《神经网络理论与应用》第五讲

Neural Network Theory and Applications

主讲教师：郑伟龙
助教：尹昊龙、史涵雯

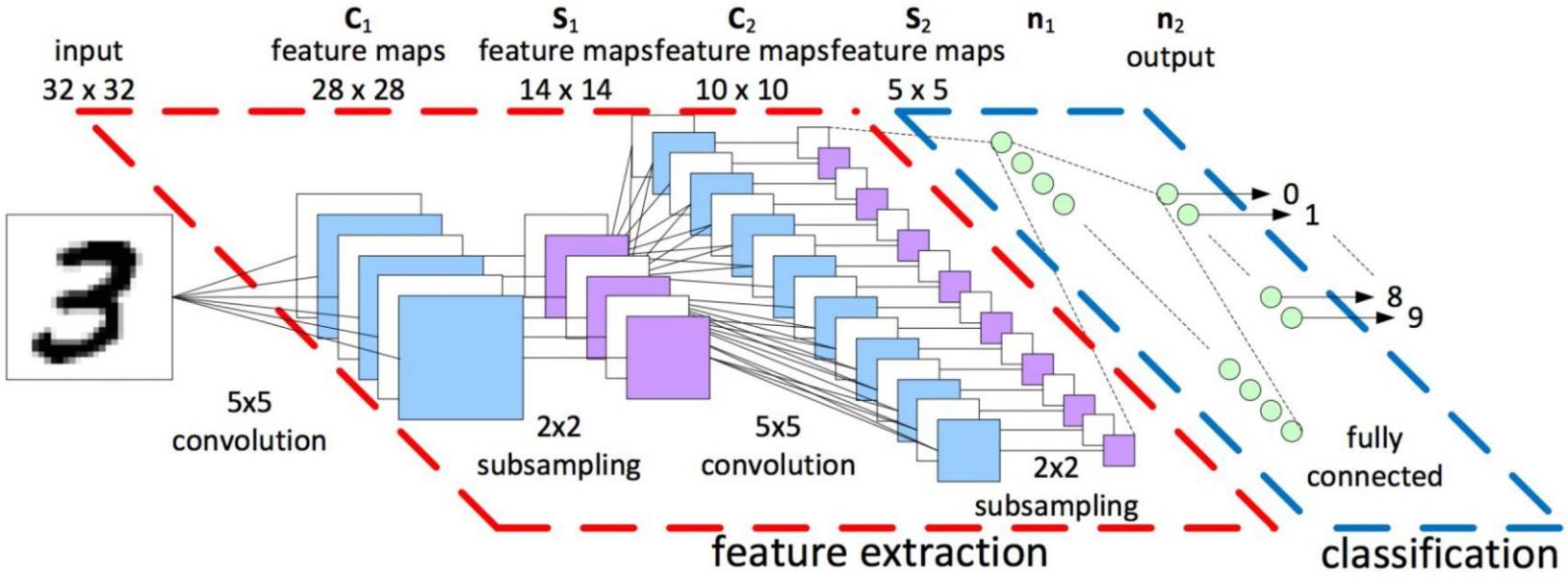
上海交通大学计算机科学与工程系

weilong@sjtu.edu.cn
http://bcmi.sjtu.edu.cn

Last Lecture Recap

- Introduction to Deep Learning
- Convolutional Neural Networks

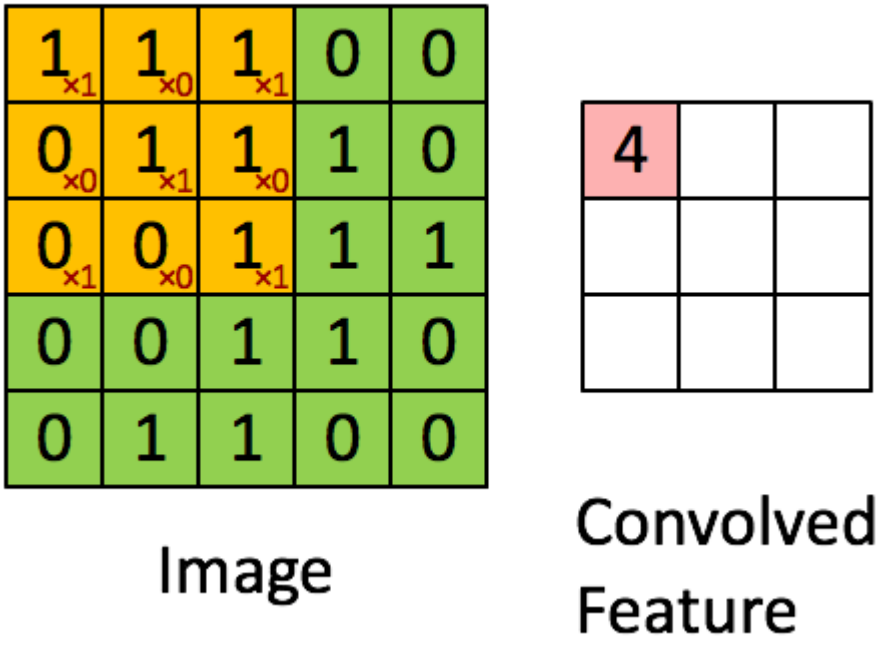
MNIST example



Recap: CNN的结构元件：卷积层

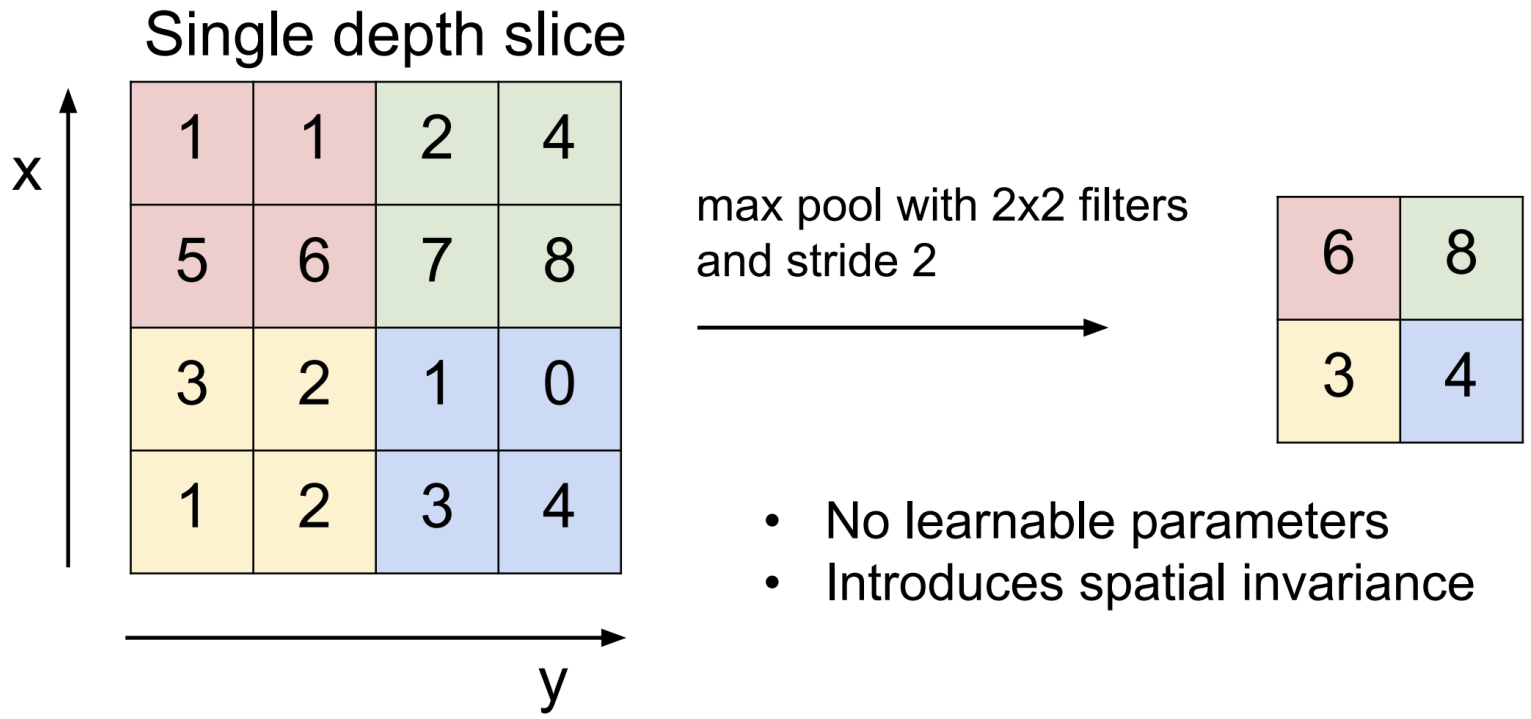
- 把滤波器想象成窗口
- 一次卷积操作是窗口内部的输入与滤波器的卷积（乘积的和）
- 窗口按照给定的步长S在输入上滑动，每次都进行一次卷积操作，即可得到输出的一个通道（特征图）

