

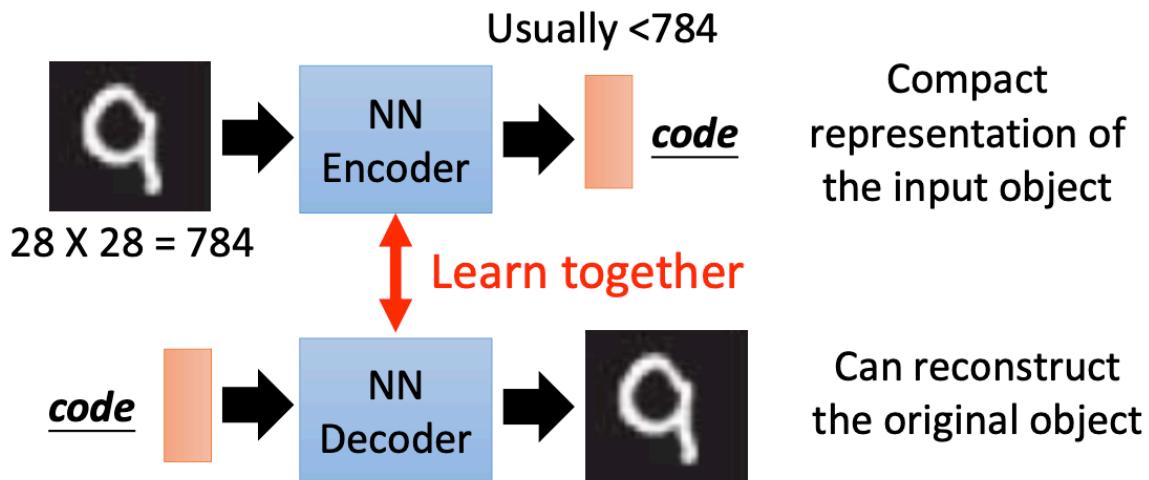
Auto-encoder

对于输入大小为 28×28 的图像，先经过encoder进行编码，得到的结果通常比784维要小，code表示比原来的image更加精简（compact）的特征；

但现在我们进行的是unsupervised learning，这个code到底长什么样子也不知道；

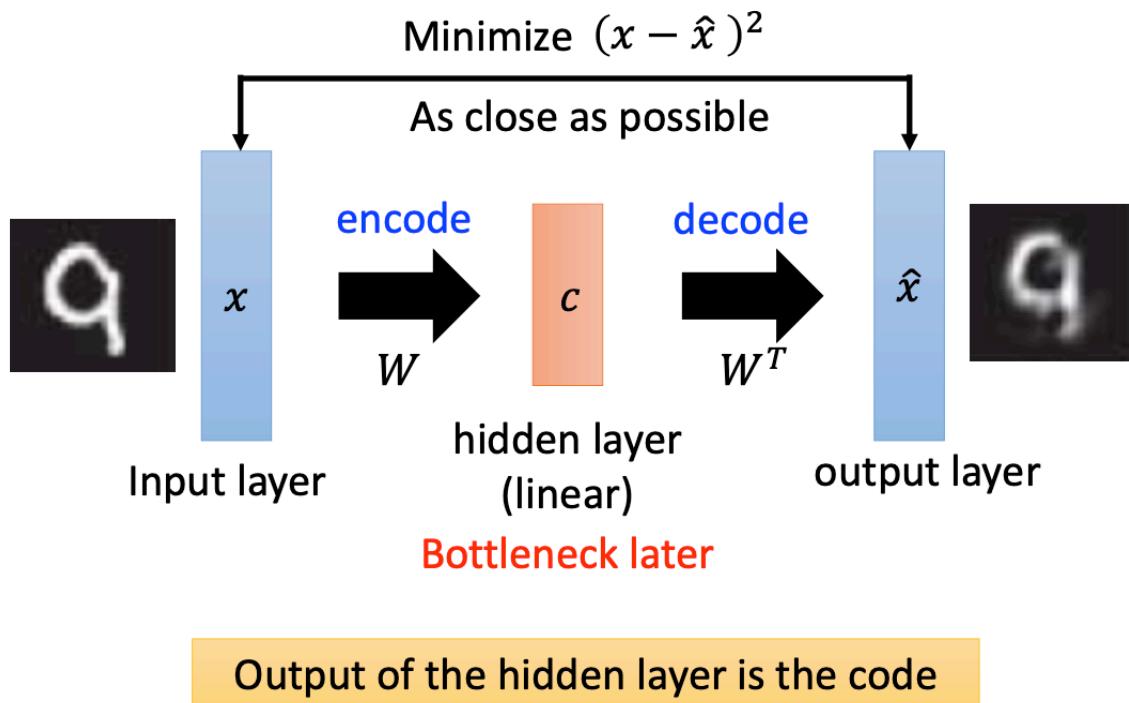
那么我们现在就先来做一个decoder，把code恢复成一张image；

decoder和encoder单独是没办法训练的，因为是无监督学习，因此现在我们将这两者一起学习



Recap: PCA

这里先回顾一下[PCA](#)，对于输入的图像 x 为784维，经过PCA降维，可以降低到324维的图像 \hat{x} ，在encoder和decoder的学习过程中，应达到的目标是最小化 $(x - \hat{x})^2$ ，这个结构可以看成是具有一个hidden layer的network



