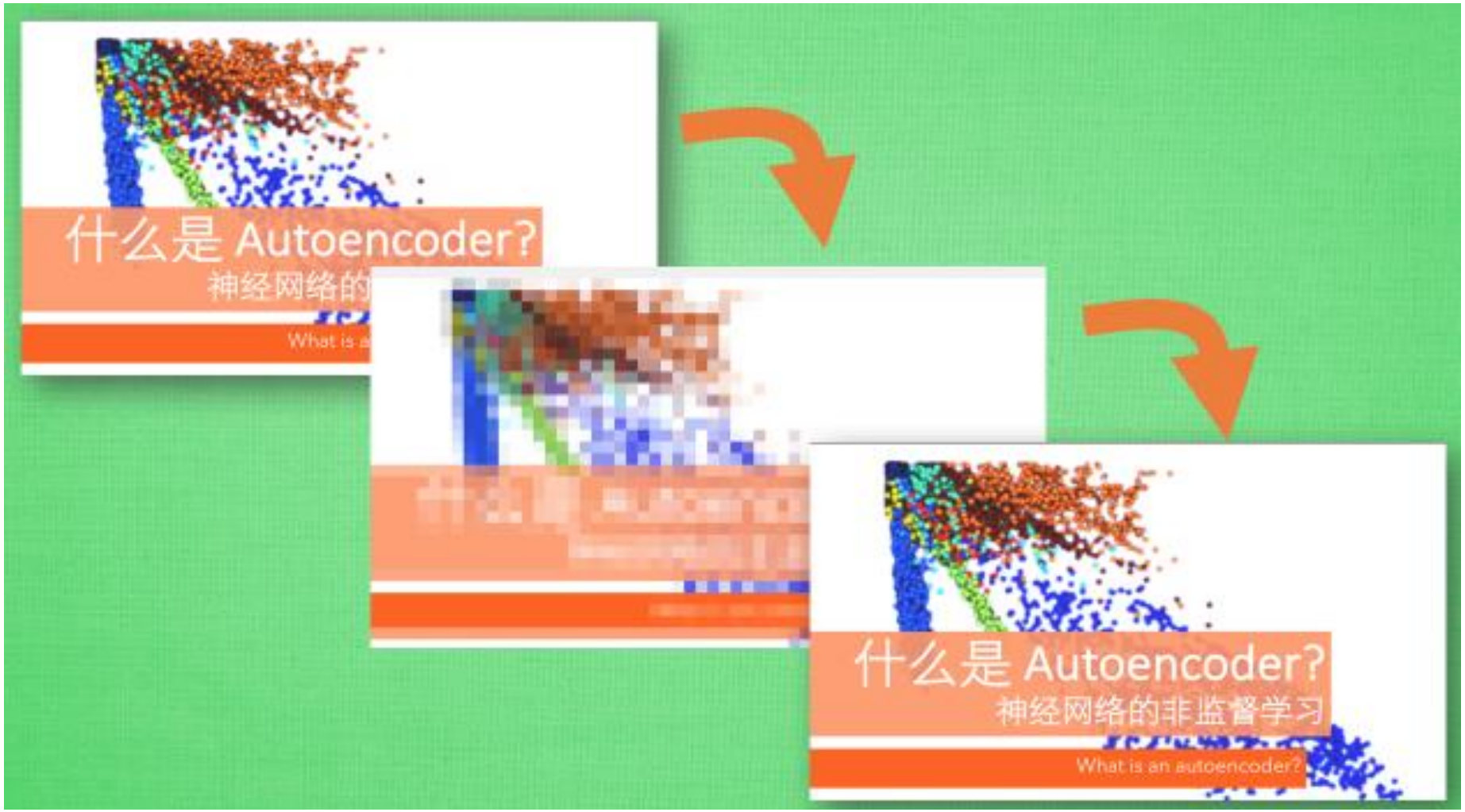