- 右图给出了H=W=5, C_1=1, C_2=1, F=3, {C}={1}, S=1, P=0 情况下的例子。



Recap: CNN的结构元件：池化层

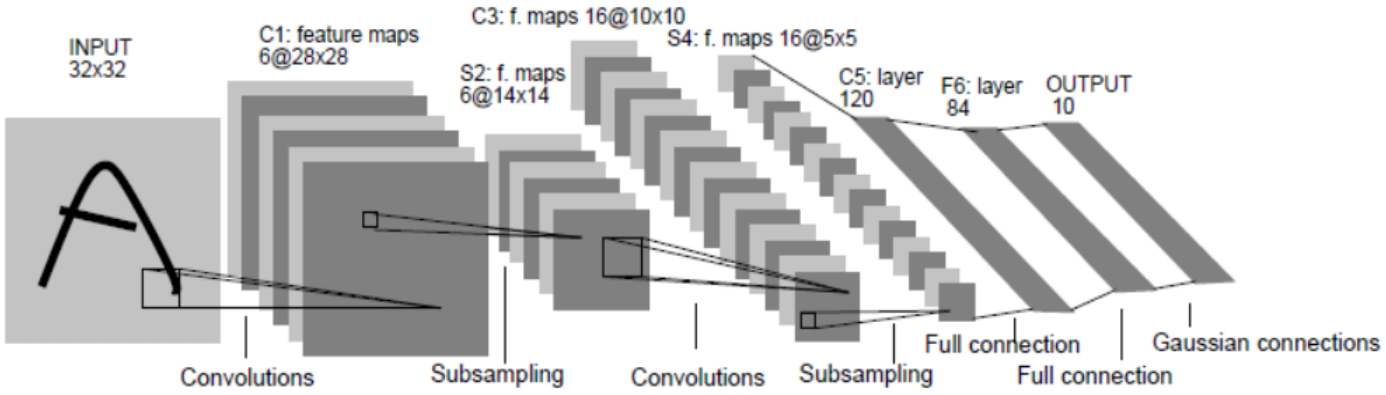
- 池化层：F*F+S，其中F是窗口长度，S是移动步长。
- 常见的CNN都是使用最大化池化层，即输出窗口中的最大值。
- 均值池化层(输出窗口内的均值)，最近常用于替代全连接层。
- 下图是F=2,S=2最大化池化的例子：



- No learnable parameters
- Introduces spatial invariance

Recap: CNN的典型结构

- 每层卷积层之后都接激活层
- 若干层卷积层之后接池化层
- 最后使用全连接层+损失函数
- **LeNet, AlexNet, VGGNet都符合这一范式**



Recap: CNN VS 全连接网络

- 局部感受野
  - 保留图像的拓扑结构
  - 提取图像的基础特征
- 权值共享
  - 减少可训练参数→防止过拟合
  - 基础特征适用于图像的所有位置→平移、扭曲不变性
- 降采样
  - 提取层次化特征
  - 减少具体位置对于最终结果的影响→平移、扭曲不变性

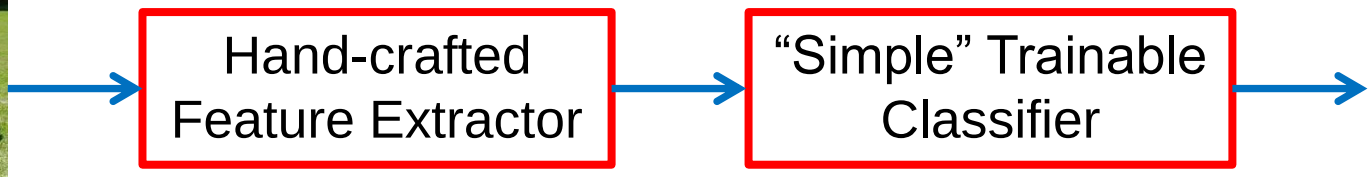
Recap: Major Breakthrough in 2006

- Biological Ability to train deep architectures by using layer-wise unsupervised learning, whereas previous purely supervised attempts had failed

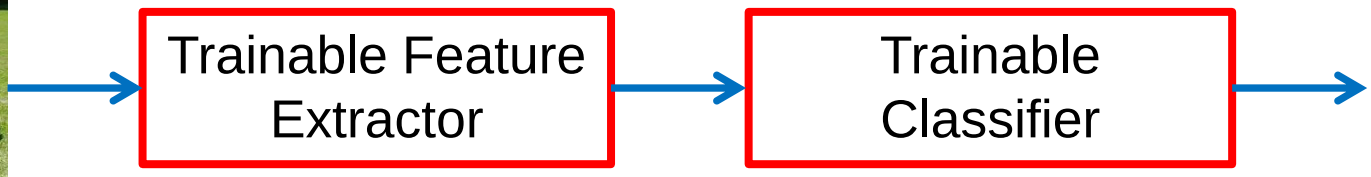
- Unsupervised feature learners:
  - RBMs
  - Auto-encoder variants
  - Sparse coding variants

Recap: Deep Learning = Learning Features

- Traditional model of pattern recognition
  - Fixed features + trainable classifier

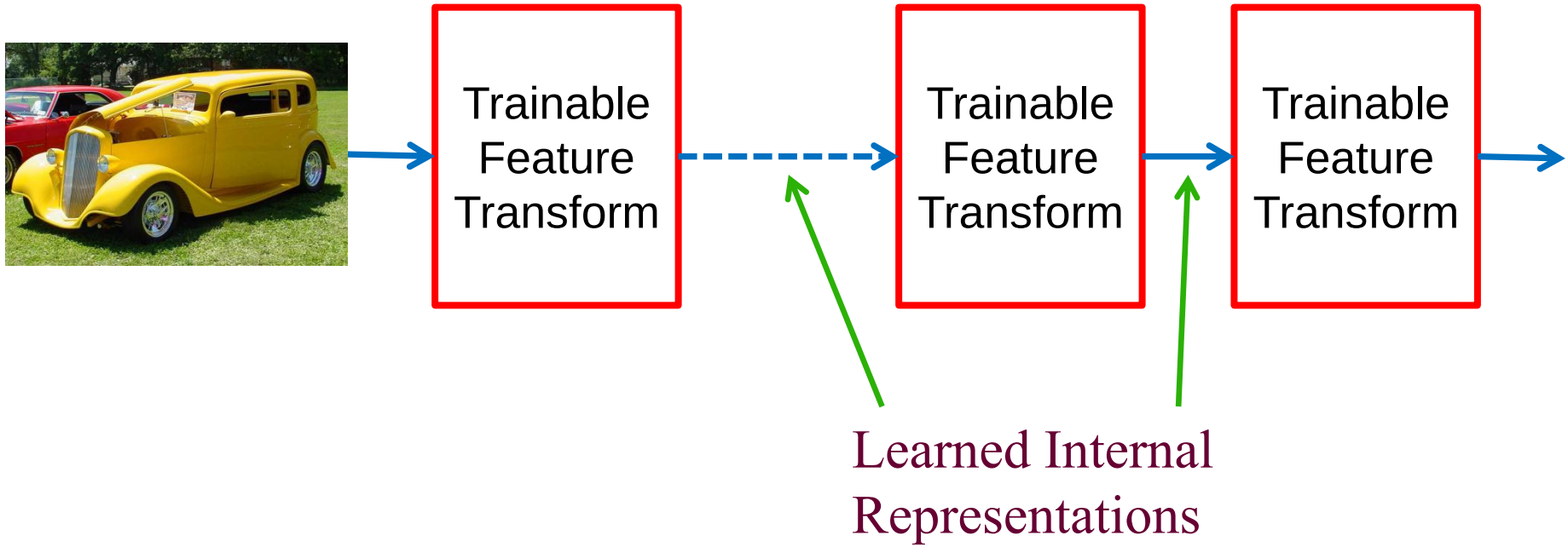


- Deep Learning
  - Trainable features + trainable classifier



Recap: Trainable Feature Hierarchies

- Each module transforms its input representation into a higher-level one.
- High-level features are more global and more invariant
- Low-level features are shared among categories



Three Types of Training Protocols

- Purely Supervised
  - Initialize parameters randomly
  - Train in supervised mode
  - Used in most practical systems for speech and image recognition
- Unsupervised, layerwise + supervised classifier on top
  - Train each layer unsupervised, one after the other
  - Train a supervised classifier on top, keeping the other layers fixed
  - Good when very few labeled samples are available
- Unsupervised, layerwise + global supervised fine-tuning
  - Train each layer unsupervised, one after the other
  - Add a classifier layer, and retrain the whole thing supervised
  - Good when label set is poor