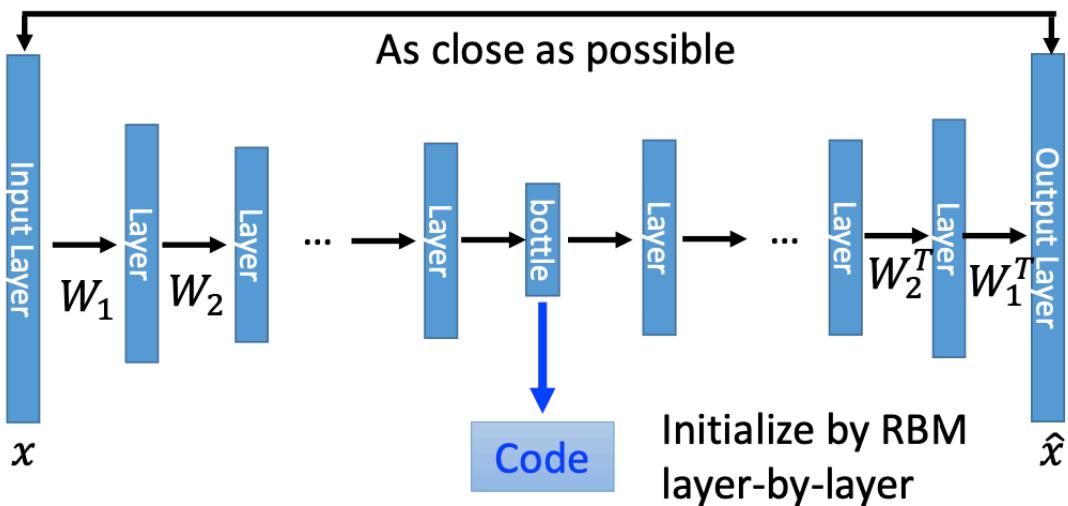
Deep Auto-encoder

Structure

中间的hidden layer也可以不止一个，也可以是多个的hidden layer，这种结构就称作deep auto-encoder

注：encoder和decoder的结构不一定非得是对称的。

- Of course, the auto-encoder can be deep

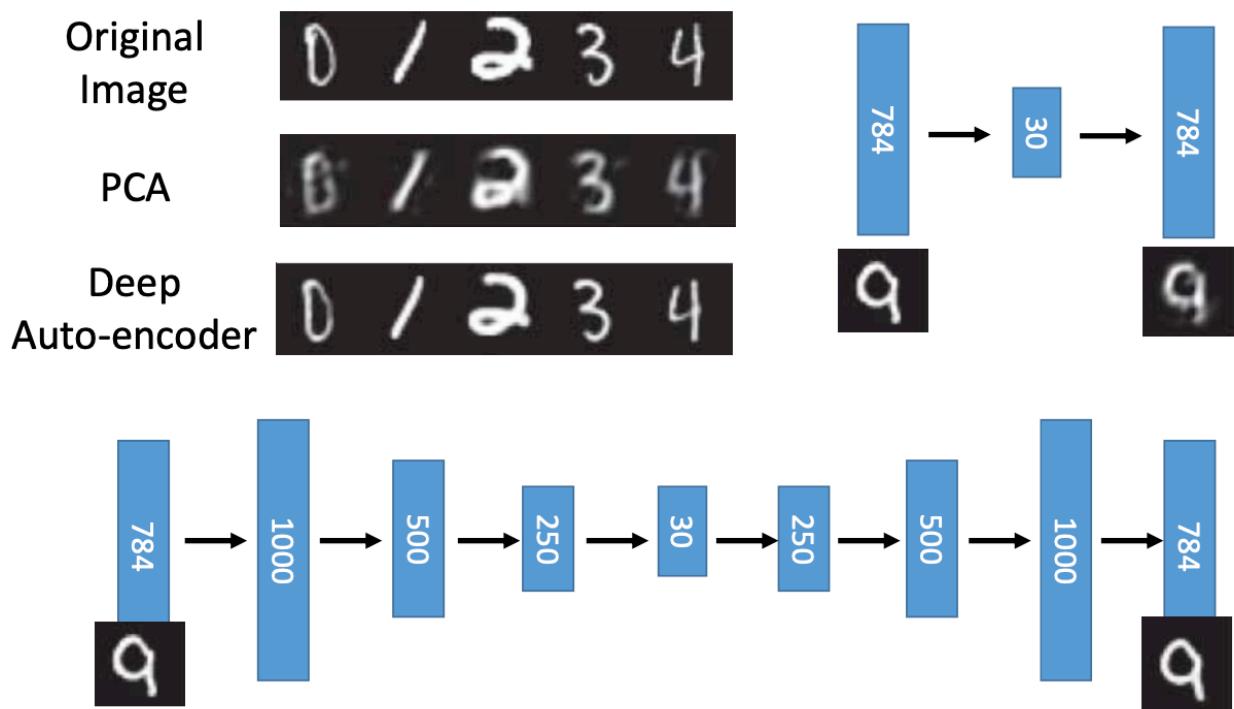


Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

PCA. vs Deep Auto-encoder

