《神经网络与深度学习》

深度生成式模型 Part-I

Deep Generative Models

王逍

xiaowang@ahu.edu.cn

计算机科学与技术学院 安徽大学

目 录

目 录

Background

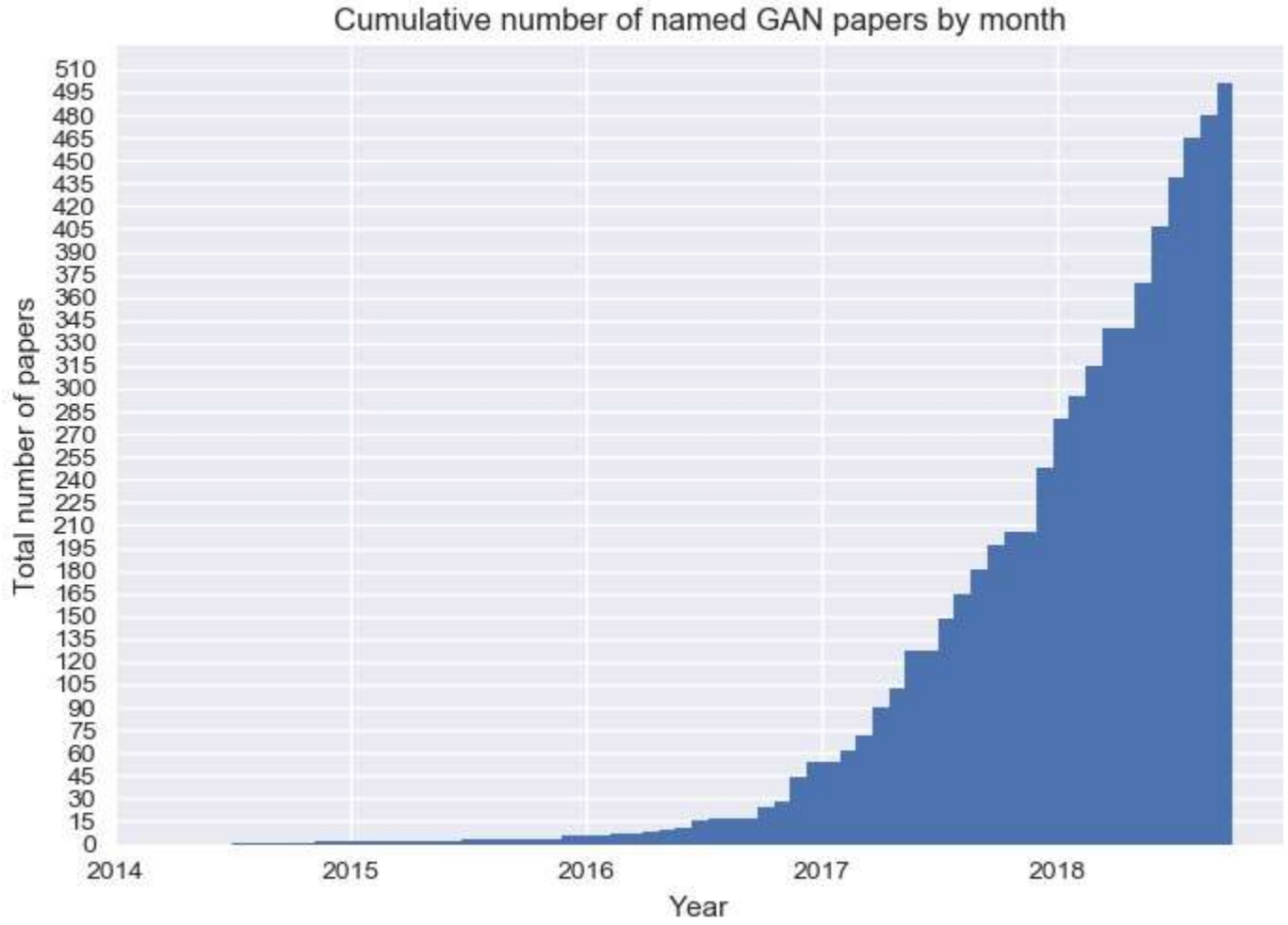
Deep Learning Driven Approaches

Learning Methods

- Supervised Learning
- Unsupervised Learning
  - **VAE**
  - **GAN**
  - **Diffusion Models**
  - ……
- Semi-Supervised Learning
- Reinforcement Learning
  - **Policy Gradient**
  - **Q-learning**
  - **A3C**
  - ……

Background

https://github.com/hindupuravinash/the-gan-zoo/blob/master/cumulative_gans.jpg

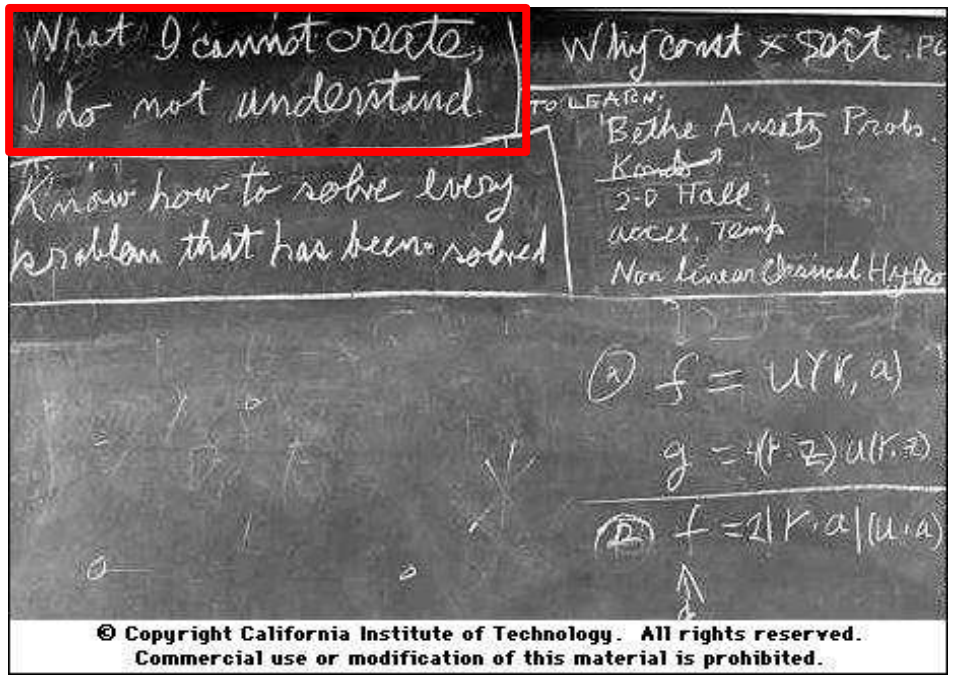


Background

Drawing?