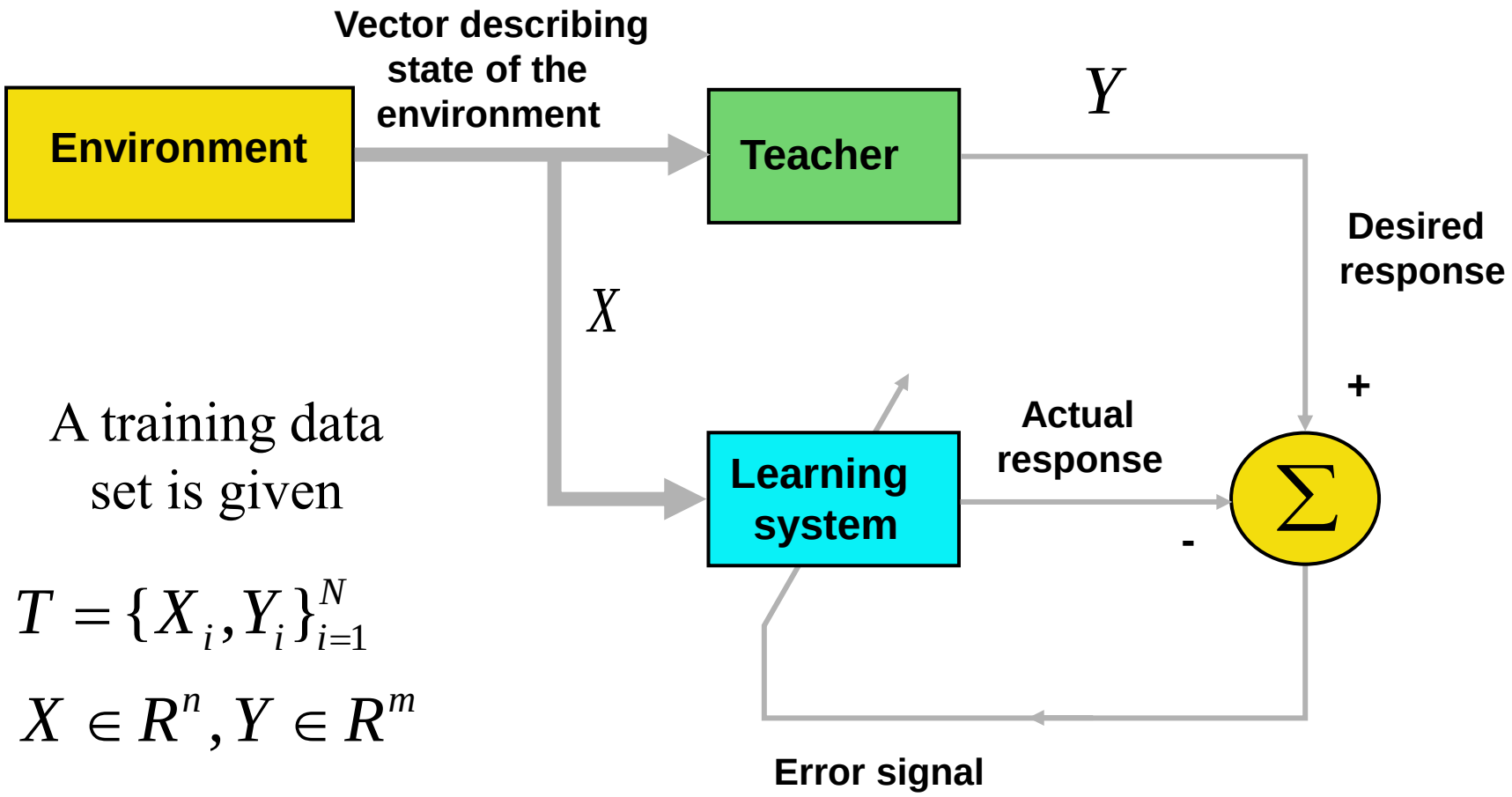
Outline of Lecture Five

- Deep Auto-encoder
- Deep Belief Networks

概要

- 什么是自编码器（Auto-Encoder）？
- 自编码器的结构、数学模型、性质
- 深度自编码器。
- 常见的自编码器/深度自编码器模型
- 自编码器/深度自编码器的应用实例

Supervised Learning



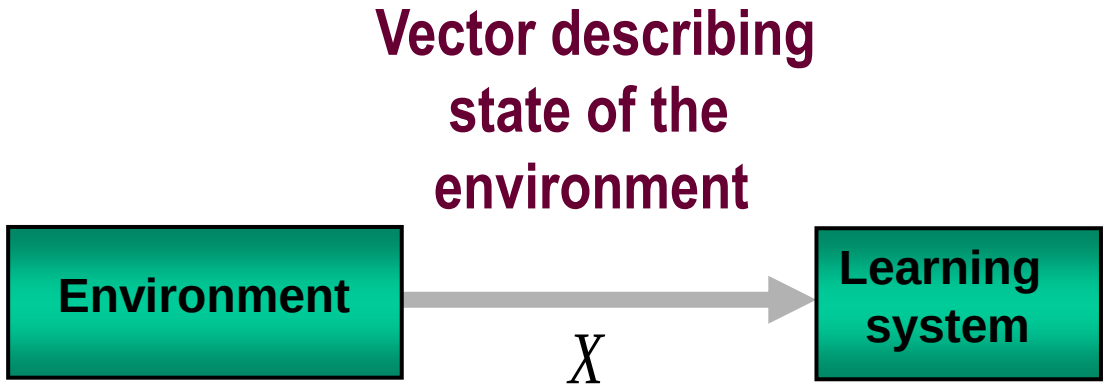
A training data set is given

$$T = \{X_i, Y_i\}_{i=1}^{N}$$

$$X \in R^n, Y \in R^m$$

Unsupervised Learning

Unsupervised Learning:
No labeled training examples are now available



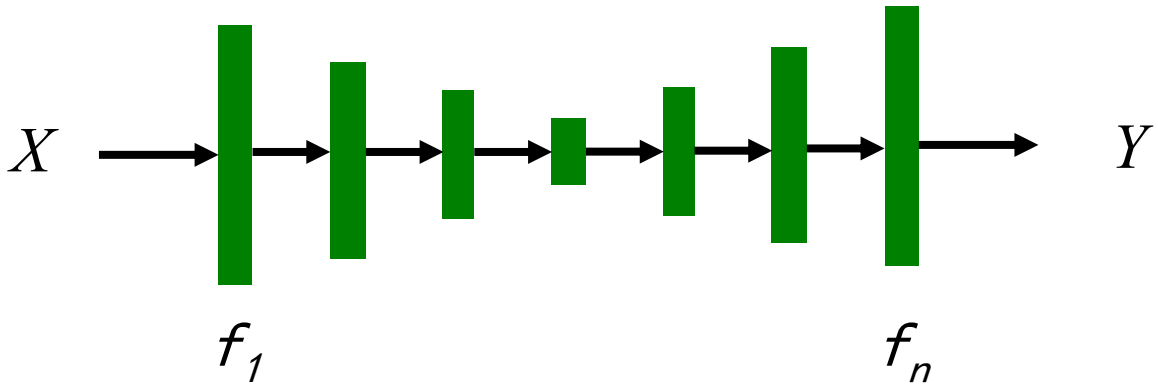
A training set without label is given

$$T = \{X_i\}_{i=1}^{N}$$

$$X \in R^n$$

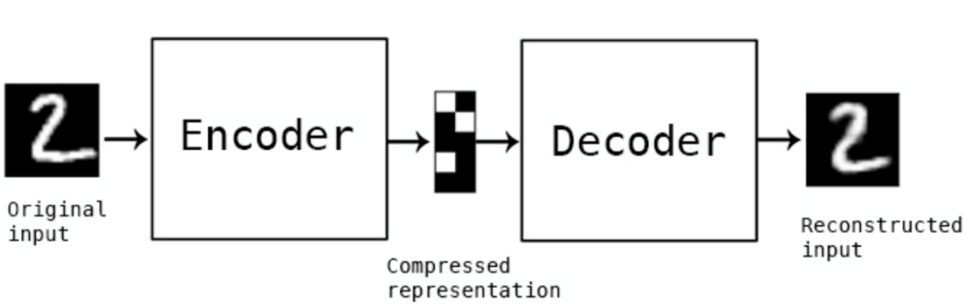
什么是Auto-Encoder？

- 从生物学大脑角度考虑，学习与重构就像编码和解码一样。
- 假设我们有一个系统F，它有n层（$f_1, f_2, \cdots, f_n$），它的输入是X，输出是Y，形象地表示为：$X \geqslant f_1 \geqslant f_2 \geqslant \cdots \geqslant f_n \geqslant Y$
- 如果输出Y 等于输入X，即输入X 经过了该系统变化之后没有任何的信息损失。这也意味着输入X 经过每一层f_i 都没有任何信息损失，即每一层都是原有信息的另外一种表示，我们就获得了输入X 的一系列的特征表示。



什么是Auto-Encoder？

- 自编码器是一种神经网络，不同于其他的网络，自编码器网络的输出用来重建网络的输入。

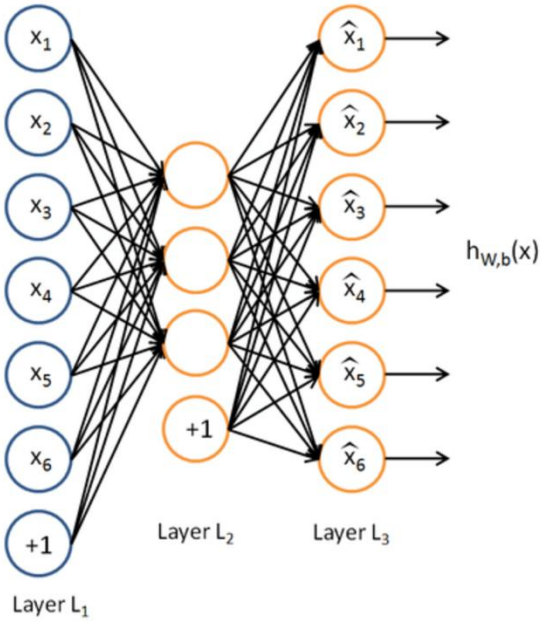


- 自编码器网络，包括编码阶段和解码阶段。编码阶段可以按照非监督学习的方式，找到输入数据的某种表示方式；解码阶段将找到的这种表示方式恢复成原始的输入。

- 自编码器可以用来降维、提取特征、预训练深度神经网络。

自编码器的结构、数学模型、性质

- 最简单的自编码器结构是一个只有一个隐层的神经网络。
- 这个网络的输入为X
- 编码阶段用方程 f 表示
- 解码阶段用方程 g 表示
- 网络的输出为 $\hat{X}$
- 数学表示为：$\hat{X} = g(f(X))$