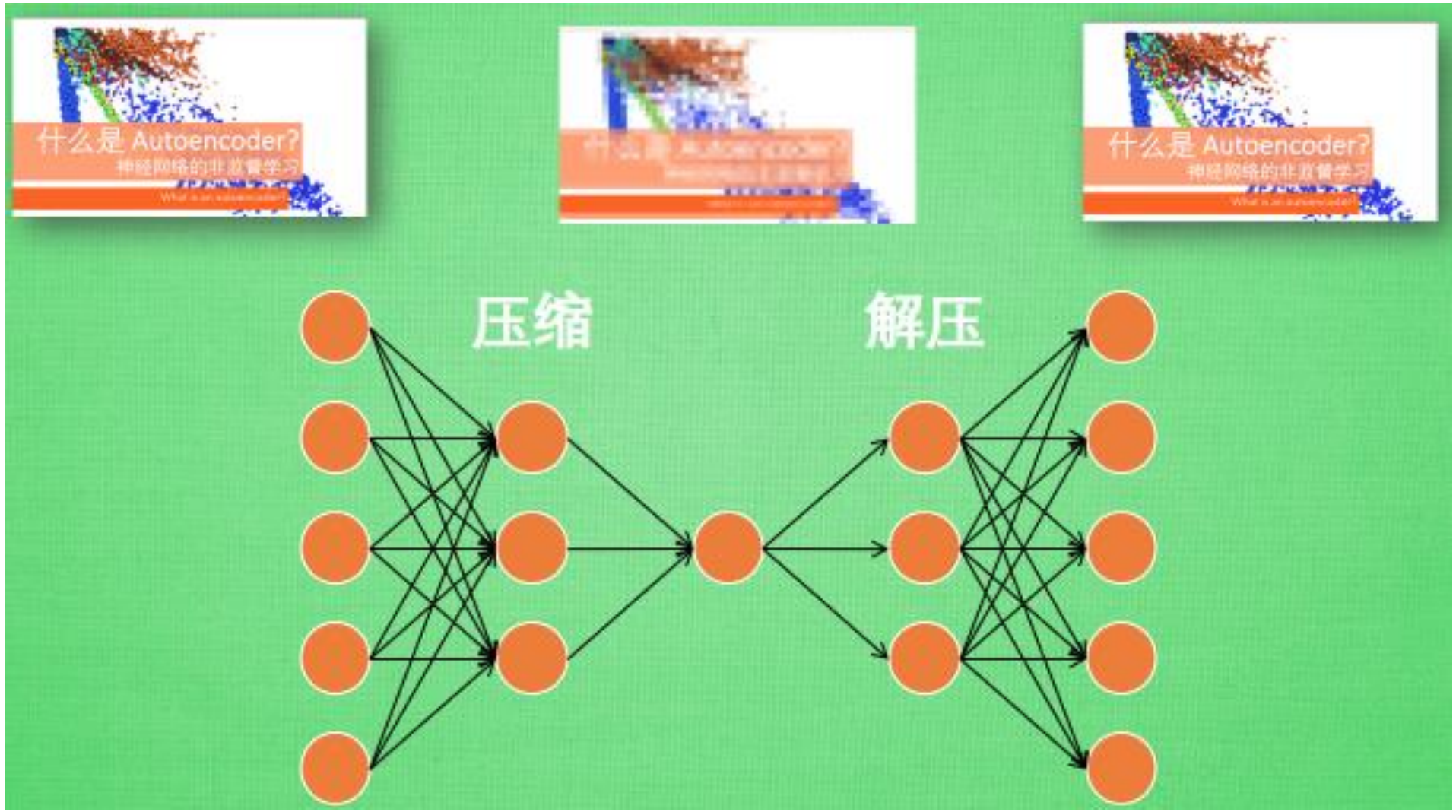
自编码器(AutoEncoder)

