Hauberg, S., & Schölkopf, B. (2021). Geometrically Enriched Latent Spaces. AISTATS 2021
[14] Falorsi, L., de Haan, P., Davidson, T. R., & Forré, P. (2019). Reparameterizing distributions on lie groups. AISTATS 2019

# Hierarchical VAEs

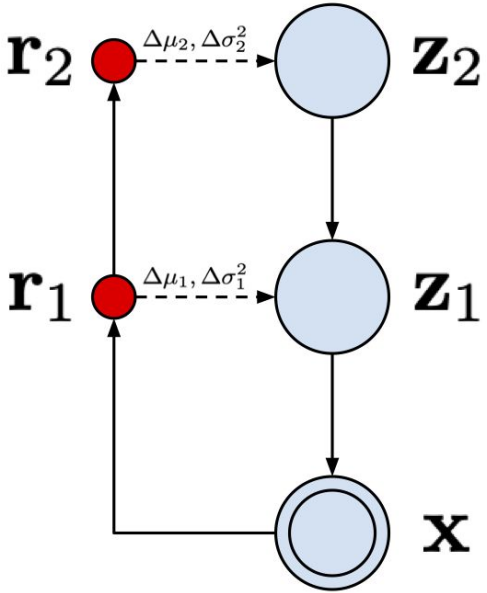
A proper **parameterization** of hierarchical VAEs is important to get SOTA.

So called top-down VAEs:

→ ResNet VAE [15], Ladder VAE [16], BIVA [17], NVAE [18], vdVAE [19]

$$p(\mathbf{x}|\mathbf{z}_1)p(\mathbf{z}_1|\mathbf{z}_2)p(\mathbf{z}_2)$$

$$q(\mathbf{z}_1|\mathbf{z}_2,\mathbf{x})q(\mathbf{z}_2|\mathbf{x})$$





[15] Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., & Welling, M. (2016). Improved variational inference with inverse autoregressive flow. NeurIPS 2016

[16] Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., & Winther, O. (2016). Ladder variational autoencoders. NeurIPS 2016

[17] Maaløe, L., Fraccaro, M., Liévin, V., & Winther, O. (2019). BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling. NeurIPS 2019

[18] Vahdat, A., & Kautz, J. (2020). Nvae: A deep hierarchical variational autoencoder. NeurIPS 2020

[19] Child, R. (2020). Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images. ICLR 2020




VU

# VAEs for joint modeling

However, still, using VAEs for modeling the marginal and sharing the encoder with the predictor requires applying a fudge factor.

The objective can take the following form [20, 21]:

$$\ell_\alpha(\mathbf{x}, y) = \alpha \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\ln p(y|\mathbf{z})\right] + ELBO(\mathbf{x})$$

The encoder is shared with the classifier.

[20] Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). Semi-supervised learning with deep generative models. NeurIPS 2014
[21] Ilse, M., Tomczak, J. M., Louizos, C., & Welling, M. (2020). DIVA: Domain invariant variational autoencoders. MIDL 2020

# VAEs for joint modeling

However, still, using VAEs for modeling the marginal and sharing the encoder with the predictor requires applying a fudge factor.

The objective can take the following form [20, 21]:

$$\ell_\alpha(\mathbf{x}, y) = \alpha \mathbb{E}_{q(\mathbf{z}|\mathbf{x})} \left[\ln p(y|\mathbf{z})\right] + ELBO(\mathbf{x})$$

The encoder is shared with the classifier.

**This is not the likelihood function anymore...**

[20] Kingma, D. P., Mohamed, S., Rezende, D. J., & Welling, M. (2014). Semi-supervised learning with deep generative models. NeurIPS 2014
[21] Ilse, M., Tomczak, J. M., Louizos, C., & Welling, M. (2020). DIVA: Domain invariant variational autoencoders. MIDL 2020

# Energy-based Models

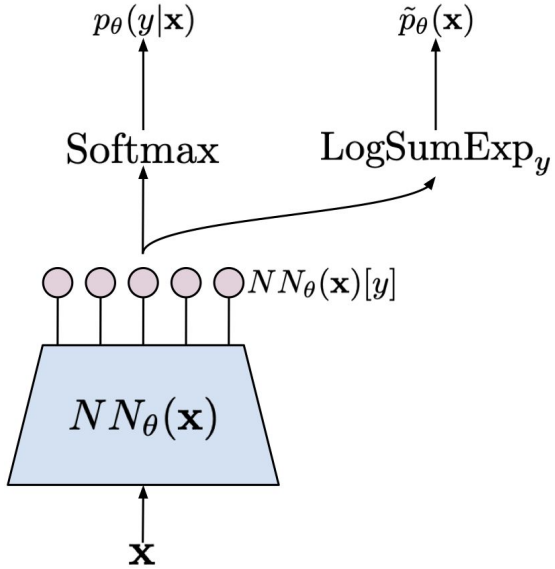
It is unclear for ARMs and EBMs what a *representation* is.