网络训练的目的，就是让重构的变量和输入的变量差距尽量小。即最小化损失函数：$L(\hat{X}, X)$

常见的损失函数是最小平方误差函数：$L(\hat{X}, X) = \|\hat{X} - X\|_2^2$

自编码器的结构、数学模型、性质

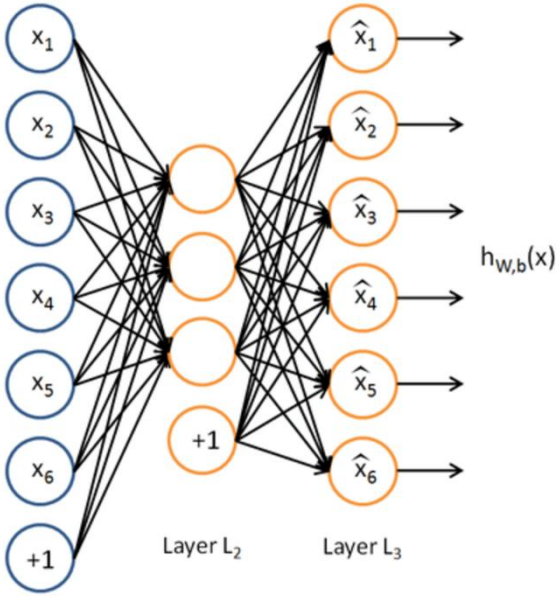
- 以只有一个隐层的自编码器为例，随机初始化权重W和偏执b，使用编码阶段和解码阶段的激活函数均为sigmoid函数，误差函数为平方误差函数。
- 编码阶段：$\sigma(W_{encode}X + b_{encode})$
- 解码阶段：

$$\hat{X} = \sigma(\mathrm{W}_{decode}\sigma(W_{encode}X + b_{encode}) + b_{decode})$$

- 损失函数为

$$L(\hat{X}, X) = \left\|\hat{X} - X\right\|_2^2$$

- 利用BP算法可以对网络进行优化求解



自编码器的结构、数学模型、性质

- 自编码器可以用来进行降维，当编码函数和解码函数为线性函数时，可以证明自编码器和PCA降维一致。当编码函数和解码函数为非线性函数式，自编码器可以当做非线性降维算法。
- 自编码器隐层可以作为输入数据的特征，供其他机器学习算法使用。
  - 当输入的数据是原始信号，自编码器隐层可以作为新的特征来使用 —— 即可以自动提取所需要的特征。
  - 当输入的数据是人工提取的特征时，可以将自编码器看做是一种特征变换的工具，将原始的特征映射到一个新的特征空间中。

Training Deep Networks

- Difficulties of supervised training of deep networks
  - Early layers of MLP do not get trained well
    - Diffusion of Gradient – error attenuates as it propagates to earlier layers
    - Leads to very slow training
    - Need a way for early layers to do effective work
  - Often not enough labeled data available while there may be lots of unlabeled data
    - Can we use unsupervised/semi-supervised approaches to take advantage of the unlabeled data
  - Deep networks tend to have more local minima problems than shallow networks during supervised training

Greedy Layer-Wise Training

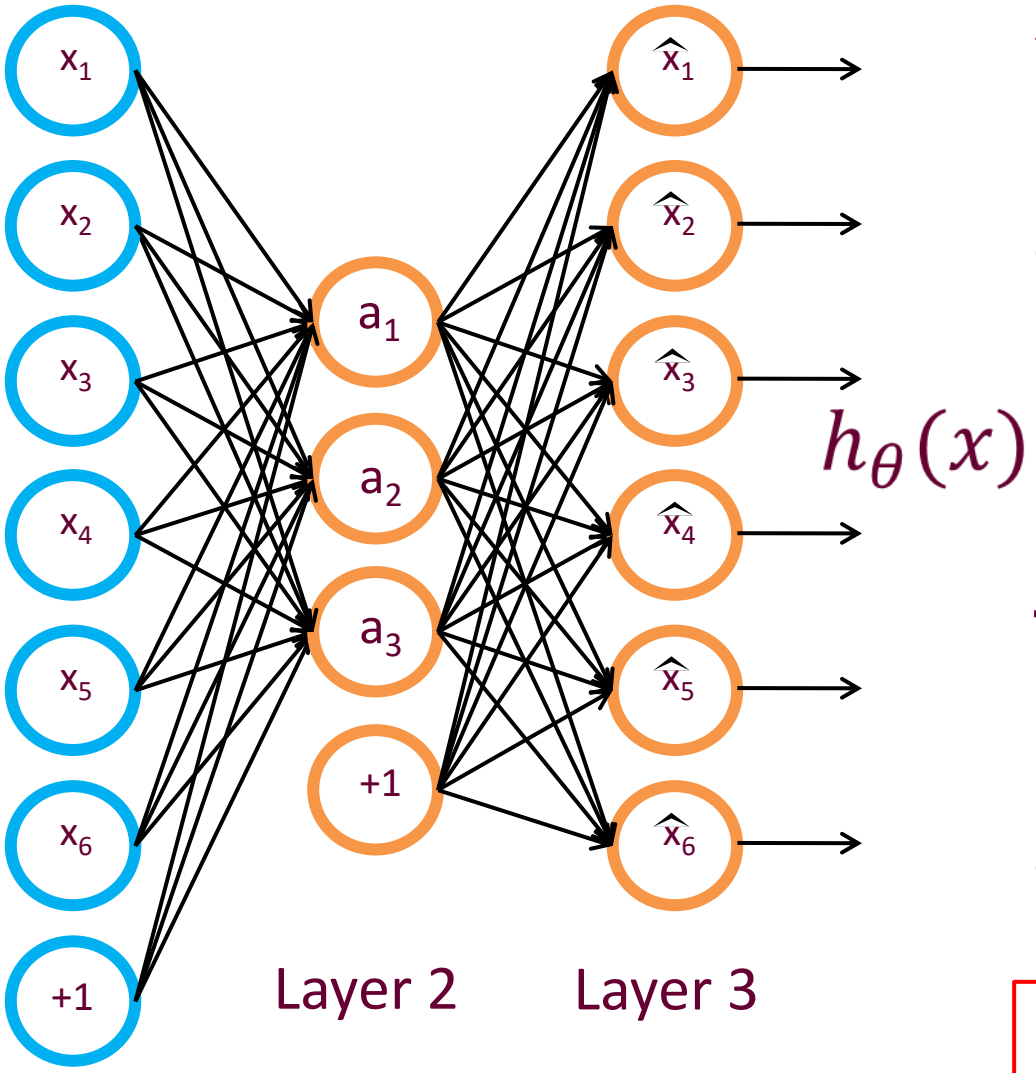
- One answer is greedy layer-wise training

1. Train first layer using unlabeled data
   - Could do supervised or semi-supervised but usually just use the larger amount of unlabeled data. Can also use labeled data but just leave out the label.
2. Then freeze the first layer parameters and start training the second layer using the output of the first layer as the unsupervised input to the second layer
3. Repeat this for as many layers as desired
   - This builds our set of robust features
4. Use the outputs of the final layer as inputs to a supervised layer/model and train the last supervised layer(s) (leave early weights frozen)
5. Unfreeze all weights and fine tune the full network by training with a supervised approach, given the pre-processed weight settings

Greedy Layer-Wise Training

- Greedy layer-wise training avoids many of the problems of trying to train a deep net in a supervised fashion
  - Each layer gets full learning focus in its turn since it is the only current "top" layer
  - Can take advantage of the unlabeled data
  - Once you finally start the supervised training portion the network weights have been adjusted so that you are already in a good error basin and now just need to fine tune. This helps with problems of
    - Ineffective early layer learning
    - Deep network local minima
- We will discuss the two most common approaches
  - Stacked Auto-Encoders
  - Deep Belief Networks

Auto-Encoders



Autoencoder.

Network is trained to output the input (learn identify function).

$$h_\theta(x) \approx x$$

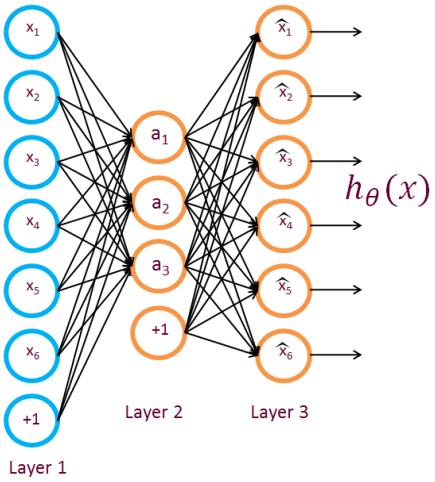
Trivial solution unless:
- Constrain number of units in Layer 2 (learn compressed representation), or
- Constrain Layer 2 to be sparse.