Writing Poems?

Background



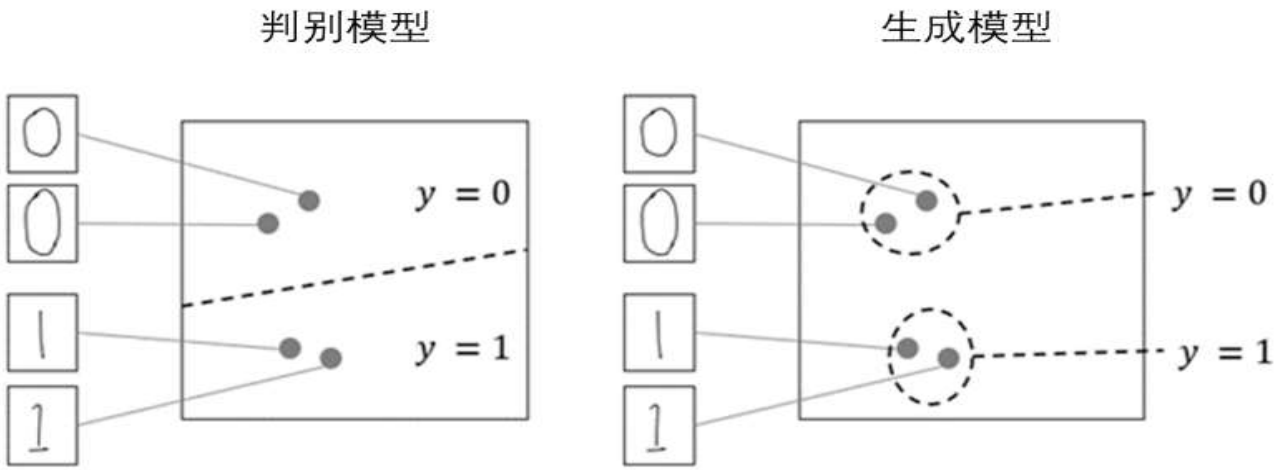
Richard Feynman

What I cannot create, I do not understand.

生成式模型

生成式模型是为了让学习机器能够产生与训练样本具有相同性质和规律的样本。

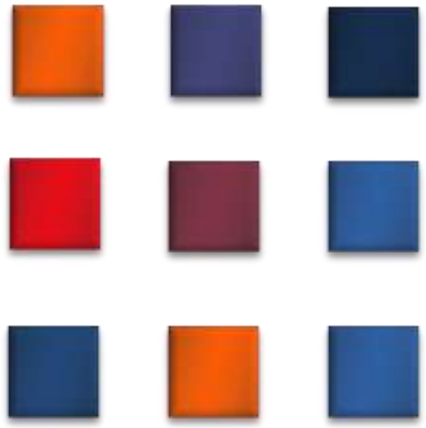
- **生成式模型（generative model）**：估计或模拟样本的概率分布
  - 对于分类问题，把类别标签看作是样本的一部分，用生成模型来学习样本和类别标签的联合概率分布，即：计算样本$\boldsymbol{x}$以及标签y联合概率分布$p(\boldsymbol{x}, y)$
- **判别式模型（discriminative model）**
  - 对后验概率$P(y|\boldsymbol{x})$直接建模