# Energy-based Models

It is unclear for ARMs and EBMs what a *representation* is.

However, for EBMs we have [22]:

$$\begin{aligned}\ln p_\theta(\mathbf{x}, y) &= \ln p_\theta(y|\mathbf{x}) + \ln p_\theta(\mathbf{x}) \\ &= \ln \frac{\exp\{NN_\theta(\mathbf{x})[y]\}}{\sum_y \exp\{NN_\theta(\mathbf{x})[y]\}} + \ln \frac{\sum_y \exp\{NN_\theta(\mathbf{x})[y]\}}{Z_\theta} \\ &= \ln \text{softmax}\{NN_\theta(\mathbf{x})[y]\} + \left(\text{LSE}_y\{NN_\theta(\mathbf{x})[y]\} - \ln Z_\theta\right)\end{aligned}$$





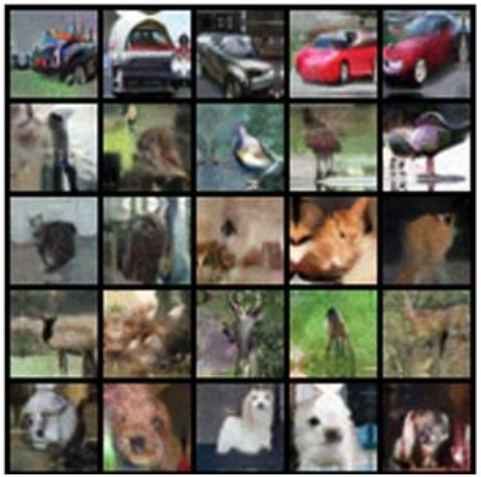

[22] Grathwohl, W., Wang, K. C., Jacobsen, J. H., Duvenaud, D., Norouzi, M., & Swersky, K. (2020). Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One. ICLR 2020

# Energy-based Models

It is unclear for ARMs and EBMs what a *representation* is.

However, for EBMs we have [22]:

| Class | Model | Accuracy% ↑ | IS↑ | FID↓ |
|---|---|---|---|---|
| **Hybrid** | Residual Flow | 70.3 | 3.6 | 46.4 |
| | Glow | 67.6 | 3.92 | 48.9 |
| | IGEBM | 49.1 | 8.3 | **37.9** |
| | JEM $p(\mathbf{x}\|y)$ factored | 30.1 | 6.36 | 61.8 |
| | JEM (Ours) | **92.9** | **8.76** | 38.4 |
| **Disc.** | Wide-Resnet | 95.8 | N/A | N/A |
| **Gen.** | SNGAN | N/A | 8.59 | 25.5 |
| | NCSN | N/A | 8.91 | 25.32 |





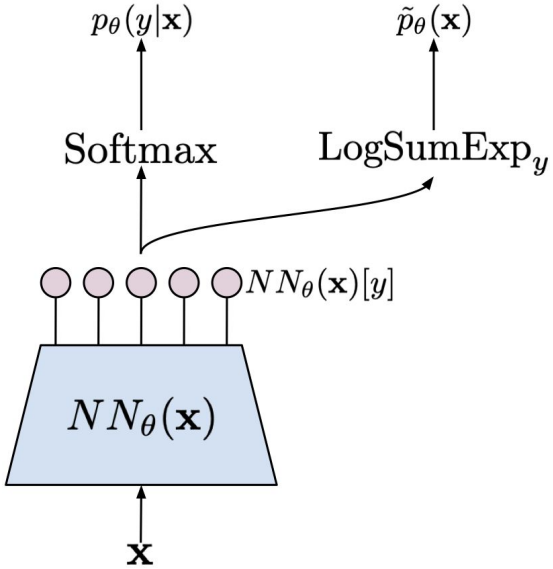
[22] Grathwohl, W., Wang, K. C., Jacobsen, J. H., Duvenaud, D., Norouzi, M., & Swersky, K. (2020). Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One. ICLR 2020

# Energy-based Models

It is unclear for ARMs and EBMs what a *representation* is.