一、什么是自编码器?

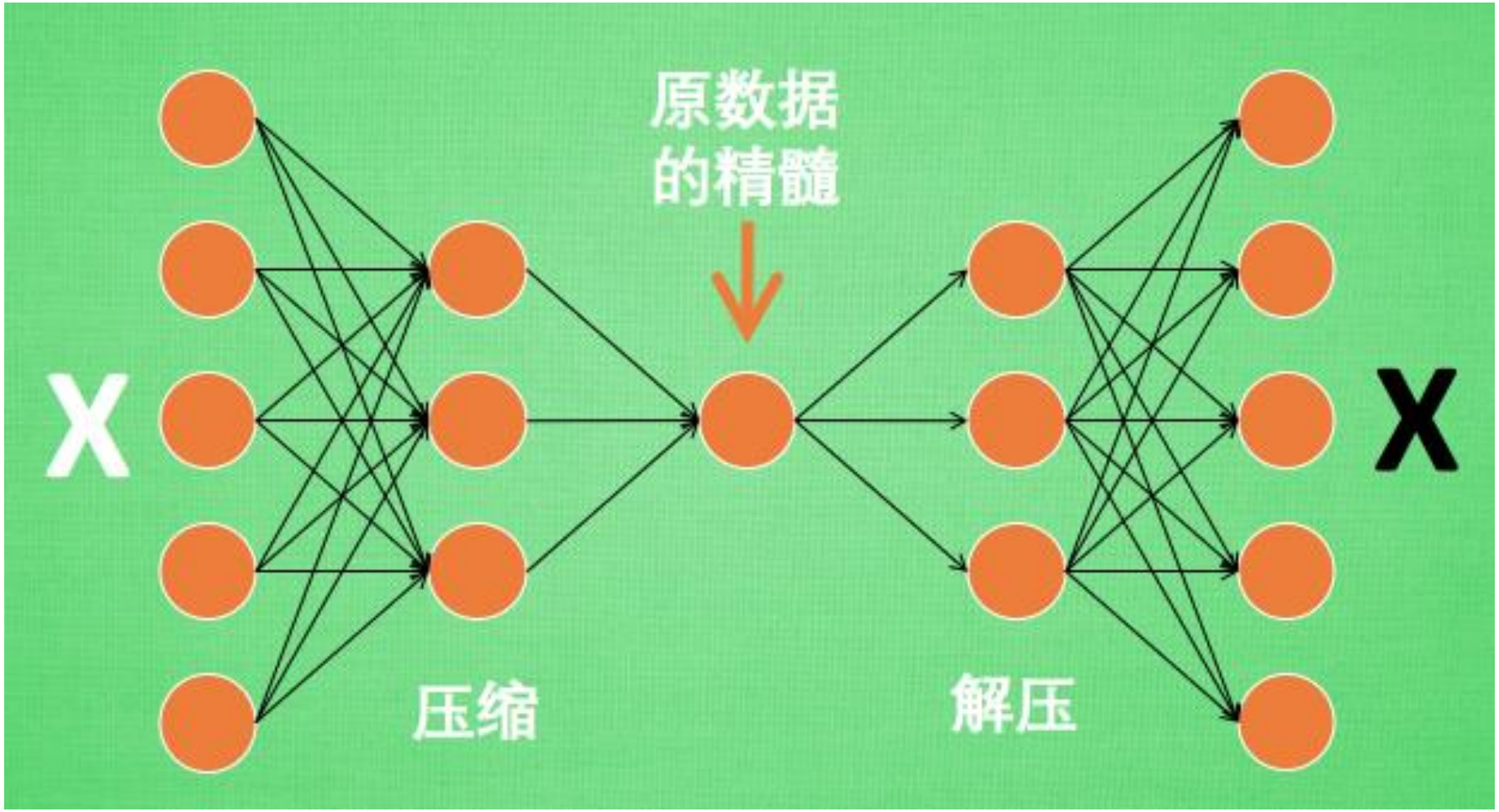
- 1、自编码器：即可以使用自身的高阶特征来编码自己。自编码器其实也是一种神经网络，它的输入和输出是一致的，它借助稀疏编码的思想，目标是使用稀疏的一些高阶特征重新组合来重构自己。因此，它的特点非常明显：
  - 1)期望输入/输出一致
  - 2)希望用高阶特征来重构自己，而不只是复制像素点