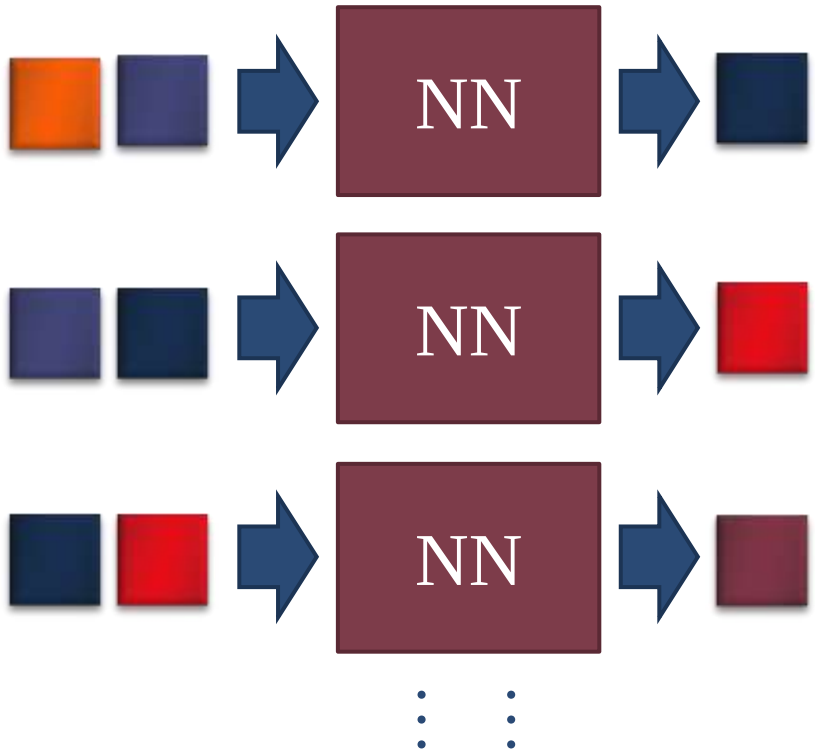


Component-by-component

- **Image generation**

E.g. 3 x 3 images



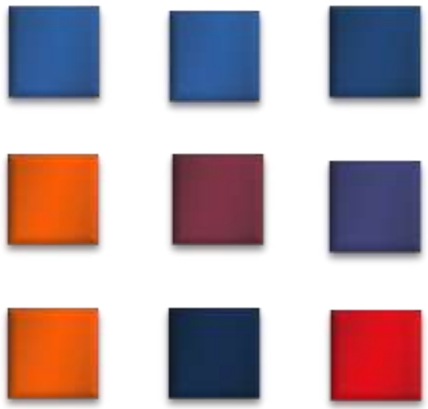


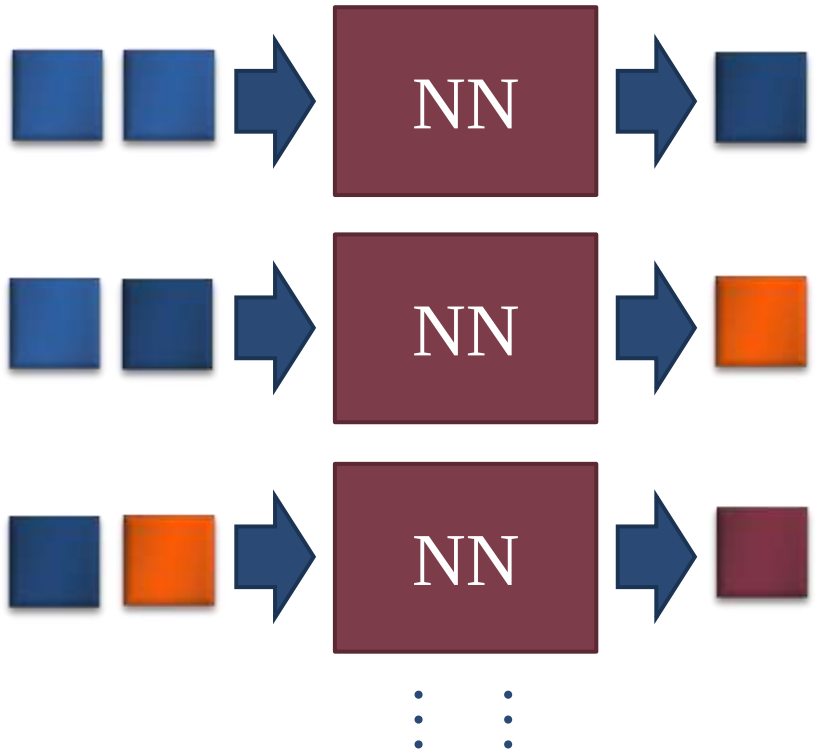
Can be trained just with a large collection of images without any annotation

Component-by-component

- **Image generation**

E.g. 3 x 3 images





Can be trained just with a large collection of images without any annotation

Component-by-component

Pixel-RNN

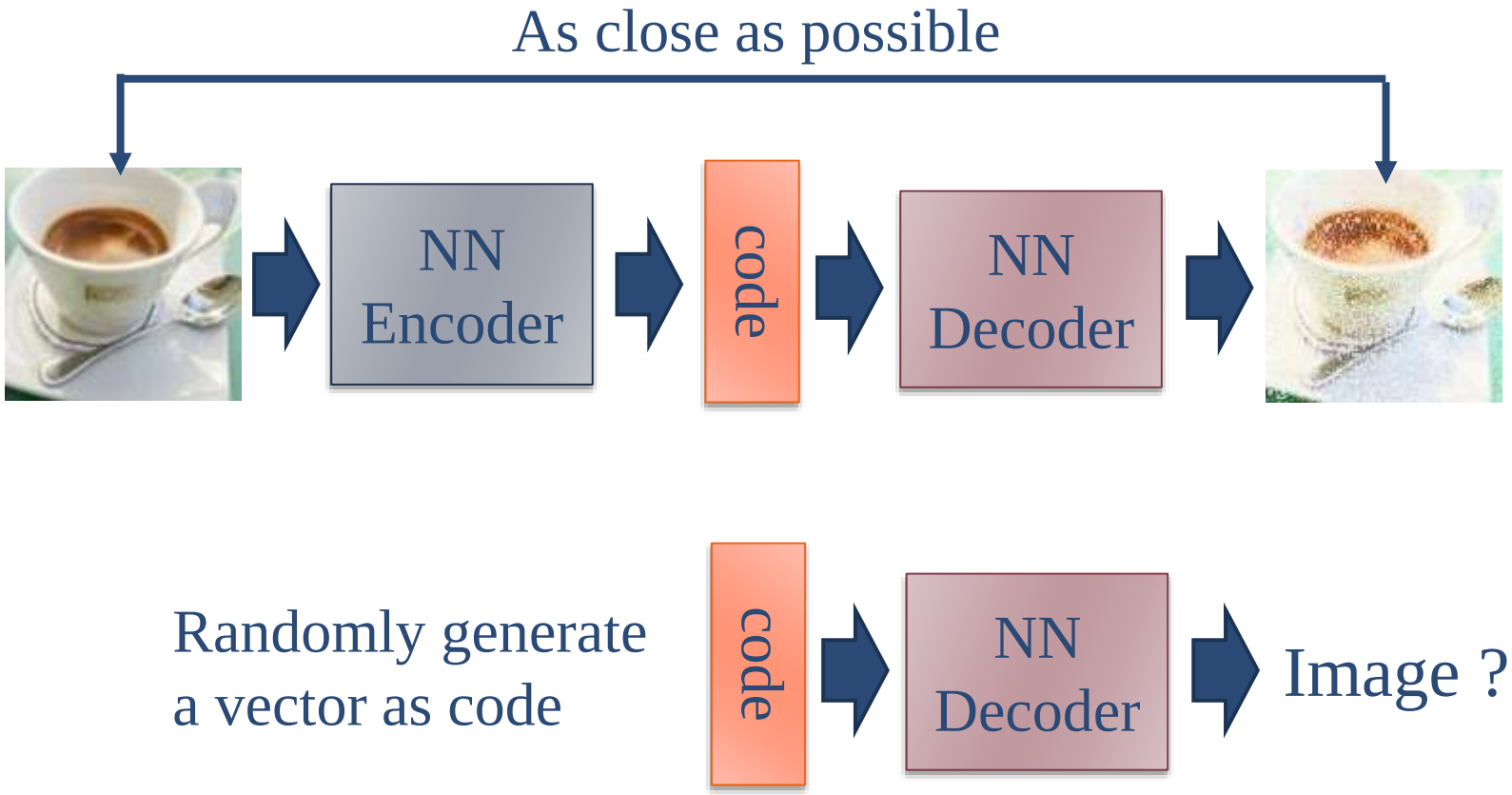
Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu, **Pixel Recurrent Neural Networks**, arXiv preprint, 2016