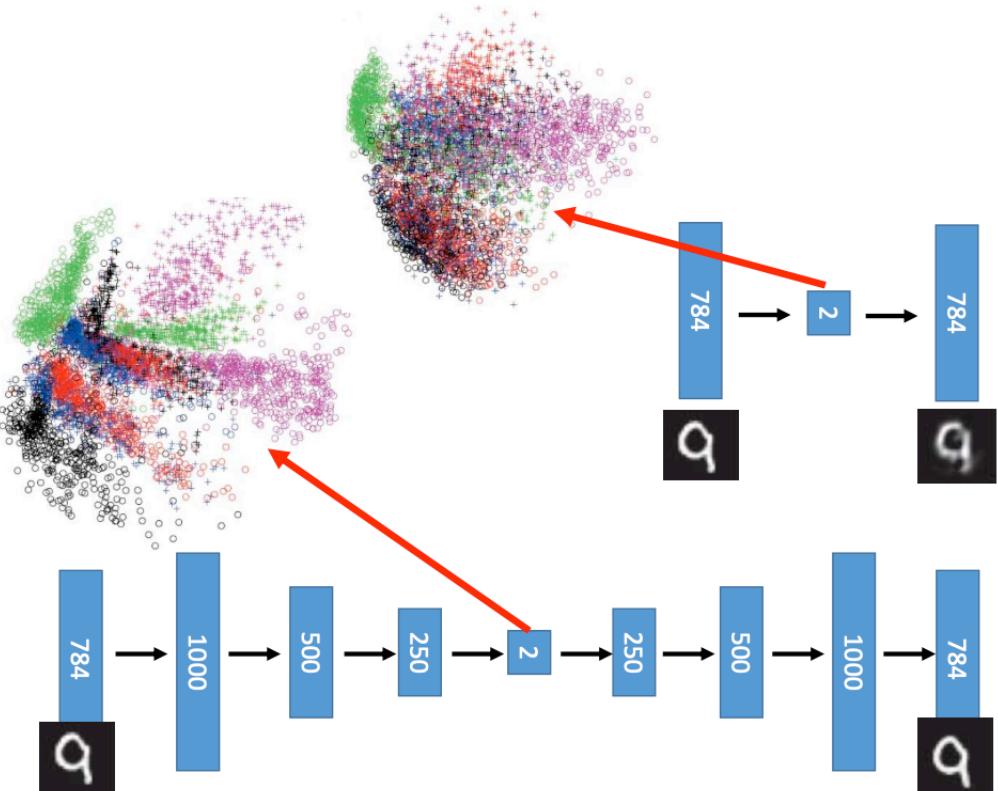
对于下图中的图像(0,1,2,3,4)，如果我们使用PCA，只有中间一个hidden layer，先从784维降到30维，再从30维恢复到784维，可以发现图像变得比较模糊；

如果使用deep auto-encoder, 先从784到1000, 1000到500, 500到250, 250到300, 再使用类似的encoder恢复图像, 可以发现结果图像非常清晰



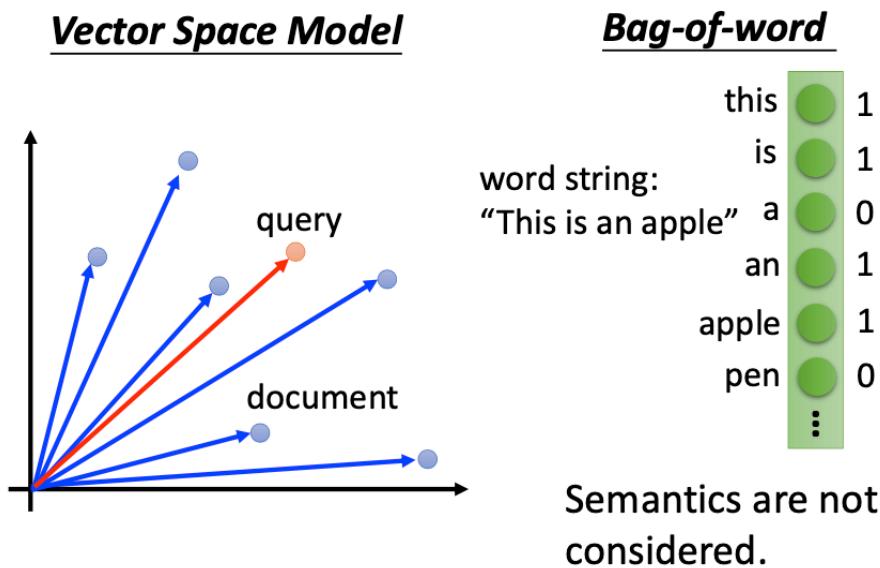
如果我们先使用PCA进行降维, 把原图从784降到2维, 对二维的数据进行可视化, 可以发现不同的digit (不同的颜色代表不同的数字) 都叠在一起了;