- 接收一张图片——打码——还原

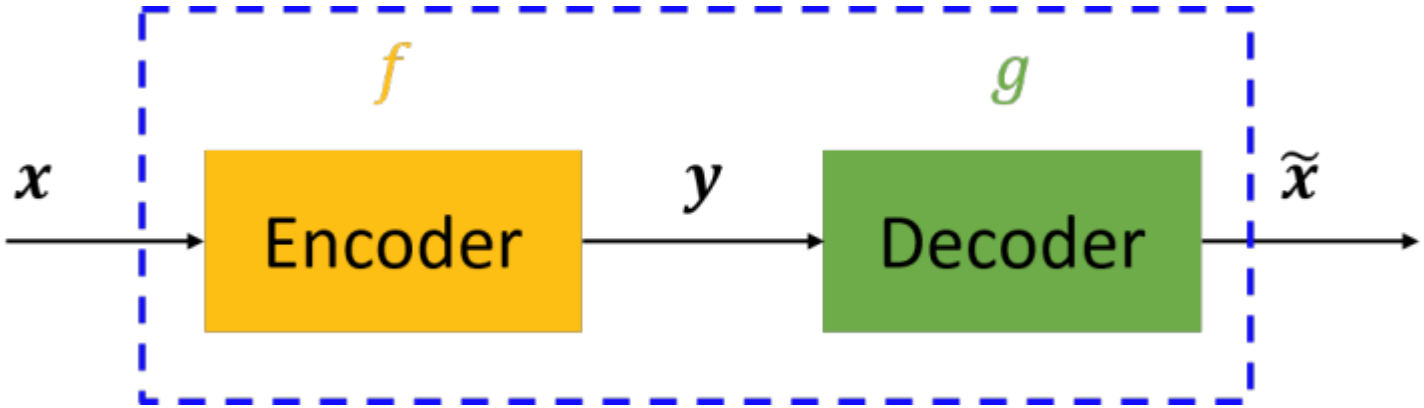

什么是 Autoencoder?
神经网络的
What is a
什么是 Autoencoder?
神经网络的非监督学习
What is an autoencoder?


什么是 Autoencoder?
神经网络的非监督学习
什么是 Autoencoder?
神经网络的非监督学习
压缩
解压

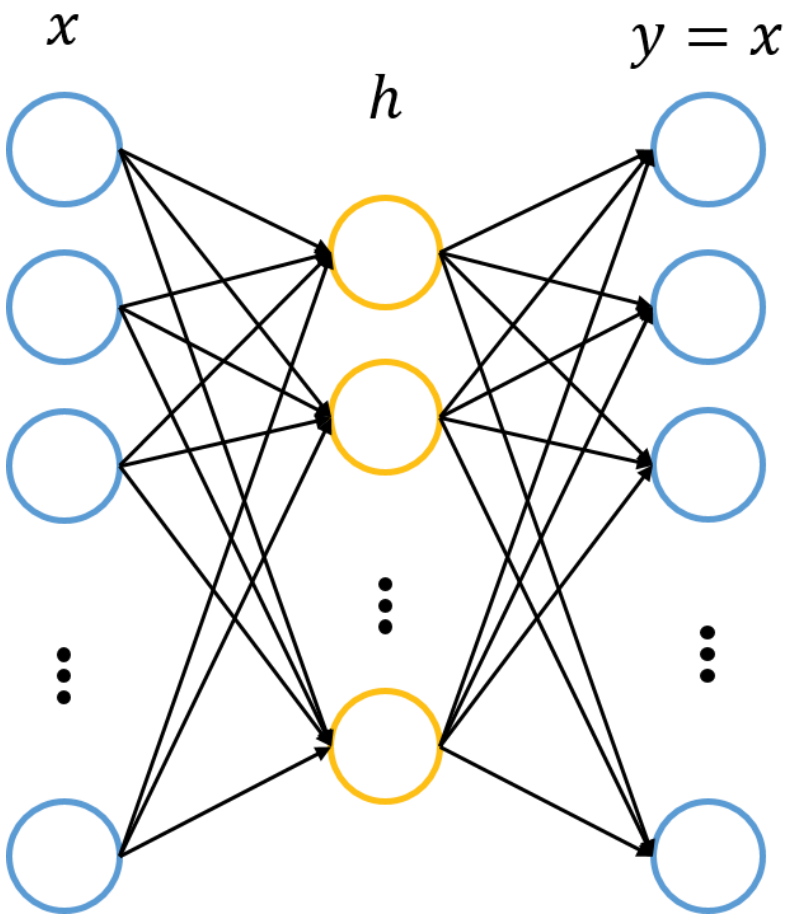
- 当压缩的时候，原有的图片质量被缩减，解压时用信息量小却包含了所有关键信息的文件恢复出原本的图片