Real World

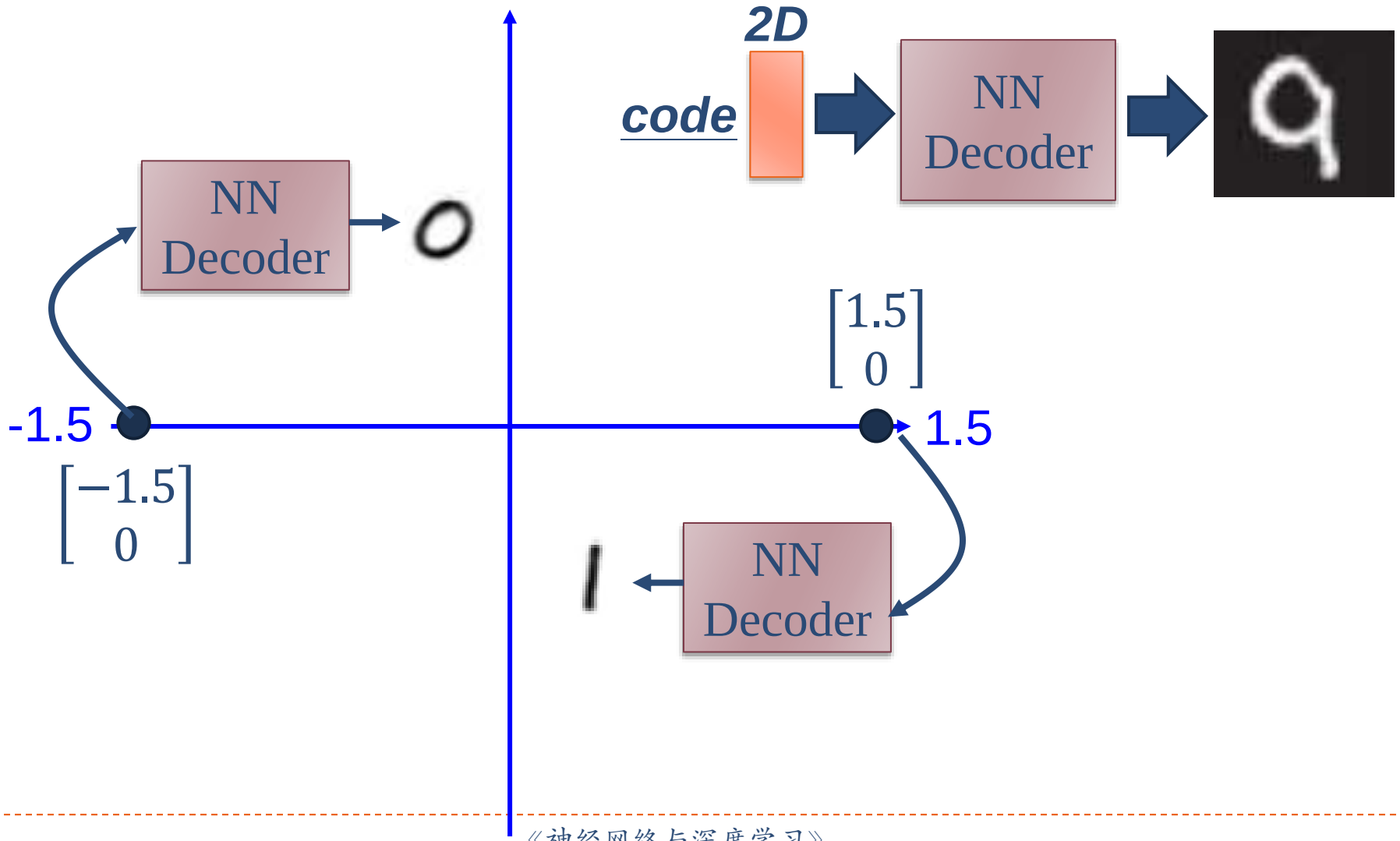
What is a latent variable?

Can we learn the **true explanatory factors**, e.g. latent variables, from only observed data?