However, for EBMs we have [22]:

$$\begin{aligned}\ln p_\theta(\mathbf{x}, y) &= \ln p_\theta(y|\mathbf{x}) + \ln p_\theta(\mathbf{x}) \\ &= \ln \frac{\exp\{NN_\theta(\mathbf{x})[y]\}}{\sum_y \exp\{NN_\theta(\mathbf{x})[y]\}} + \ln \frac{\sum_y \exp\{NN_\theta(\mathbf{x})[y]\}}{Z_\theta} \\ &= \ln \operatorname{softmax}\{NN_\theta(\mathbf{x})[y]\} + \left(\operatorname{LSE}_y\{NN_\theta(\mathbf{x})[y]\} - \ln Z_\theta\right)\end{aligned}$$




**A single, shared neural net.**

**The objective is the likelihood function!**



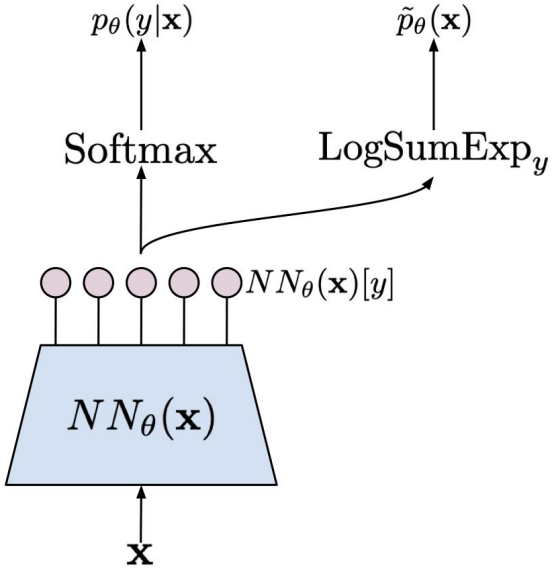
[22] Grathwohl, W., Wang, K. C., Jacobsen, J. H., Duvenaud, D., Norouzi, M., & Swersky, K. (2020). Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One. ICLR 2020

# Energy-based Models

It is unclear for ARMs and EBMs what a *representation* is.

However, for EBMs we have [22]:

$$\begin{aligned}\ln p_\theta(\mathbf{x}, y) &= \ln p_\theta(y|\mathbf{x}) + \ln p_\theta(\mathbf{x}) \\ &= \ln \frac{\exp\{NN_\theta(\mathbf{x})[y]\}}{\sum_y \exp\{NN_\theta(\mathbf{x})[y]\}} + \ln \frac{\sum_y \exp\{NN_\theta(\mathbf{x})[y]\}}{Z_\theta} \\ &= \ln \operatorname{softmax}\{NN_\theta(\mathbf{x})[y]\} + \left(\operatorname{LSE}_y\{NN_\theta(\mathbf{x})[y]\} - \ln Z_\theta\right)\end{aligned}$$




**A single, shared neural net.**

**The objective is the likelihood function!**

**But what is the representation here?**



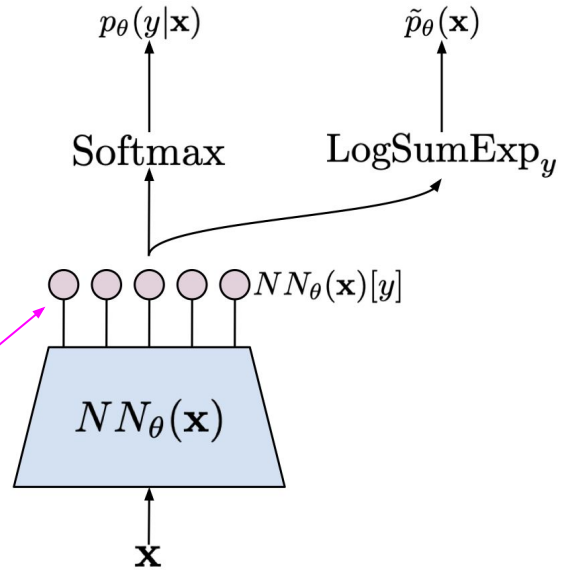
[22] Grathwohl, W., Wang, K. C., Jacobsen, J. H., Duvenaud, D., Norouzi, M., & Swersky, K. (2020). Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One. ICLR 2020

# Energy-based Models

It is unclear for ARMs and EBMs what a *representation* is.

However, for EBMs we have [22]:

$$\begin{aligned}\ln p_\theta(\mathbf{x}, y) &= \ln p_\theta(y|\mathbf{x}) + \ln p_\theta(\mathbf{x}) \\ &= \ln \frac{\exp\{NN_\theta(\mathbf{x})[y]\}}{\sum_y \exp\{NN_\theta(\mathbf{x})[y]\}} + \ln \frac{\sum_y \exp\{NN_\theta(\mathbf{x})[y]\}}{Z_\theta} \\ &= \ln \text{softmax}\{NN_\theta(\mathbf{x})[y]\} + \left(\text{LSE}_y\{NN_\theta(\mathbf{x})[y]\} - \ln Z_\theta\right)\end{aligned}$$




**A single, shared neural net.**

**The objective is the likelihood function!**

**But what is the representation here?**

**logits?**

**How to obtain a representation?**



[22] Grathwohl, W., Wang, K. C., Jacobsen, J. H., Duvenaud, D., Norouzi, M., & Swersky, K. (2020). Your Classifier is Secretly an Energy Based Model and You Should Treat it Like One. ICLR 2020

# A quick summary

The likelihood function:

- Potential issues with mixture models (unboundness).
- Fine for discrete distributions.