- 2、搭建一个自动编码器需要完成下面三样工作：
- 1)搭建编码器：这部分**能将输入压缩成潜在空间表征**，可以用编码函数 $y = f(x)$ 表示。
- 2)搭建解码器：这部分**重构来自潜在空间表征的输入**，可以用解码函数 $\tilde{x} = g(y) = g(f(x))$ 表示。
- 3)设定一个损失函数，用以衡量由于压缩而损失掉的信息。



- 结合神经网络：
  - **网络中输出应和输入相同**，即y=x，因此自编码器的输入、输出应用相同相同的结构。
  - **限制中间隐含层节点的数量**，使用少量稀疏的高阶特征来重构输入。利用训练数据训练这个网络，训练结束后，这个网络学习出了x→h→x的能力。h是至关重要，在尽量不损失信息量的情况下，对原始数据的另一种表达。



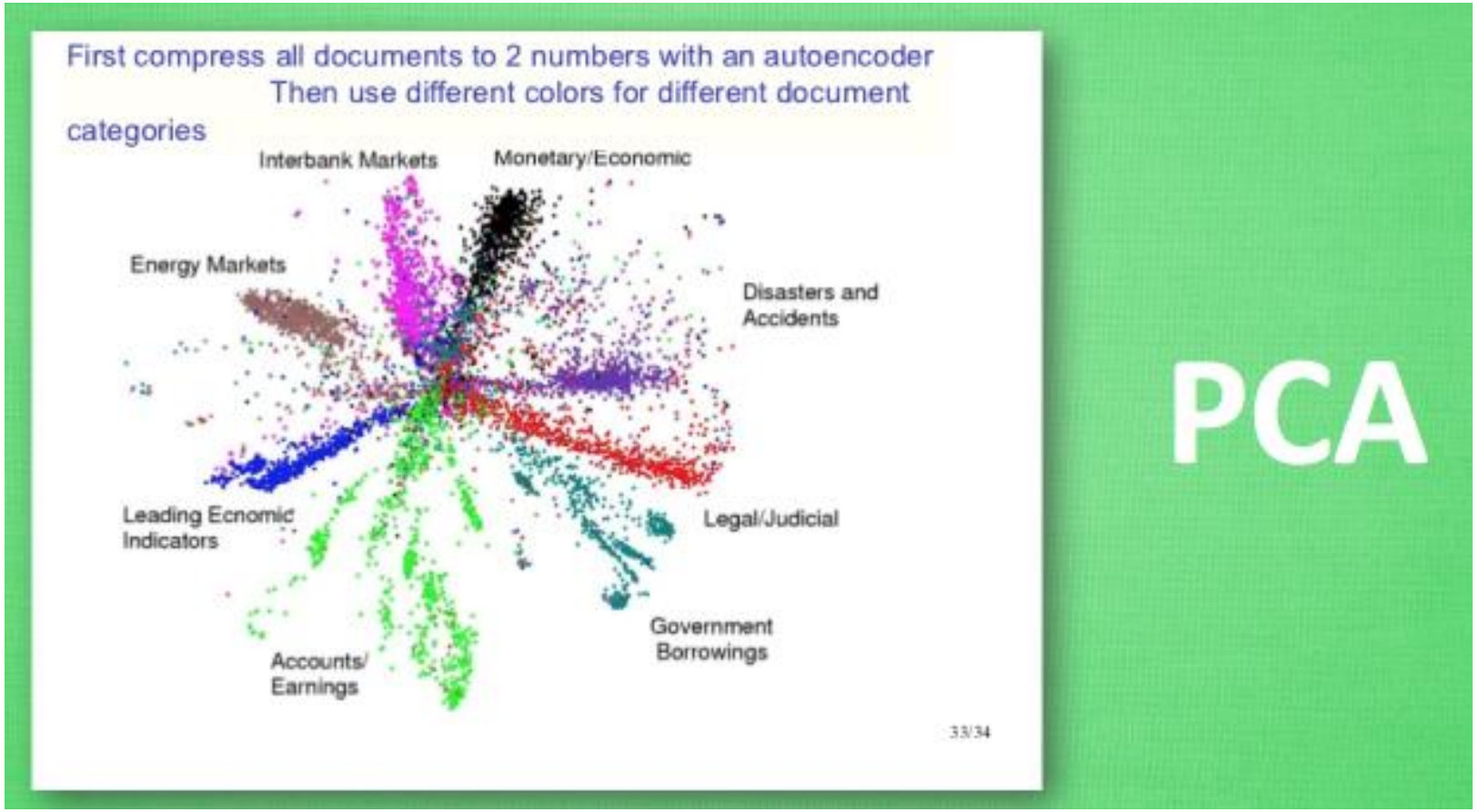
公式表达如下：

$$y = f\theta(x) = f(Wx + b)$$

$$\tilde{x} = g\theta'(y) = f(W'y + b')$$

$$L(x, \tilde{x}) = L(x, g(f(x)))$$

- 1）Encoder 编码器，编码器能得到原数据的精髓，然后我们只需要再创建一个效的神经网络学习这个精髓的数据，不仅减少了神经网络的负担，而且同样能达到很好的效果。



- 自编码可以像 PCA 一样给特征属性降维。

- 定义编码器函数：

```
# Building the encoder
def encoder(x):
    # Encoder Hidden layer with sigmoid activation #1
    layer_1 =
    # Eecoder Hidden layer with sigmoid activation #2
    layer_2 =
    return layer_2
```