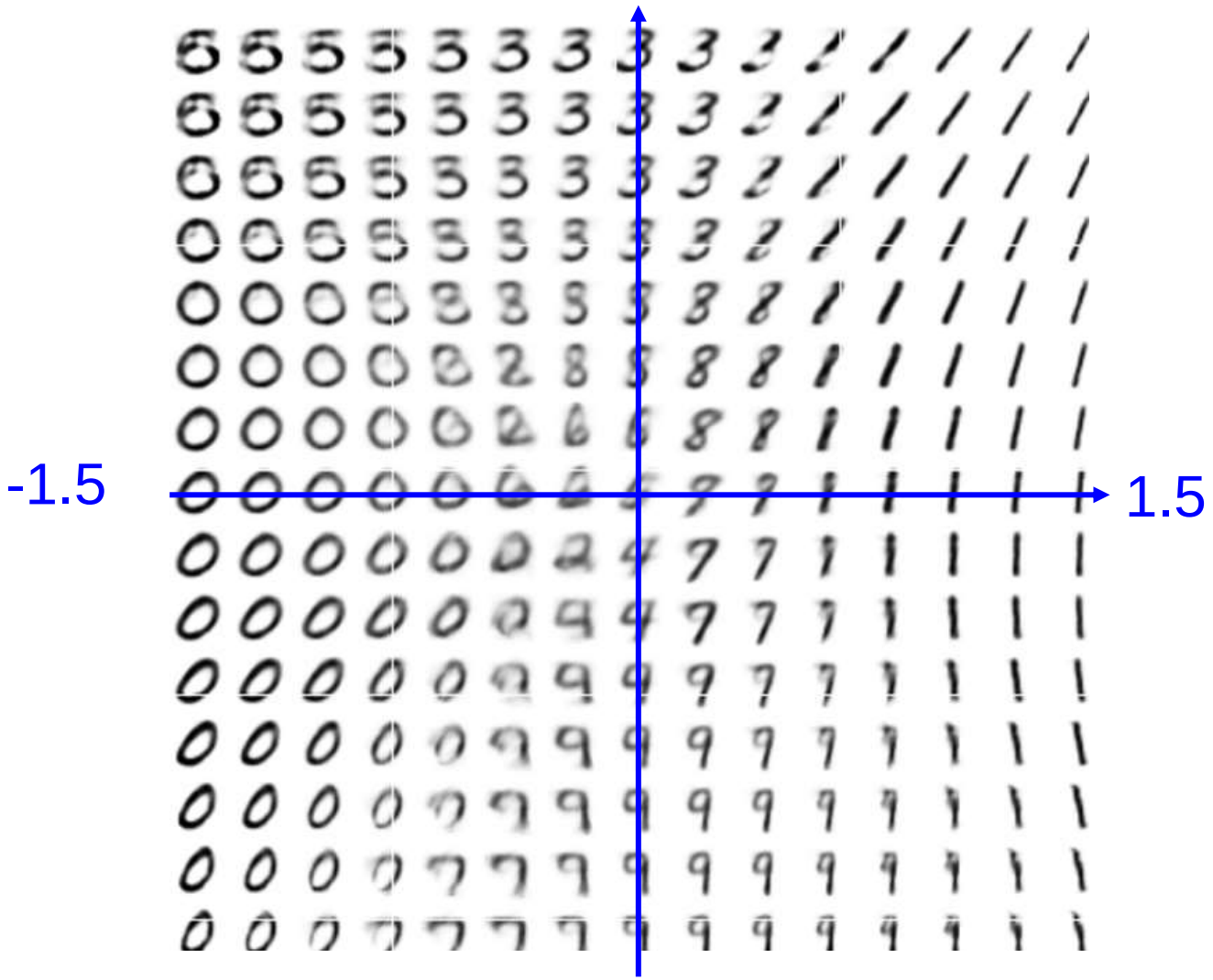
Auto-encoder



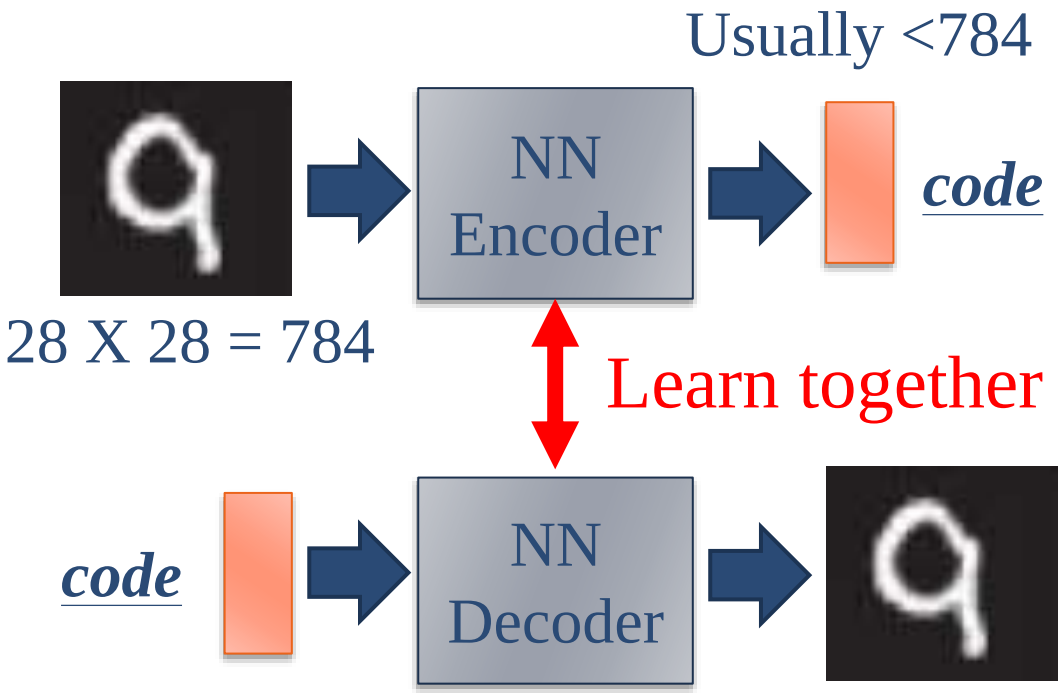
Auto-encoder



Auto-encoder



Auto-encoder



Compact representation of the input object

Can reconstruct the original object

AE 的局限性

人们发现对于已经训练的很好的自编码器，随机的给定编码并无法让自编码器产生出符合原样本特点的新样本。

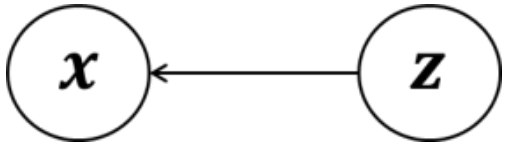
因为我们对这些**有效编码在编码空间中的分布**并不了解。任意的给一组编码，很难恰巧落在有效编码的区域中，因此无法生成与训练样本相似的有效新样本。

问题：如何约束或者估计样本在隐层空间中的分布呢?

变分自编码器（VAE）

- 通过训练样本学习隐节点变量概率分布
- 看似与自编码器有相同构造，数学模型完全不同，本质上是概率图模型
- 模型的参数推断采用变分推断（Variational Inference ）方法

VAE 的概率图模型

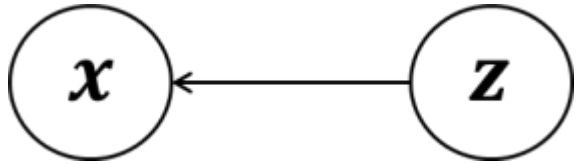


- 对样本x，从一组隐含向量z中依一定的概率模型产生出来的

$$p(x,z) = p(x|z)p(z)$$

变分自编码器（VAE）

- 目标：推断z的后验概率密度$p(z|x)$



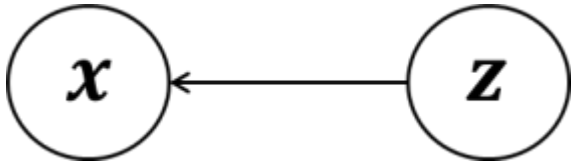
$$p(z|x) = \frac{p(x|z)p(z)}{p(x)}$$

其中：

$$p(x) = \int_z p(z)p(x|z)dz$$

- $p(x)$是观测样本x的边缘密度，也称作“证据”（Evidence），但难以求解
- 可以利用马尔可夫链蒙特卡洛（MCMC）求解，但效率较低

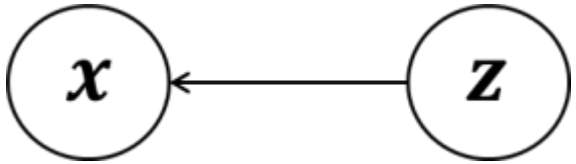
变分自编码器（VAE）



- 变分推断：对函数和泛函的微小变化（称作变分）寻找泛函的极值。
- 为了推断$p(\boldsymbol{z}|\boldsymbol{x})$，引入某个易计算的变分函数$q(\boldsymbol{z})$，
- 如何定义“接近”？
- 衡量概率密度函数之间的相似度：KL散度
- 设x是一个离散的随机变量，它的概率分布函数是$P(x)$
- 定义信息量$I(x) = -\log P(x)$，则随机事件的熵：

$$H(x) = -\sum_{x} P(x)\log P(x)$$

变分自编码器（VAE）



- 同一随机变量，还存在另一个概率分布函数$Q(x)$
- **P(x)和Q(x)的差异定义为它们熵的差**，即**KL散度**：

$$D_{KL}(Q||P) = \sum_x Q(x)\log Q(x) - \sum_x Q(x)\log P(x) = \sum_x Q(x)\log\frac{Q(x)}{P(x)}$$