# A quick summary

The likelihood function:

- Potential issues with mixture models (unboundness).
- Fine for discrete distributions.

Marginal models:

- For ARMs and EBMs: Unclear what a *representation* is.
- For VAEs and flows: We don't know how to use the likelihood function.

# A quick summary

The likelihood function:

- Potential issues with mixture models (unboundness).
- Fine for discrete distributions.

Marginal models:

**Do we really need one?**

- For ARMs and EBMs: Unclear what a *representation* is.
- For VAEs and flows: We don't know how to use the likelihood function.

**Where is the problem?**



VU

# Do we need a notion of *representation learning*?

# What is a *representation*?

Learning a joint distribution raises questions:

- Is there something wrong with the likelihood function? Or rather:
- How should we parameterize distributions?

Interestingly, EBMs don’t need to define an explicit representation and they can use the likelihood function!

# What is a *representation*?

Learning a joint distribution raises questions:

- Is there something wrong with the likelihood function? Or rather:
- How should we parameterize distributions?

Interestingly, EBMs don't need to define an explicit representation and they can use the likelihood function!

**So do we really need a *representation*?**

**Or maybe we misinterpret the *representation*?**



VU

# Compression

Shannon's source coding theorem:

*The length of a message representing some data is proportional to the entropy of this data.*

# Compression

Shannon's source coding theorem:

*The length of a message representing some data is proportional to the entropy of this data.*

In other words:

***Maximizing the model log-likelihood*** *is equivalent to* ***minimizing the expected number of bits required per message****, if the encoder is optimal.*

So we want to get $p_{real}(\mathbf{x}) \approx p_{\theta}(\mathbf{x})$.

# Compression

Shannon’s theorem could be read as follows:

# Compression

Shannon's theorem could be read as follows:

→ A model maximizing the likelihood function is **useful for compression**.

# Compression

Shannon's theorem could be read as follows:

→ A model maximizing the likelihood function is **useful for compression**.

→ There is no need to define a separate variable to send a message.

A good model $p_\theta(\mathbf{x})$ is all we need! 😊

# Compression

Shannon's theorem could be read as follows:

→ A model maximizing the likelihood function is **useful for compression**.

→ There is no need to define a separate variable to send a message.

A good model $p_\theta(\mathbf{x})$ is all we need! 😊

→ A message is discrete in nature (i.e., symbols and bits).

# Compression

Shannon's theorem could be read as follows:

→ A model maximizing the likelihood function is **useful for compression**.

→ There is no need to define a separate variable to send a message.

A good model $p_\theta(\mathbf{x})$ is all we need! 😊

→ A message is discrete in nature (i.e., symbols and bits).

In information theory we have a "natural" notion of compressed representation of data. Do we need more?

# Compression

Shannon's theorem could be read as follows:

→ A model maximizing the likelihood function is **useful for compression**.

→ There is no need to define a separate variable to send a message.

A good model $p_\theta(\mathbf{x})$ is all we need! 😊

→ A message is discrete in nature (i.e., symbols and bits).

In information theory we have a "natural" notion of compressed representation of data. Do we need more?

**But then, how to use compression in the context of prediction?**

# Questions

VU VRIJE UNIVERSITEIT AMSTERDAM

# Questions:

1. Is the likelihood function enough?
    a. If NO, then what priors/regularizers are important? (*Explicit*)
    b. If YES, what parameterizations are important? (*Implicit*)

# Questions:

1. Is the likelihood function enough?

   **What would Bayesianists say about it?**

   a. If NO, then what priors/regularizers are important? (*Explicit*)

   b. If YES, what parameterizations are important? (*Implicit*)



VU

# Questions:

1. Is the likelihood function enough?
    a. If NO, then what priors/regularizers are important? (*Explicit*)
    b. If YES, what parameterizations are important? (*Implicit*)

2. How to obtain a *representation* from EBMs or ARMs?

# Questions:

1. Is the likelihood function enough?
    a. If NO, then what priors/regularizers are important? (*Explicit*)
    b. If YES, what parameterizations are important? (*Implicit*)

2. How to obtain a *representation* from EBMs or ARMs?

3. Is the notion of *representation learning* even necessary? Is it a different label for compression?

# Thank you!

VU VRIJE UNIVERSITEIT AMSTERDAM