如果使用deep autoencoder, 可以发现这个数字都是分开的



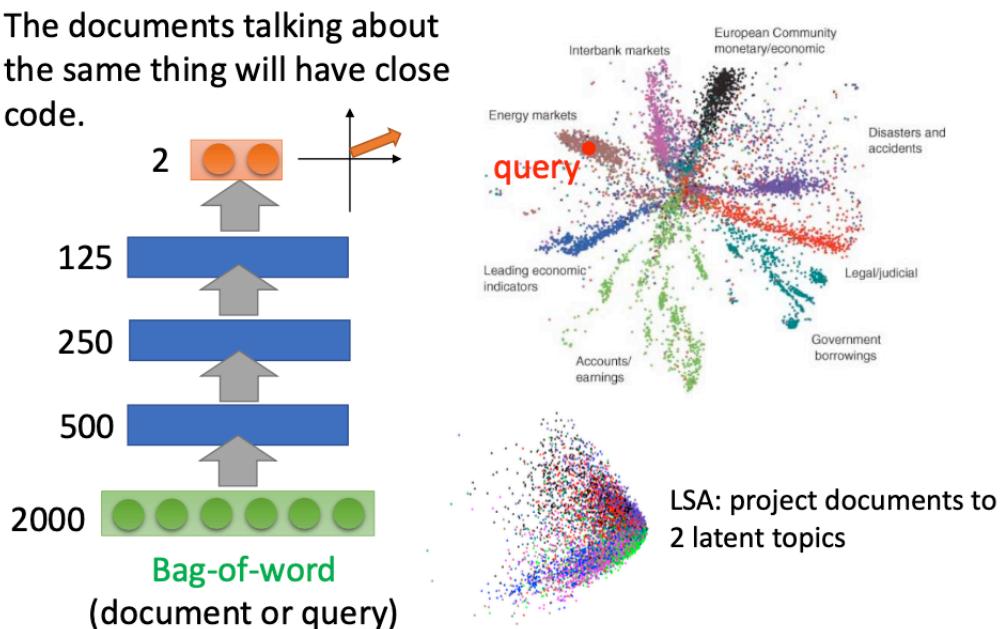
Text Retrieval

下图中蓝色的圆点都表示一个document，我们将输入的query也加入这个document，再计算查询的词汇query和每个document之间的inner product或similarity等，距离最近的document，similarity的值是最大的，因此会retrieval距离红色箭头最近的其他两个蓝色箭头



还有另外一种方法，使用一个vector来表示所有的词汇，vector对应的值就是每个character出现的次数；但这种方式没有考虑原来的语义顺序，每个character都是independent的

在下图中，假设bag里面有2000个词汇，把输入的document或query变成对应的vector，再输入相应的encoder，降维成2维，对这二维的数据进行可视化，如右图所示，不同的颜色代表不同的document；



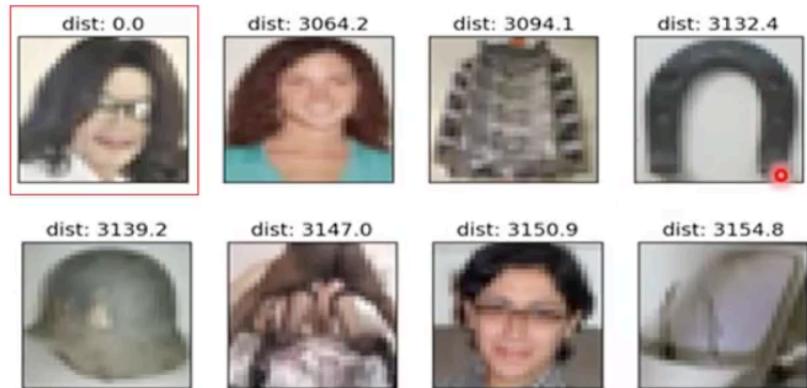
如果我们现在输入一个query，再降维到二维，可以发现和图中对应的document是有关的，我们就可以看到query所属的类别是Energy market