- 可等价写为：

$$D_{KL}(Q||P) = -\sum_x Q(x)\log\frac{P(x)}{Q(x)}$$

- 一般情况下，$D_{KL}(P||Q) \neq D_{KL}(Q||P)$
- 此外，还满足非负性：$D_{KL}(P||Q) \geq 0$

变分自编码器（VAE）

- 在变分推理中，我们引入变分函数$q(\boldsymbol{z})$来逼近难求解的后验概率密度函数$p(\boldsymbol{z}|\boldsymbol{x})$，使得两者尽可能接近。
- KL散度为：

$$D_{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) = \int_{\boldsymbol{z}} q(\boldsymbol{z})\log\frac{q(\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}d\boldsymbol{z} = \int_{\boldsymbol{z}} q(\boldsymbol{z})\log\frac{q(\boldsymbol{z})p(\boldsymbol{x})}{p(\boldsymbol{x},\boldsymbol{z})}d\boldsymbol{z}$$

$$= \int_{\boldsymbol{z}} q(\boldsymbol{z})(\log q(\boldsymbol{z}) + \log p(\boldsymbol{x}) - \log p(\boldsymbol{x},\boldsymbol{z}))d\boldsymbol{z}$$

$$= \log p(\boldsymbol{x}) + \int_{\boldsymbol{z}} q(\boldsymbol{z})\log q(\boldsymbol{z})d\boldsymbol{z} - \int_{\boldsymbol{z}} q(\boldsymbol{z})\log p(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{z}$$

- 由此可得：

$$\log p(\boldsymbol{x}) = D_{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) - \int_{\boldsymbol{z}} q(\boldsymbol{z})\log q(\boldsymbol{z})d\boldsymbol{z} + \int_{\boldsymbol{z}} q(\boldsymbol{z})\log p(\boldsymbol{x},\boldsymbol{z})d\boldsymbol{z}$$

$$= D_{KL}(q(\boldsymbol{z})||p(\boldsymbol{z}|\boldsymbol{x})) + \int_{\boldsymbol{z}} q(\boldsymbol{z})\log\frac{p(\boldsymbol{x},\boldsymbol{z})}{q(\boldsymbol{z})}d\boldsymbol{z}$$

第一项始终 >= 0

变分自编码器（VAE）

- 由非负性可知：

$$\log p(\boldsymbol{x}) \geq \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z}$$

$\log p(\boldsymbol{x})$为变分下界

- 令

$$\mathcal{L}(q) \triangleq \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z}$$

- $\mathcal{L}(q)$称作**证据下界**，简写为ELBO
- 最大化证据下界ELBO即等价于最小化引入的变分函数与隐含变量密度函数之间的KL散度。

变分自编码器（VAE）

$$\mathcal{L}(q) \triangleq \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z}$$

- 分解$\mathcal{L}(q)$中联合概率有：

$$\mathcal{L}(q) = \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log p(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z} + \int_{\boldsymbol{z}} q(\boldsymbol{z}) \log \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} d\boldsymbol{z}$$
$$= E_{\boldsymbol{z} \sim q(\boldsymbol{z})} \log p(\boldsymbol{x}|\boldsymbol{z}) - D_{KL}(q(\boldsymbol{z}) || p(\boldsymbol{z}))$$

上式中$q(\boldsymbol{z})$可看做在观测样本下从给定函数族中求解的概率密度函数，因此可以看作$\boldsymbol{x}$到$\boldsymbol{z}$的条件密度$q(\boldsymbol{z}|\boldsymbol{x})$

- 变分推断就是最大化证据下界 ELBO，
  - 一方面，可以使得隐含变量解释观测数据的似然函数尽可能大；
  - 另一方面，使得变分函数尽可能靠近隐含变量的先验密度函数（KL散度尽可能小）。

变分自编码器（VAE）

第一项：反映了模型对样本重构的程度

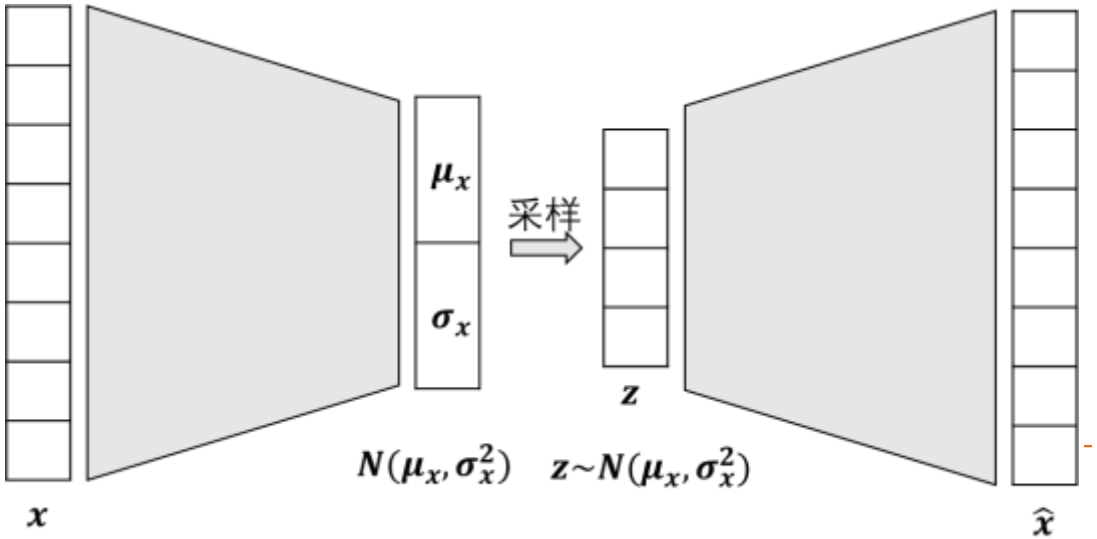
- VAE中用编码层来实现$q(\boldsymbol{z}|\boldsymbol{x})$，用解码层来实现$p(\boldsymbol{x}|\boldsymbol{z})$
- 目标函数：最大化ELBO：

$$\max \ \mathcal{L}(q) = E_{\boldsymbol{z}\sim q(\boldsymbol{z})}\log p(\boldsymbol{x}|\boldsymbol{z}) - D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$$

第二项：要求自编码器的编码层节点尽量符合一个给定的概率密度函数 p(z)