Auto-Encoders

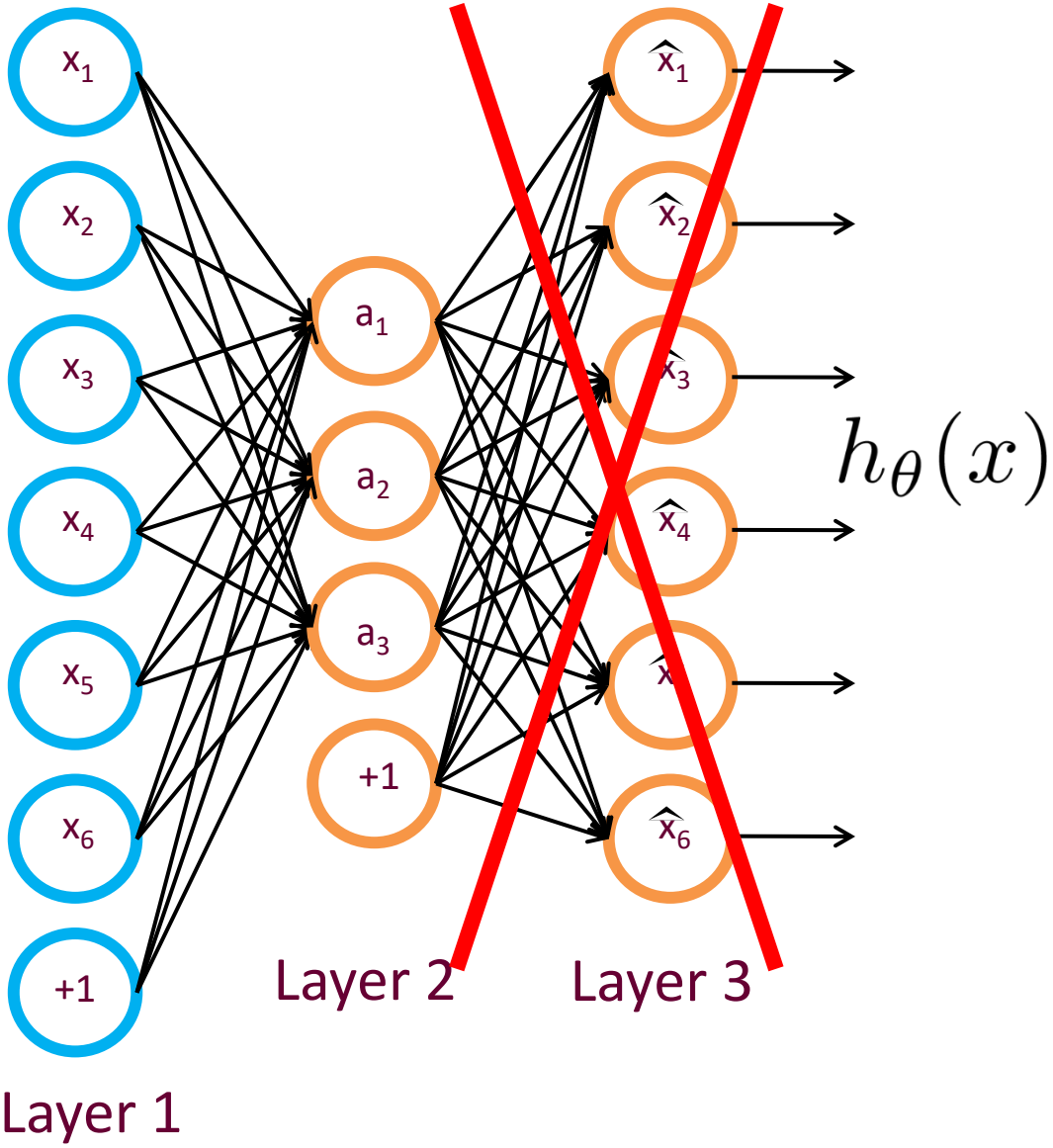
Training a sparse autoencoder.

Given unlabeled training set x_1, x_2, …

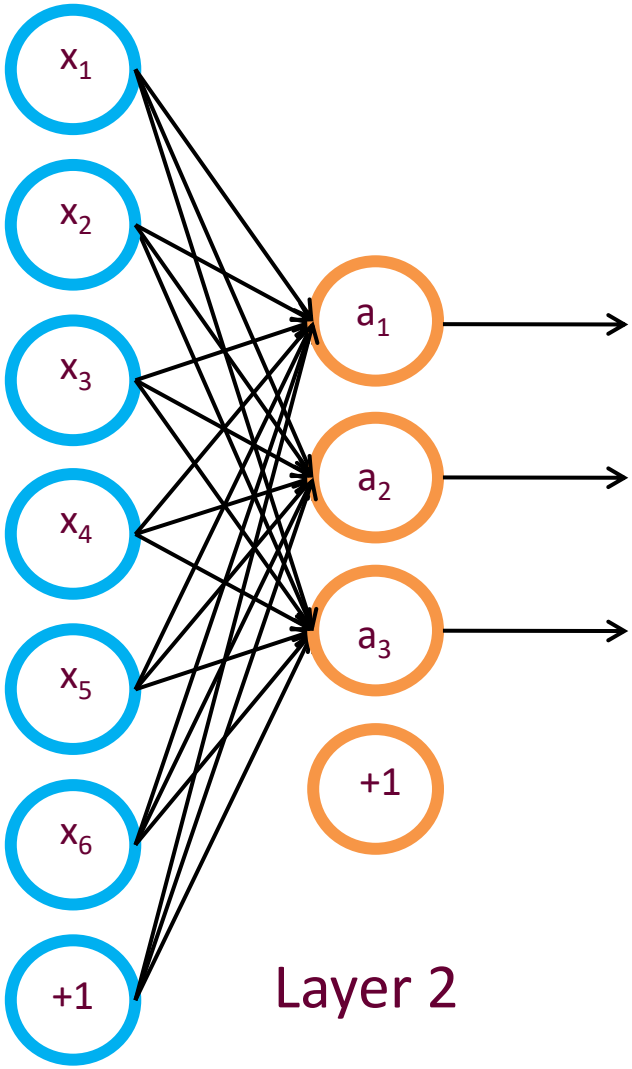
$$\min_{\theta} \underbrace{\left\| h_\theta(x) - x \right\|^2}_{\text{Reconstruction error term}} + \lambda \underbrace{\sum_i |a_i|}_{L_1 \text{ sparsity term}}$$