但用LSA就得不到对应的结果

Similar Image Search

以图找图，如果我们只是做pixel上的相似程度，那么我们可以得到以下的结果，迈克杰克逊和马蹄铁也很像，这显然不符合常理

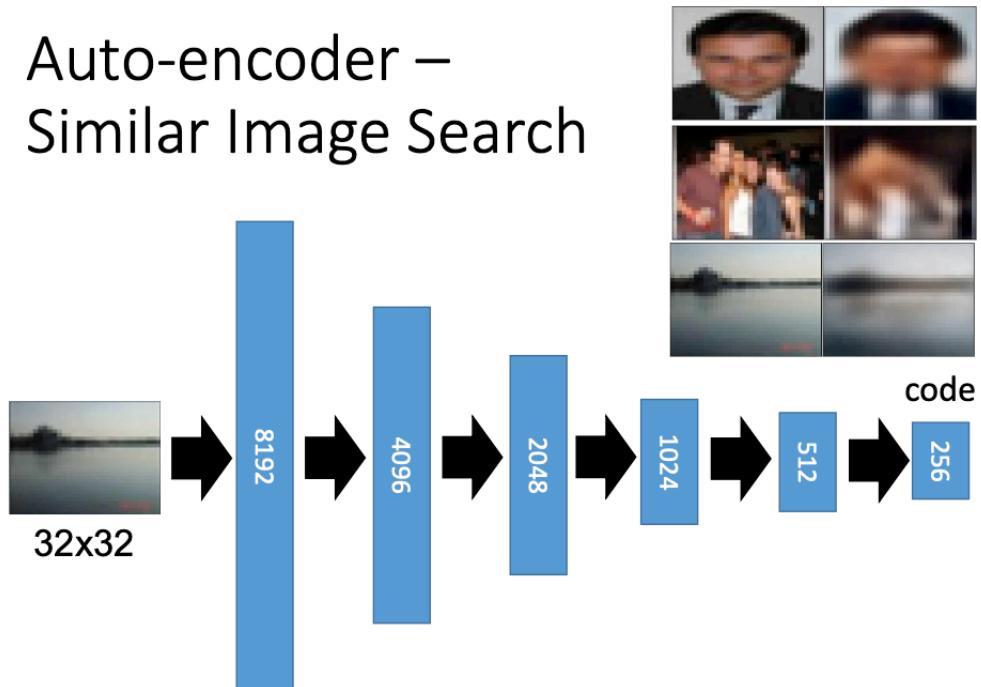
Retrieved using Euclidean distance in pixel intensity space



(Images from Hinton's slides on Coursera)

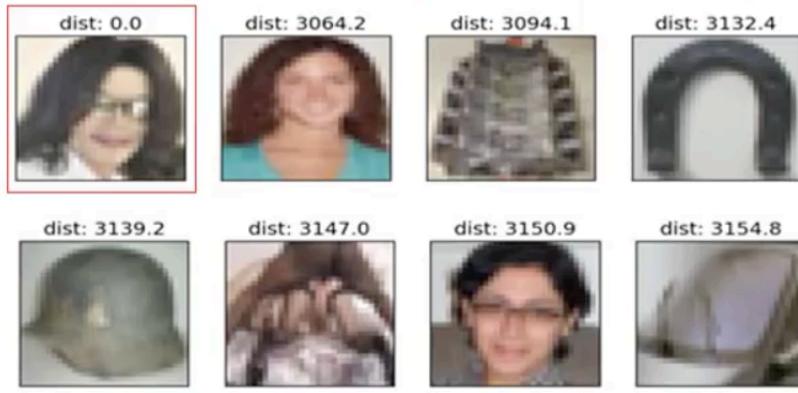
我们可以把输入的image经过deep auto-encoder，变成一个code，再去做搜寻；由于auto-encoder是unsupervised learning，收集多少数据都行，不缺数据