- 指定隐层为各项独立的多元正态分布$\boldsymbol{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$，目标等价于：

$$\min \quad \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|^2 + D_{KL}(q(\boldsymbol{z}|\boldsymbol{x})||N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2))$$



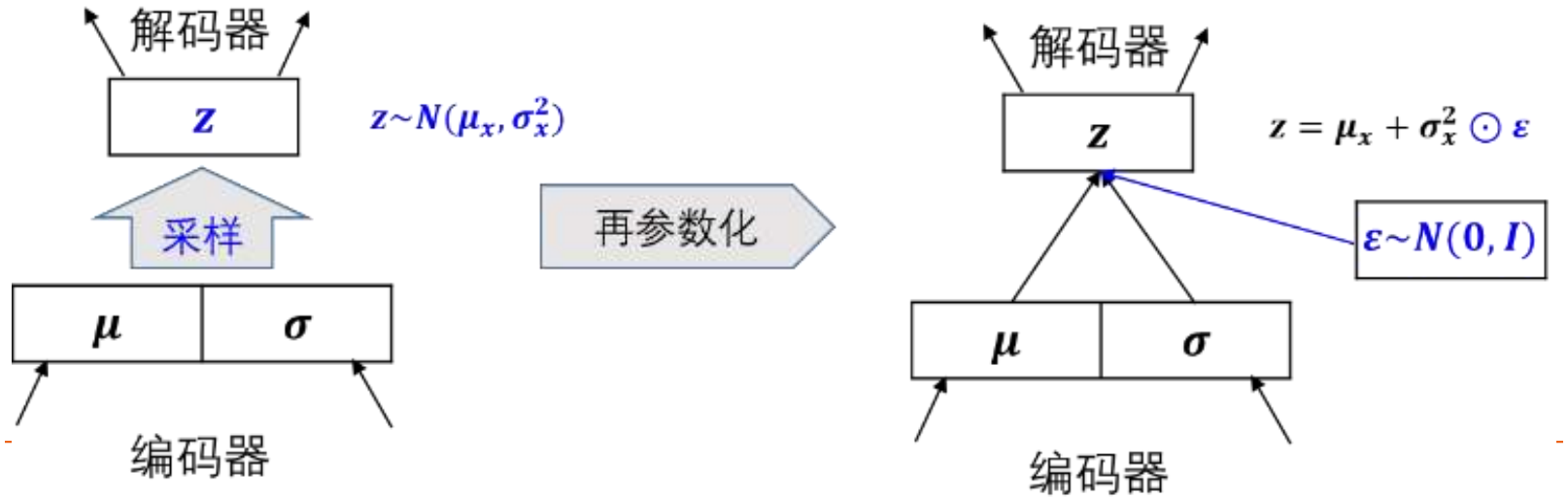
变分自编码器（VAE）

- 由于中间存在采样，梯度无法传播到编码层
- 解决办法：**“再参数化技巧”**
  - 原模型中解码层输入向量z通过采样得到：

$$z|x \sim N(\mu_x, \sigma_x^2)$$

  - 再参数化，系数ε保留了随机性，但不是训练的参数，使得梯度可以传播：

$$z|x = \mu_x + \sigma_x^2 \odot \varepsilon, \qquad \varepsilon \sim N(0, I)$$



VAEs: Latent perturbation

Slowly increase or decrease a **single latent variable**
Keep all other variables fixed

Head pose

Different dimensions of z encodes **different interpretable latent features**

代码实现：

```python
class VAE(nn.Module):

    def __init__(self, encoder_layer_sizes, latent_size,
                 conditional=False, num_labels=0):

        super().__init__()

        if conditional:
            assert num_labels > 0

        assert type(encoder_layer_sizes) == list
        assert type(latent_size) == int
        assert type(decoder_layer_sizes) == list

        self.latent_size = latent_size

        self.encoder = Encoder(encoder_layer_sizes, later
        self.decoder = Decoder(decoder_layer_sizes, later
```

```python
    def reparameterize(self, mu, log_var):

        std = torch.exp(0.5 * log_var)
        eps = torch.randn_like(std)

        return mu + eps * std


    def inference(self, z, c=None):

        recon_x = self.decoder(z, c)

        return recon_x
```

```python
    def forward(self, x, c=None):

        if x.dim() > 2:
            x = x.view(-1, 28*28)

        means, log_var = self.encoder(x, c)
        z = self.reparameterize(means, log_var)
        recon_x = self.decoder(z, c)

        return recon_x, means, log_var, z
```

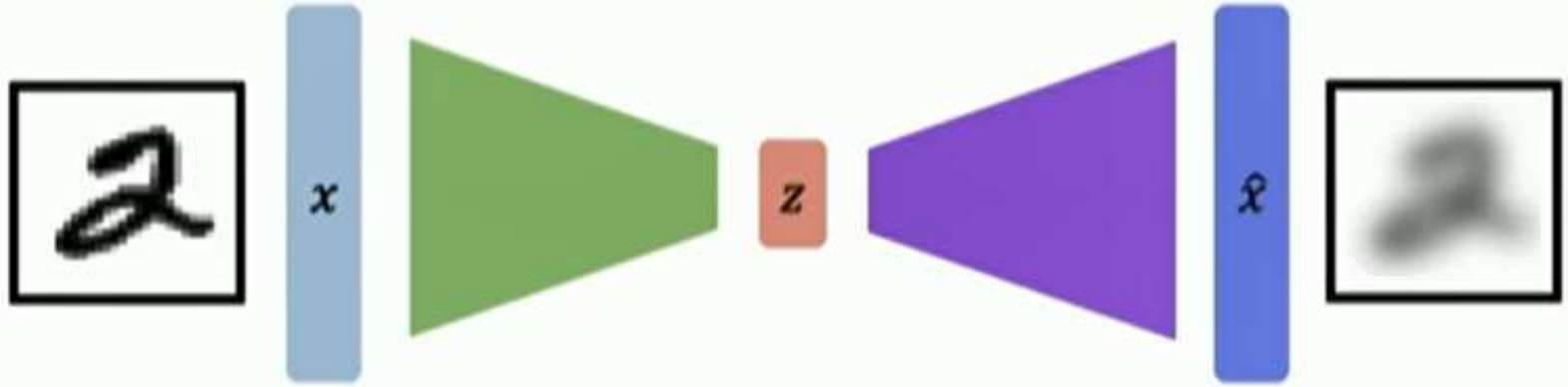
代码实现：

```python
class Encoder(nn.Module):

    def __init__(self, layer_sizes, latent_size, conditional, r

        super().__init__()

        self.conditional = conditional
        if self.conditional:
            layer_sizes[0] += num_labels

        self.MLP = nn.Sequential()

        for i, (in_size, out_size) in enumerate(zip(layer_sizes
            self.MLP.add_module(name="L{:d}".format(i), module=
            self.MLP.add_module(name="A{:d}".format(i), module=

        self.linear_means = nn.Linear(layer_sizes[-1], latent_s
        self.linear_log_var = nn.Linear(layer_sizes[-1], latent

    def forward(self, x, c=None):

        if self.conditional:
            c = idx2onehot(c, n=10)
            x = torch.cat((x, c), dim=-1)

        x = self.MLP(x)

        means = self.linear_means(x)
        log_vars = self.linear_log_var(x)

        return means, log_vars
```

```python
class Decoder(nn.Module):

    def __init__(self, layer_sizes, latent_size, conditional, num_labels):

        super().__init__()

        self.MLP = nn.Sequential()

        self.conditional = conditional
        if self.conditional:
            input_size = latent_size + num_labels
        else:
            input_size = latent_size

        for i, (in_size, out_size) in enumerate(zip([input_size]+layer_sizes[:-1], layer_sizes)):
            self.MLP.add_module(name="L{:d}".format(i), module=nn.Linear(in_size, out_size))

            if i+1 < len(layer_sizes):
                self.MLP.add_module(name="A{:d}".format(i), module=nn.ReLU())
            else:
                self.MLP.add_module(name="sigmoid", module=nn.Sigmoid())

    def forward(self, z, c):

        if self.conditional:
            c = idx2onehot(c, n=10)
            z = torch.cat((z, c), dim=-1)

        x = self.MLP(z)

        return x
```