Auto-Encoders



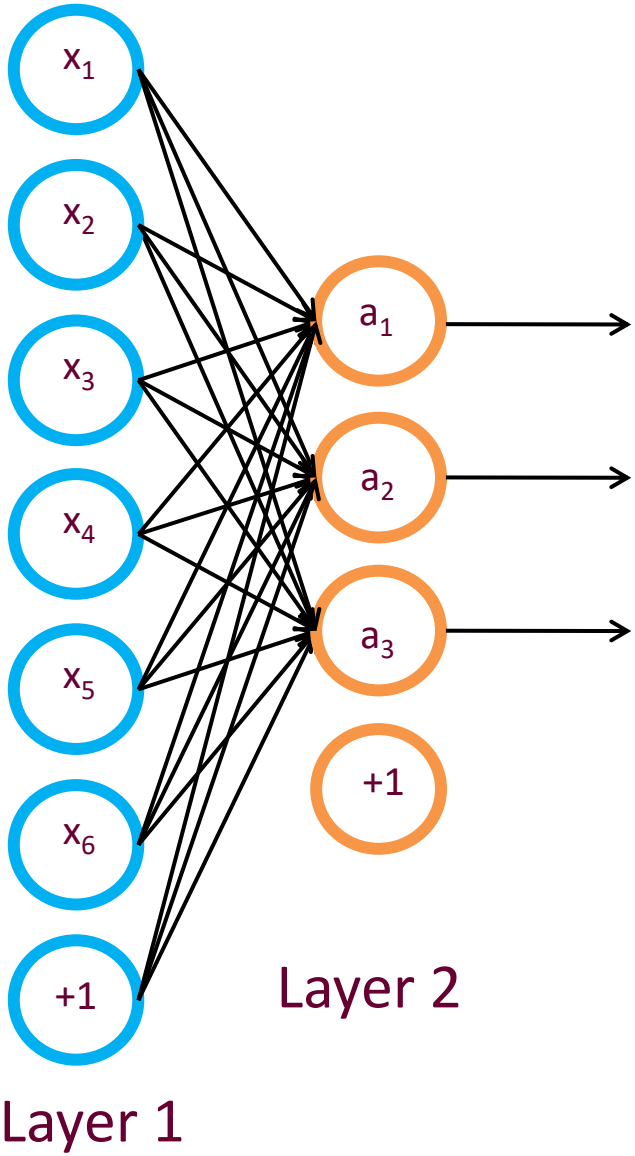
Auto-Encoders



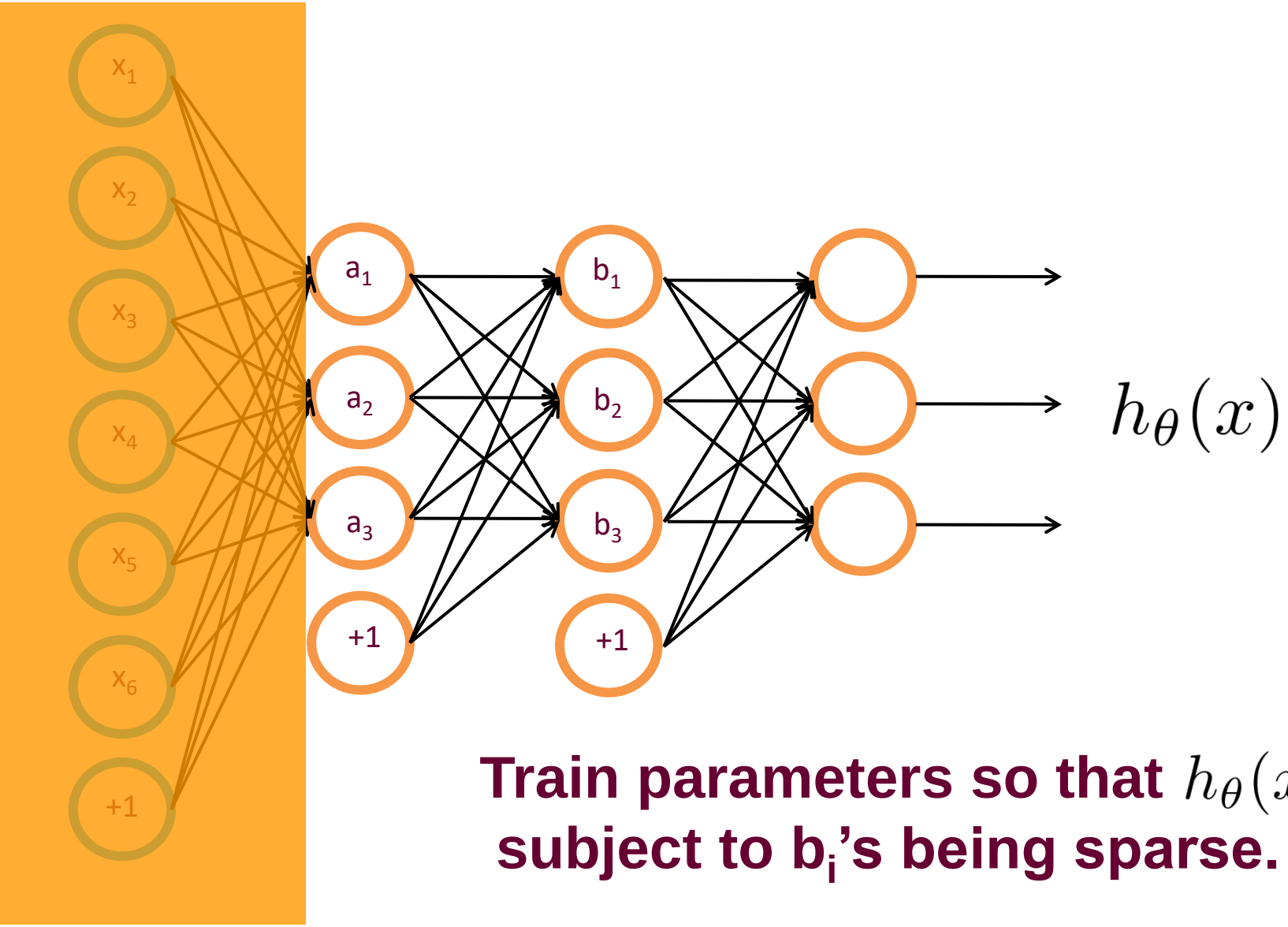
$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

New representation for input.