- 2）Decoder 解码器，解码器在训练的时候是要将精髓信息解压成原始信息, 那么这就提供了一个解压器的作用, 甚至我们可以认为是一个生成器 (类似于GAN). 那做这件事的一种特殊自编码叫做 Variational Autoencoders（VAE）。参考网址：http://kvfrans.com/variational-autoencoders-explained/

- 定义编码器函数：

```
# Building the decoder
def decoder(x):
    # Dncoder Hidden layer with sigmoid activation #1
    layer_1 =
    # Decoder Hidden layer with sigmoid activation #2
    layer_2 =
    return layer_2
```

- 3)设定损失函数

```
# Define loss and optimizer, minimize the squared error
cost =tf.reduce_mean(tf.pow(y_true - y_pred, 2))
optimizer =tf.train.AdamOptimizer(learning_rate).minimize(cost)
```

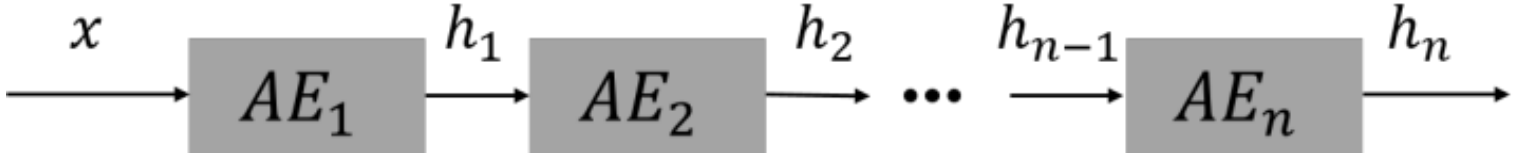
二、自编码器作用

- 1、数据去噪
  - Pascal Vincent的论文Stacked Denoising Autoencoders：Learning Useful Representations in a Deep Networks with a local Denoising Criterion.
- 2、进行可视化降维
  - Hinton教授在Science发表文章Reducing the dimensionality of data with neural networks，讲解了使用自编码器对数据进行降维的方法。
- 3、预训练与深度学习
  - 1）layer-wise pre-training（逐层预训练）
  - 2）fune-tuning（微调）