VAE summary

1. Compress representation of world to something we can use to learn
2. Reconstruction allows for unsupervised learning (no labels!)
3. Reparameterization trick to train end-to-end
4. Interpret hidden latent variables using perturbation
5. Generating new examples



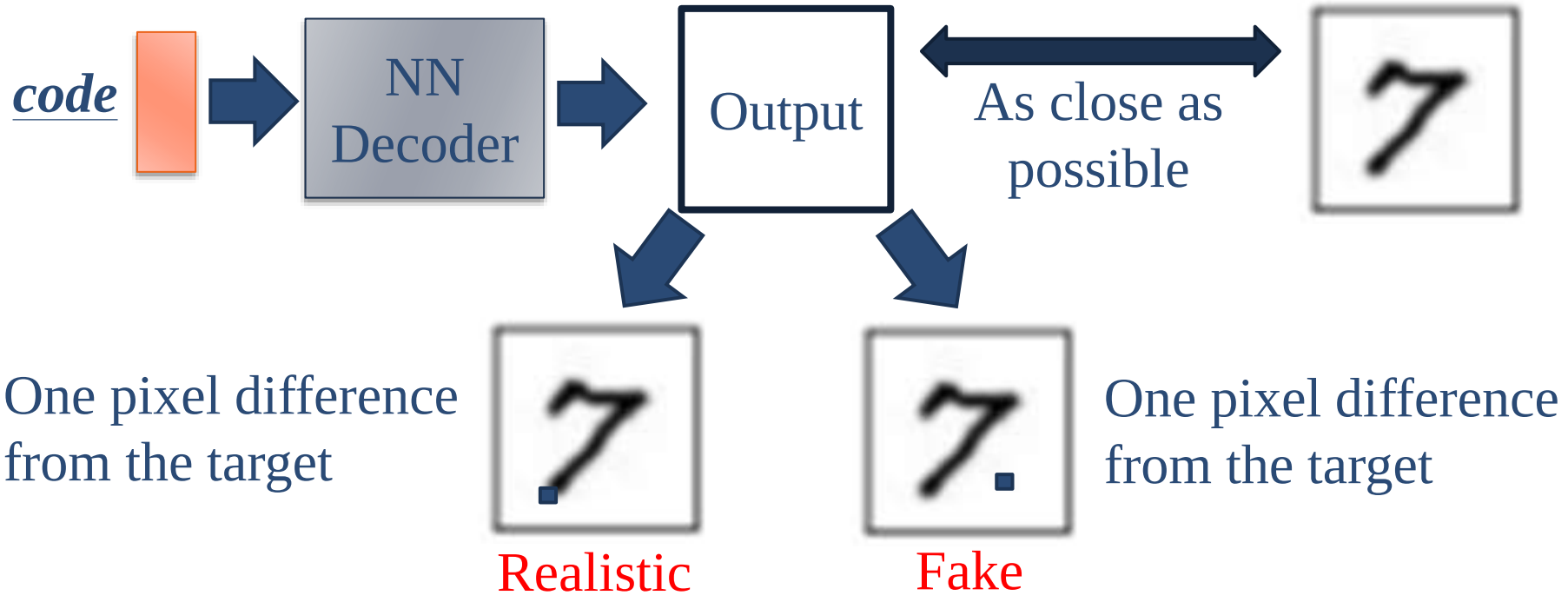
VAE 相对于传统的AE的优势和解决的问题：

1. **生成新样本能力**： VAE 是一种生成模型，可以学习数据的分布，因此可以用来生成与训练数据相似但不完全相同的新样本。传统的AE通常不具备这种生成能力。
2. **概率性表达**： VAE 引入了概率性，可以在潜在空间中对样本进行采样，从而允许对样本的不确定性进行建模。传统的AE只能生成一个确定的解码结果。
3. **连续的潜在表示**： VAE 的潜在空间通常是连续的，这使得它可以对样本之间的关系进行建模，比如在潜在空间中进行插值可以得到连续的过渡效果。传统的AE的潜在空间可能是离散的或者不可解释的。
4. **抽象特征表示**： 由于 VAE 引入了概率性，它可以学习到更具有意义的抽象特征表示，而传统的AE可能会过度拟合训练数据。
5. **自编码器的正则化**： VAE 使用了变分推断技巧来优化目标函数，这可以被看作是一种正则化方法，有助于避免过拟合。

总的来说，VAE 在生成模型中引入了概率性，使得它能够对数据的分布进行建模，从而在生成新样本、学习抽象特征表示等方面有明显的优势。然而，VAE 也有一些挑战，比如训练过程中可能会涉及复杂的变分推断等技巧，需要仔细地设计网络结构和损失函数。

Problems of VAE

- **It does not really try to simulate real images**



VAE may just memorize the existing images, instead of generating new images.

谢谢！