Auto-Encoders

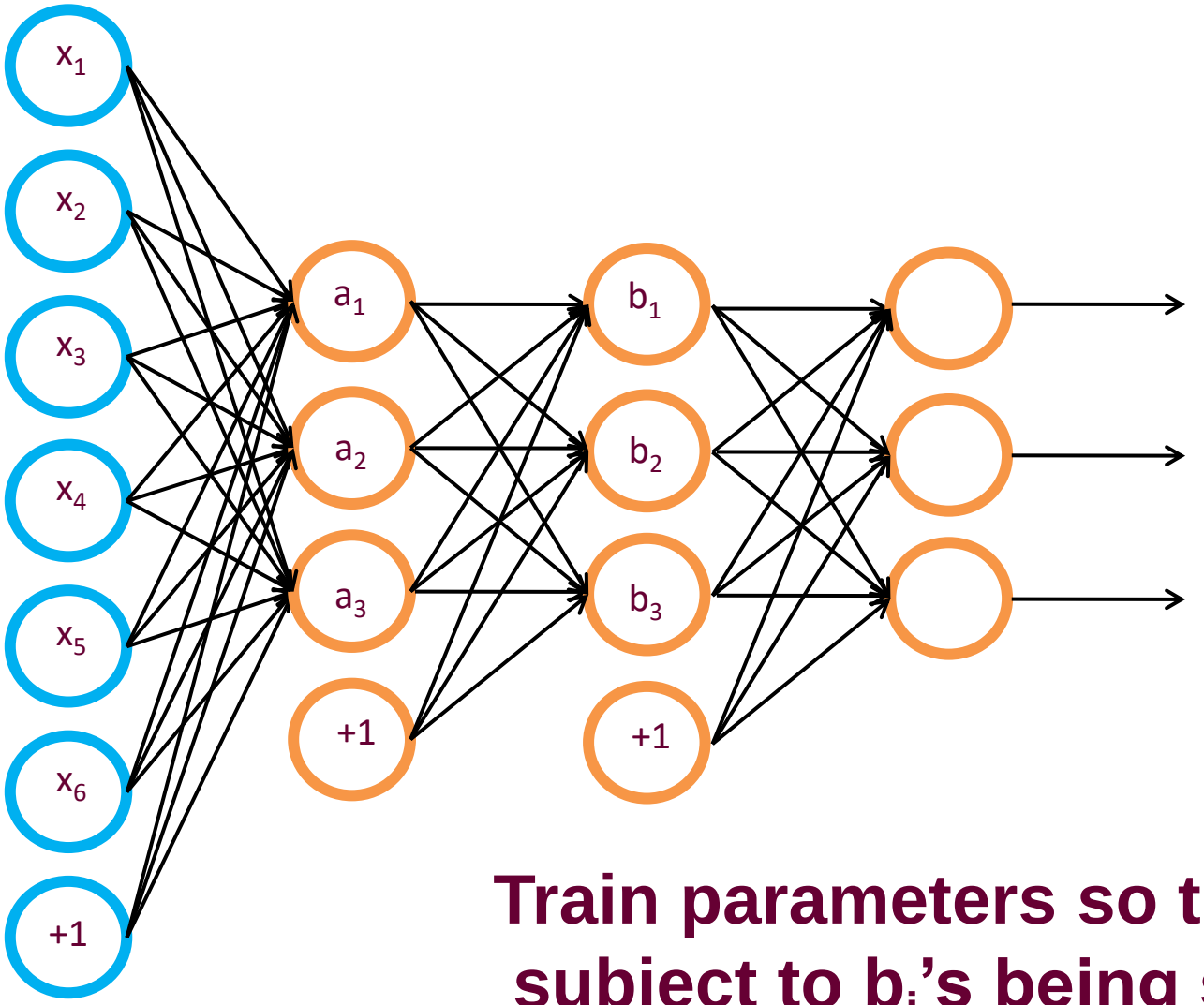


Auto-Encoders



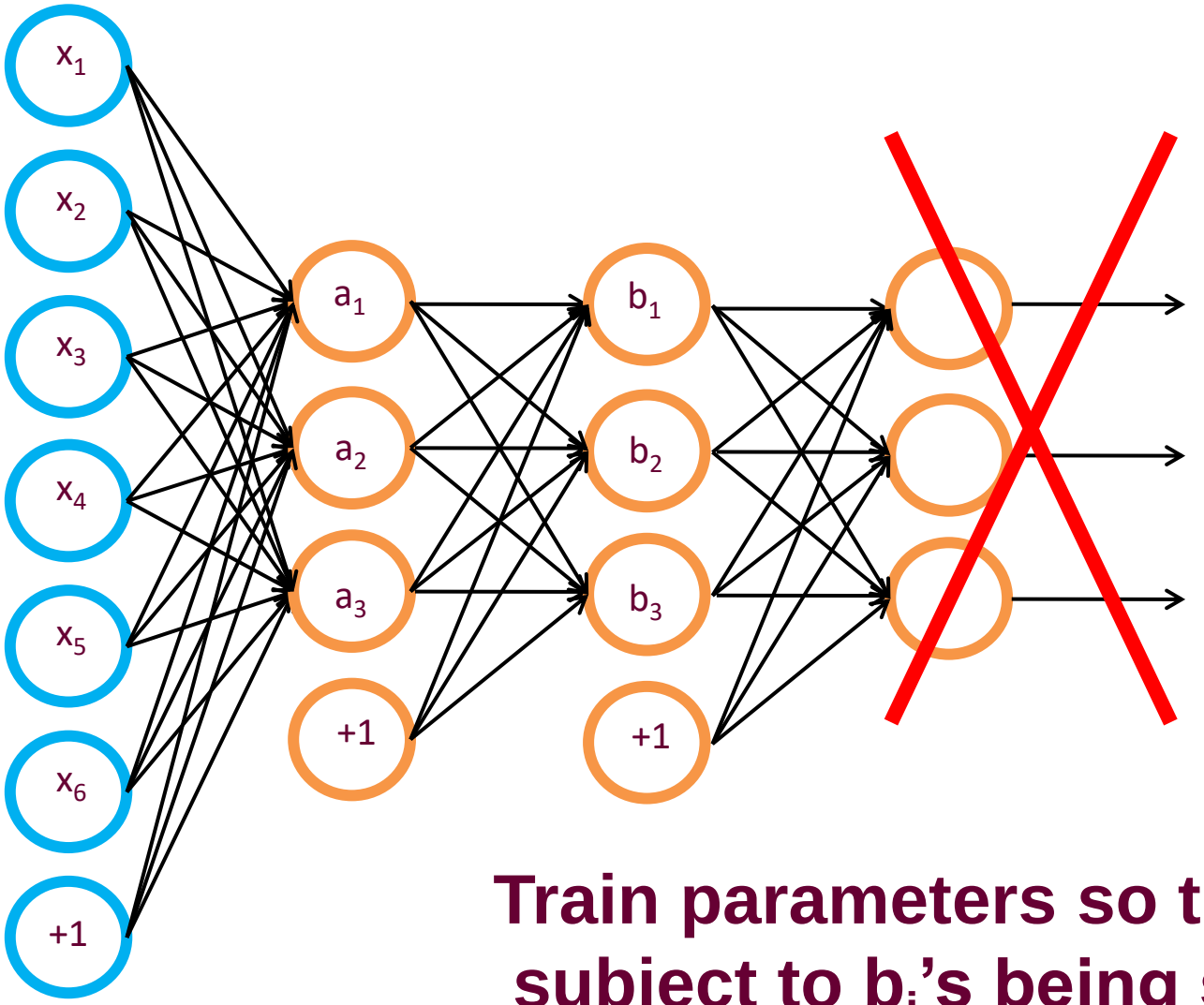
Train parameters so that $h_\theta(x) \approx a$, subject to b_i's being sparse.

Auto-Encoders



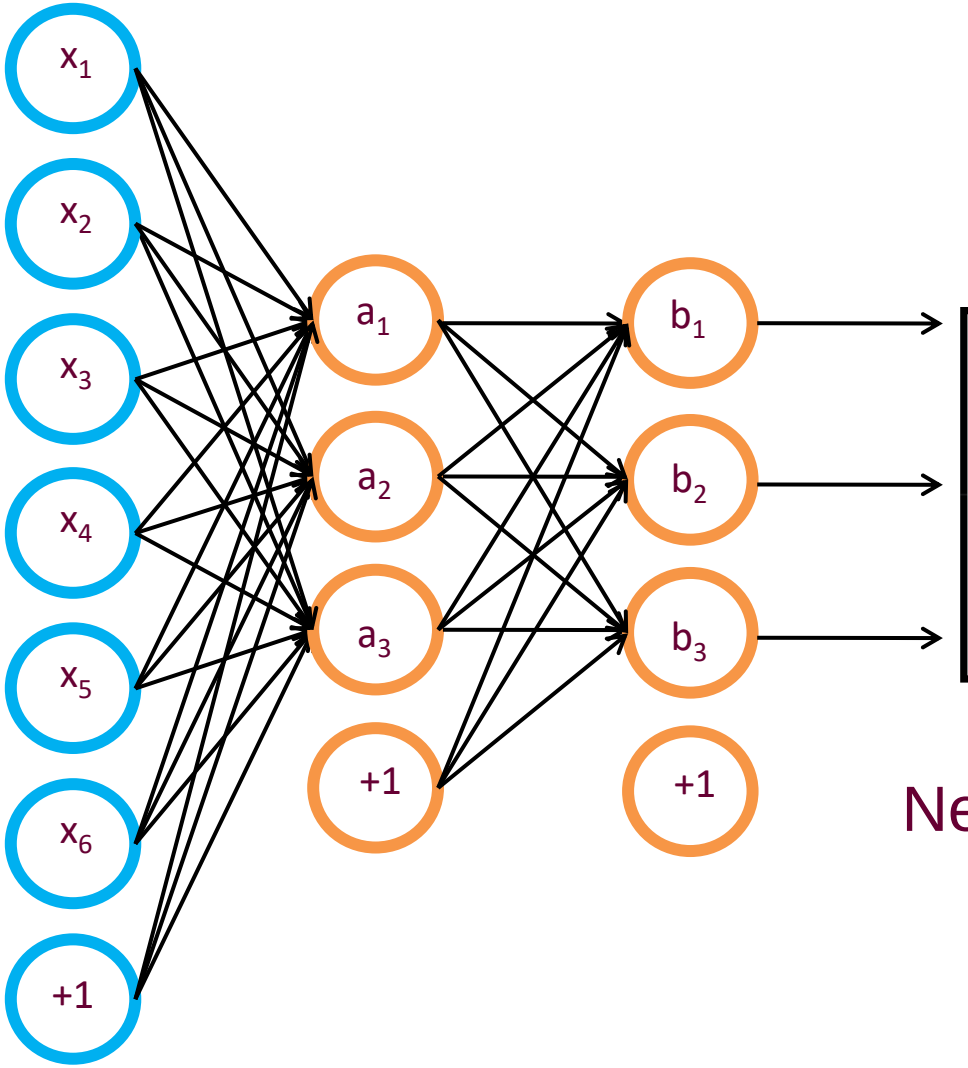
Train parameters so that $h_\theta(x) \approx a$, subject to b_i's being sparse.

Auto-Encoders



Train parameters so that $h_\theta(x) \approx a$, subject to b_i's being sparse.