Auto-encoder – Similar Image Search

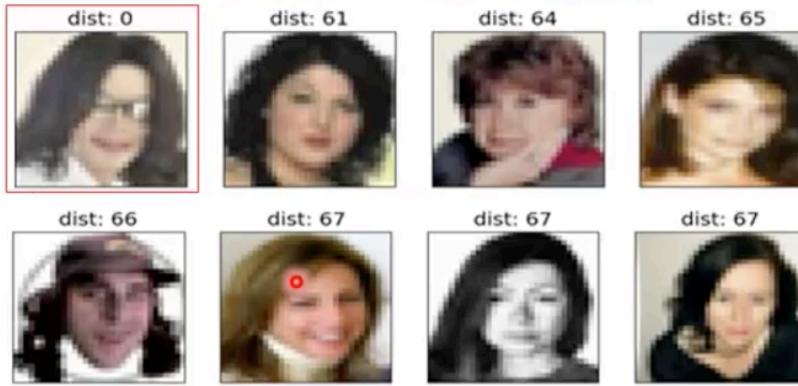


用deep auto-encoder来找类似迈克杰克逊，也可以得到更好的结果（至少全是人脸）

Retrieved using Euclidean distance in pixel intensity space



retrieved using 256 codes



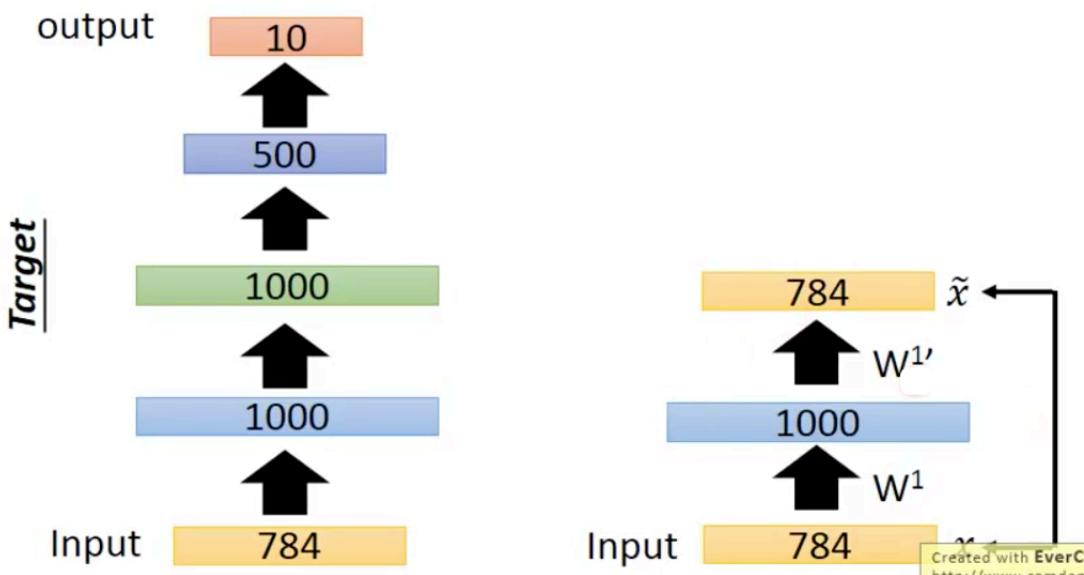
Pre-training DNN

autoencoder也可以用到network的pre-training过程，来初始化所需要的weight；在下图中，如果我们要对第一个hidden layer的weight进行初始化，那么我们可以使用autoencoder，先将784维到1000维，再进行reconstruct，使1000维降到784维，来使 x, \hat{x} 之间的差值最小

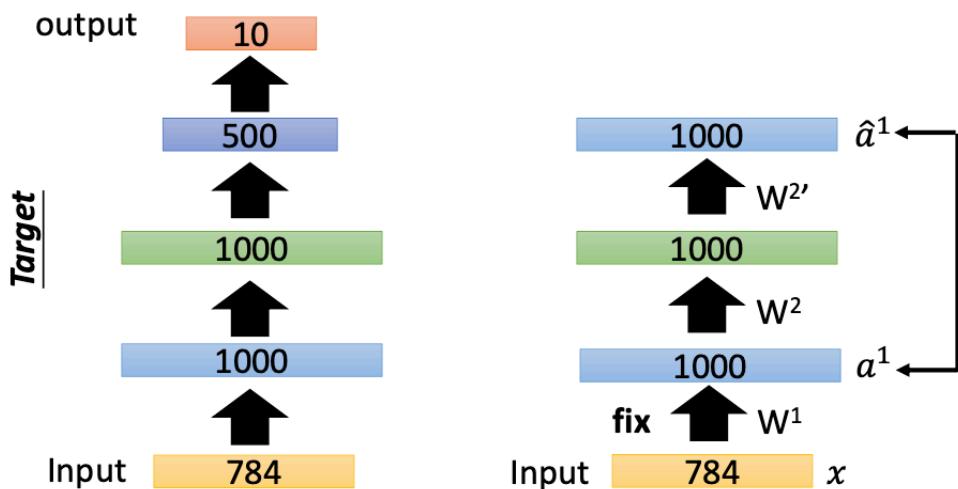
但这样可能会出现一个问题，由于1000维比784维更高维，autoencoder很可能将input直接embed到1000维里，再进行恢复，这样很可能network什么都没有学到；

对于这个问题，我们可以把1000维再加上一个regularization，比如L1，这1000维的数据中有某些维必须是为0的，这样就可以避免autoencoder直接将input进行embed