三、自编码器的变种形式

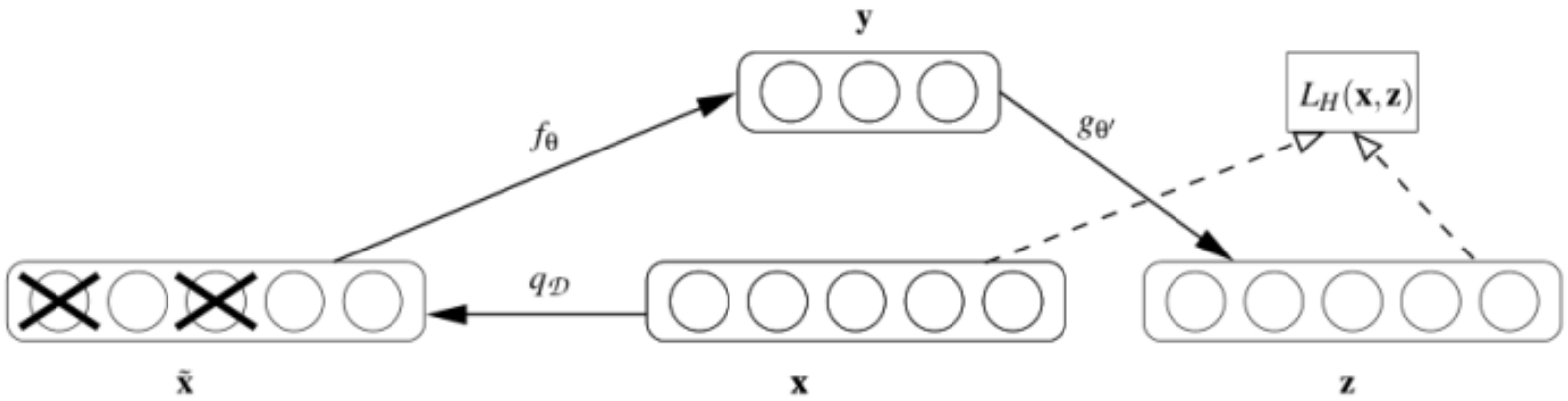
- 1)堆叠自编码器（Stacked Auto-Encoder,SAE)
  - 单个自编码器通过虚构x→h→x三层网络，能够学习一种特征变换$h = f\theta(x)$
    实际上当训练结束后，输出层已经没什么意义了，一般将其去掉。