- Greedy Layer-wise Pre-training again

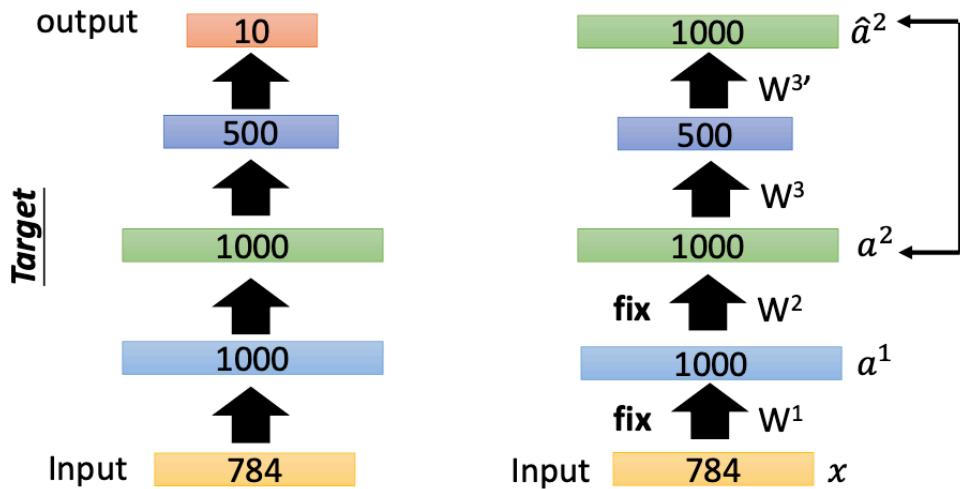


我们可以学习这样的一个autoencoder，先将input转化成一个1000维的vector，再把这个vector转化为1000维的code，再把这个code转化为1000维的vector；使input和output越接近越好，把 W^2 的值保存下来，



再继续下一个layer，使用第三个autoencoder

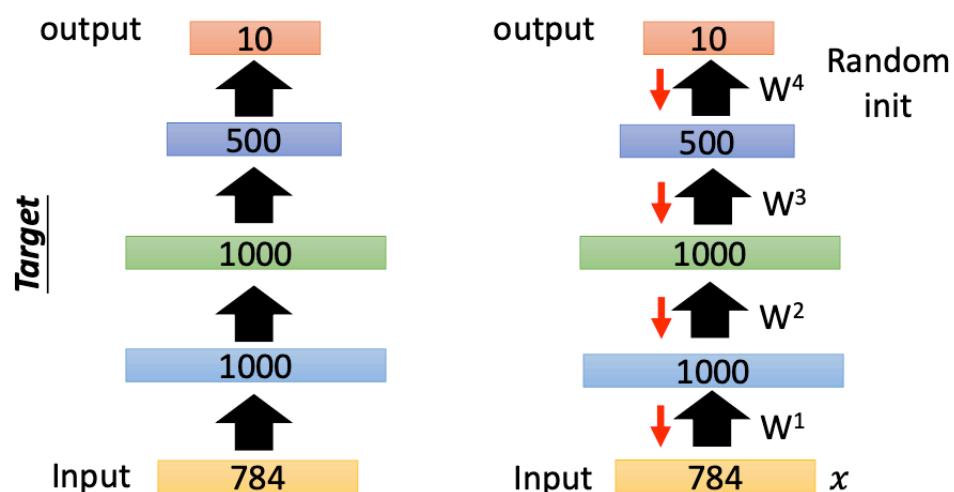
- Greedy Layer-wise Pre-training again



学习到 W^1, W^2, W^3 之后，就把这个weight作为初始值， W^4 则进行random init，再使用back propagation进行fine-tune

- Greedy Layer-wise Pre-training again

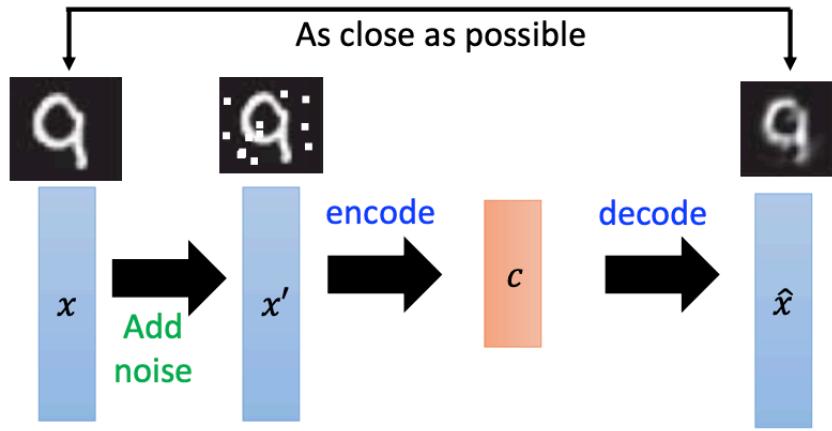
Find-tune by backpropagation



如果我们有大量的unlabeled data, labeled data只有很小一部分，那么我们就可以通过这种无监督学习的方式来初始化weight，再使用剩下的label data来对network进行fine-tune即可