  - 将h再当作原始信息，训练一个新的自编码器，得到新的表达。

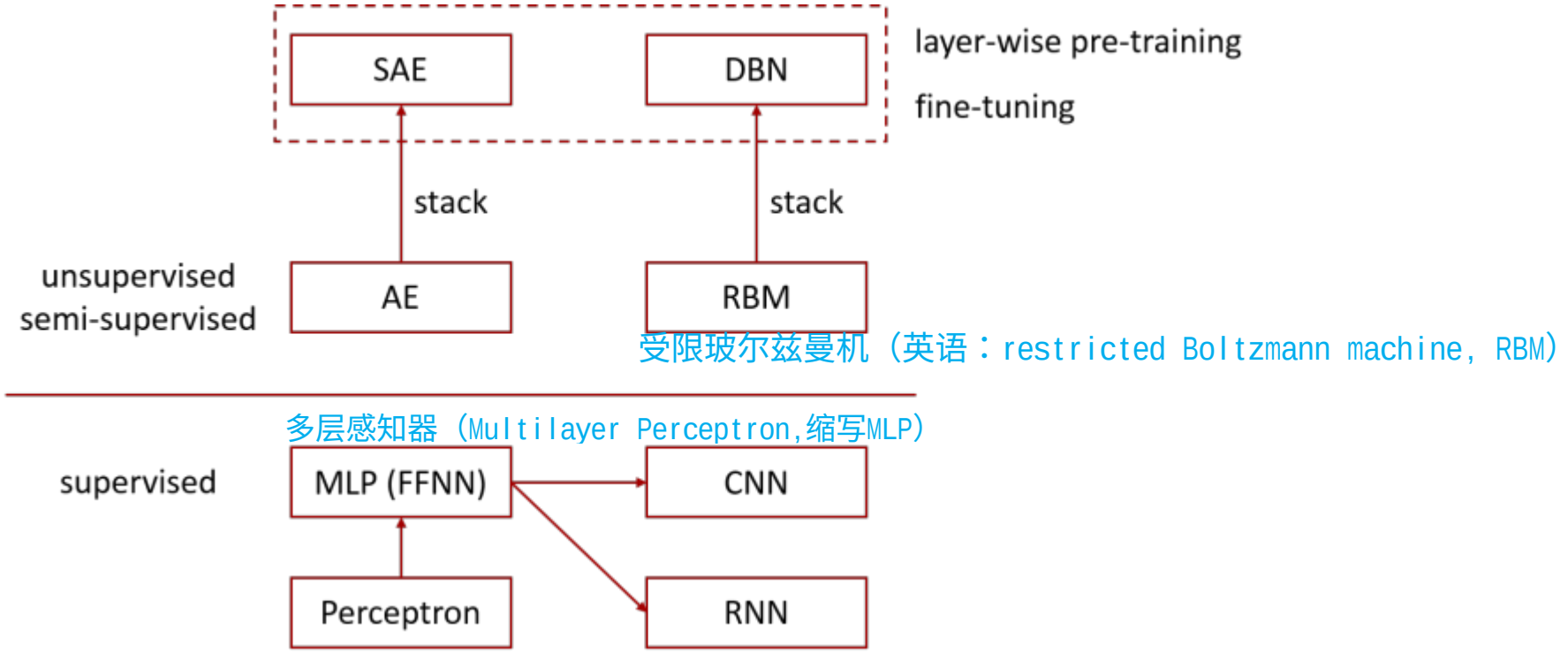


- 2)稀疏自编码器(Sparse AE)
  - 高维而稀疏的隐层表达式好的。隐藏表达h中一些节点是被抑制的（对于sigmoid单元即输出为0），指定一个稀疏性参数 ρ ，代表隐藏神经元的平均活跃程度。
  - 目标函数添加惩罚项：$\sum_{i \in h} KL(\rho \| \tilde{\rho}_a)$
    - KL散度，衡量神经元的实际激活度 $\tilde{\rho}_a$ 与期望激活度 ρ 之间的差异。

- 3)降噪自编码器(Denoising AE)
  - DAE核心思想，一个能够从中恢复出原始信号的表达未必是最好的，能够对“被污染/破坏”的原始数据编码、解码，然后还能恢复真正的原始数据，这样的特征才是好的。
  - 监督的误差从 $L(x,g(f(x)))$ 变为 $L(x,g(f(\tilde{x})))$ ，$\tilde{x}$ 为被破坏的数据。
  - DAE希望学到的特征变换尽可能鲁棒，能够在一定程度上对抗原始数据的污染、缺失。



- 各模型在神经网络/深度学习框架中的位置，简单总结



第二次作业：

- 1、在MNIST数据集上通过AE重构数字

  网络结构：
  - 压缩环节：784→256→128（使用sigmoid激活函数）
  - 解压环节：128→266→784（使用sigmoid激活函数）
  - 损失函数：原始数据与还原后的拥有784Features的数据进行cost对比。

- 2、可视化降维:显示Encoder之后数据（）

  网络结构：
  - 压缩环节：784→128→64→10→2 (encode最后一层不使用激活函数)
  - 解压环节：2→10→64→128→784

```
plt.scatter()
plt.colorbar()
plt.show()
```