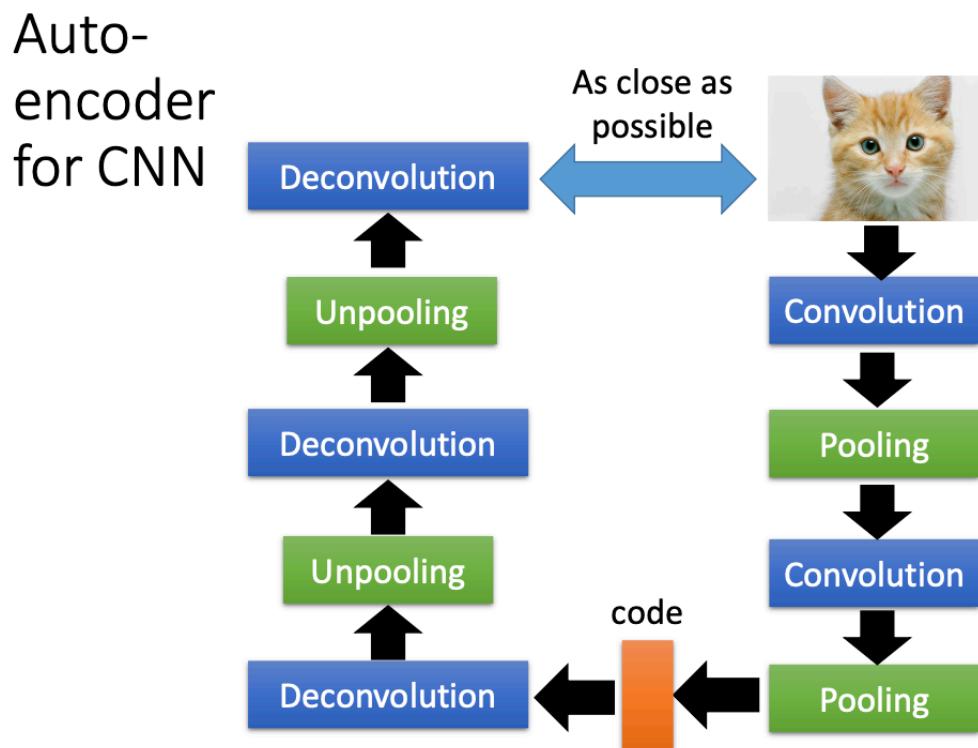
De-noising auto-encoder

对于原来的input x ，我们先加入一些noise得到 x' ，进行encode之后再进行decode，使之和最早的input之间的差值最小；encoder现在不仅学习到了encode这件事，还学习到了把noise过滤掉这件事



Vincent, Pascal, et al. "Extracting and composing robust features with denoising autoencoders." *ICML*, 2008.

Auto-encoder for CNN

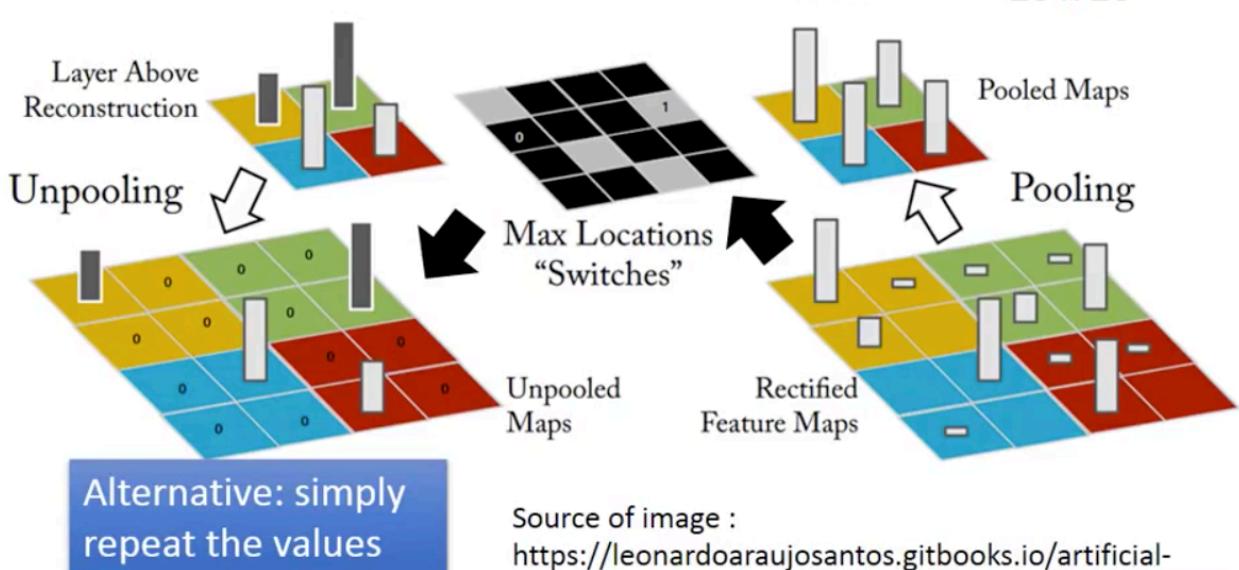
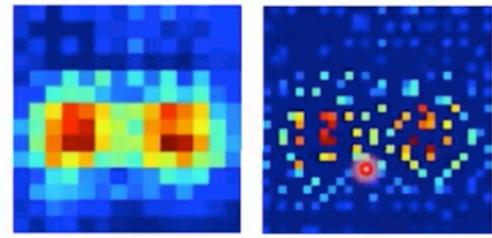


CNN -Unpooling

在pooling时，会选择四个方框中的最大值，并记住最大值的location；

在upooling时，就会用到上面的location

CNN -Unpooling



Deconvolution

Deconvolution 其实就是在做 convolution,

CNN - Deconvolution

Actually, deconvolution is convolution.