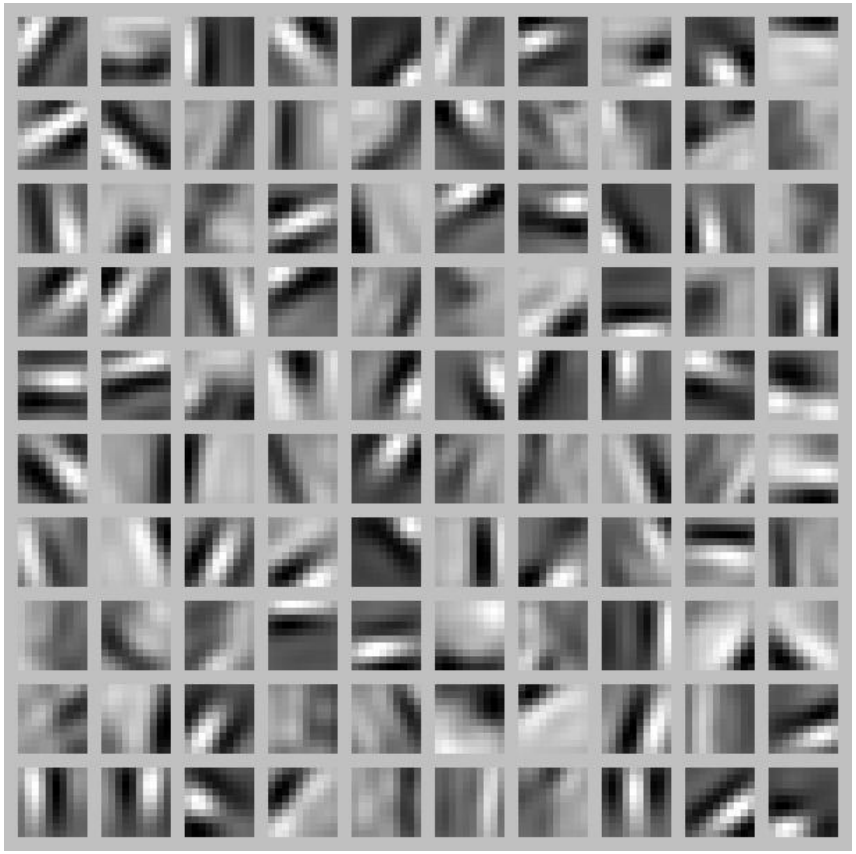
Auto-Encoders



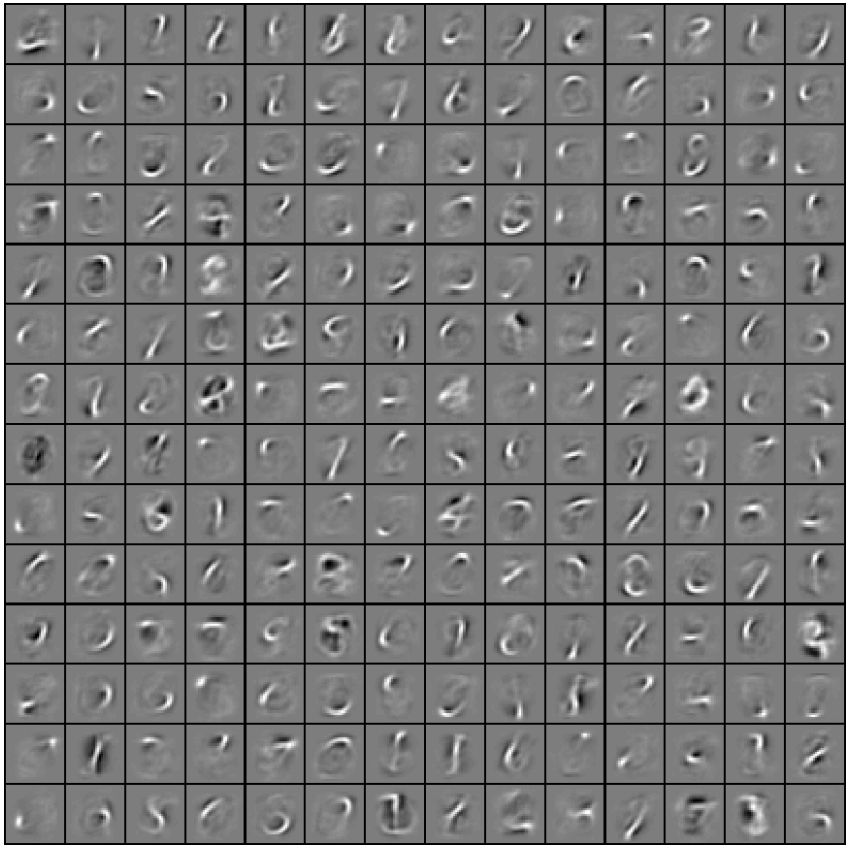
$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

New representation for input.

Sparse Representation



Natural Image



MNIST Dataset

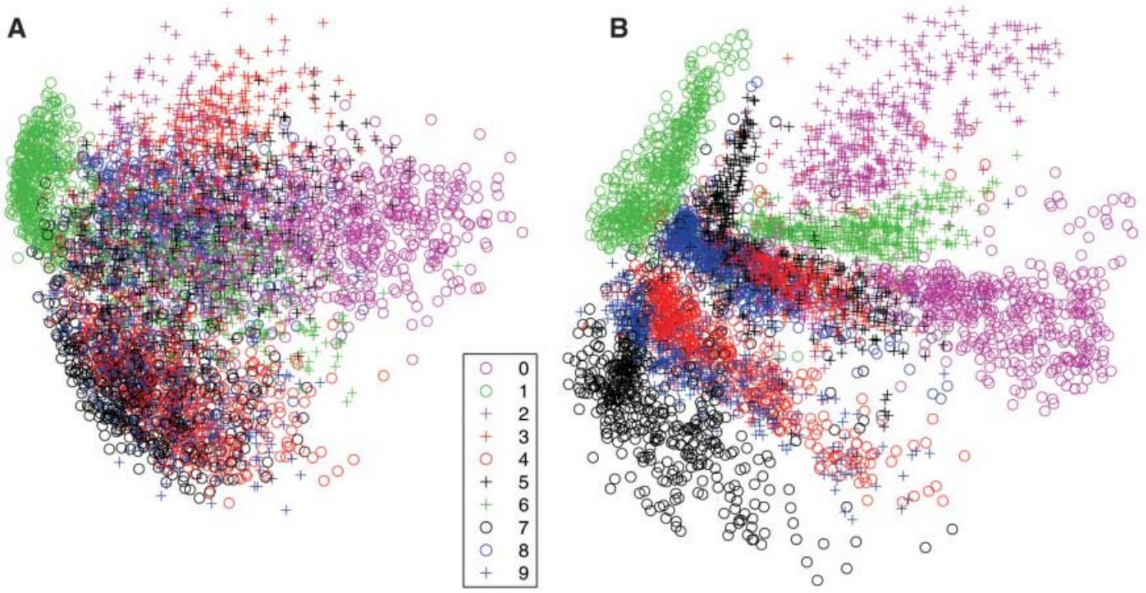
深度自编码器

- 单隐层自编码器网络结构简单，易于实现，但是这种结构不能学到复杂的特征。
- 深度自编码器利用多个隐层，对输入的数据进行多层次的特征学习，能够学习到比较复杂的特征。
- 实现深度自编码器的方式有多种：
  - Stacked Auto-Encoder (SAE)，
  - Deep Auto-Encoder (DAE) – RBM based
  - 卷积自编码器（Convolutianl Auto-Encoder，CAE）
  - 与LSTM网络结合的自编码器结构，常用于序列学习的编码和解码。

深度自编码器降维

- 数据降维
  - A图是PCA对MNIST降维结果
  - B图是Deep Auto-Encoeder的降维结果
  - B图降维效果好于A图

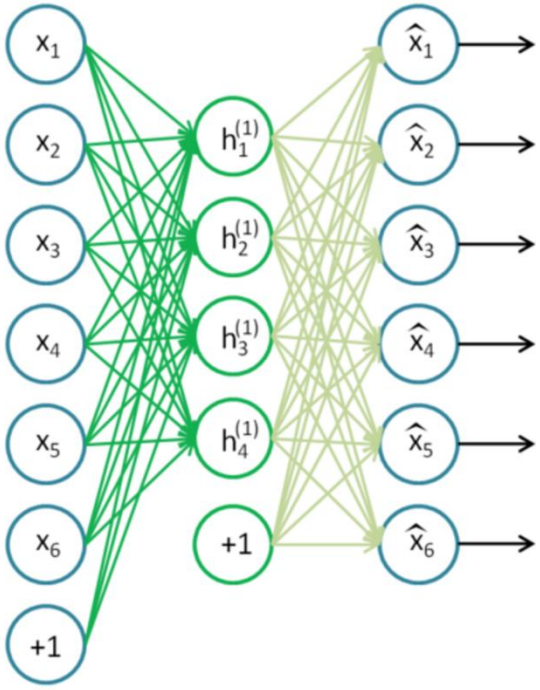
Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

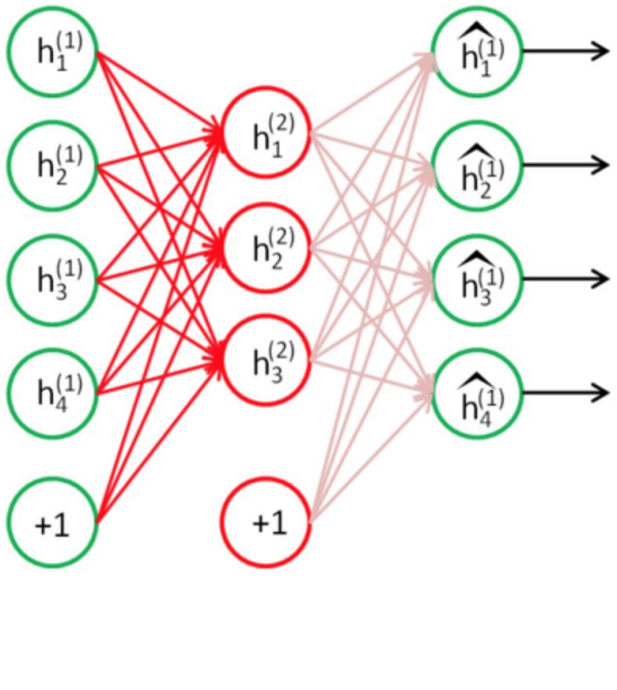


Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.

深度自编码器 - SAE

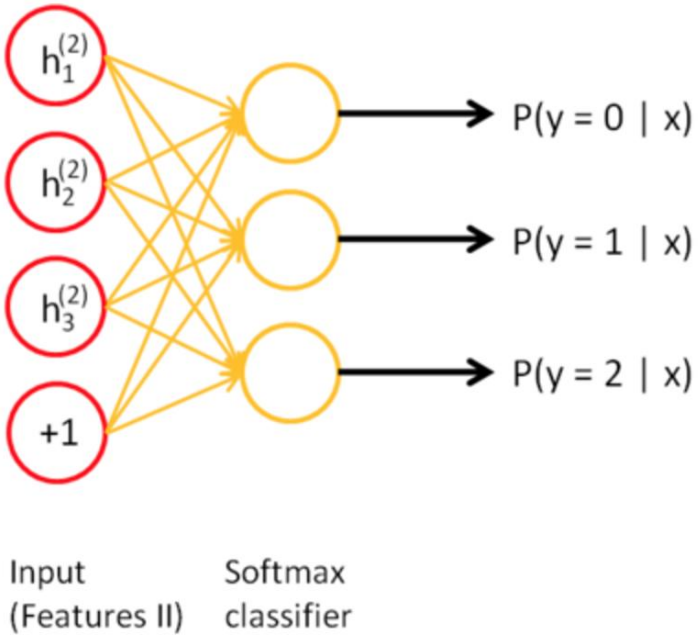
- Stacked Auto-Encoder采用逐层贪婪训练算法进行训练。
- 首先对输入层建立单隐层自编码器，隐层作为Feature 1
- 然后以第一个自编码器得到的Feature 1作为输入，建立新的单隐层自编码器，隐层作为Feature 2.





深度自编码器 - SAE

- 然后以第一个自编码器得到的Feature 1作为输入，建立新的单隐层自编码器，隐层作为Feature 2.
- 对于Feature 2的处理按照任务不同有多种，常见的是接一个softmax层作为分类器来使用；也可以展开该结构，作为多隐层的自编码器。

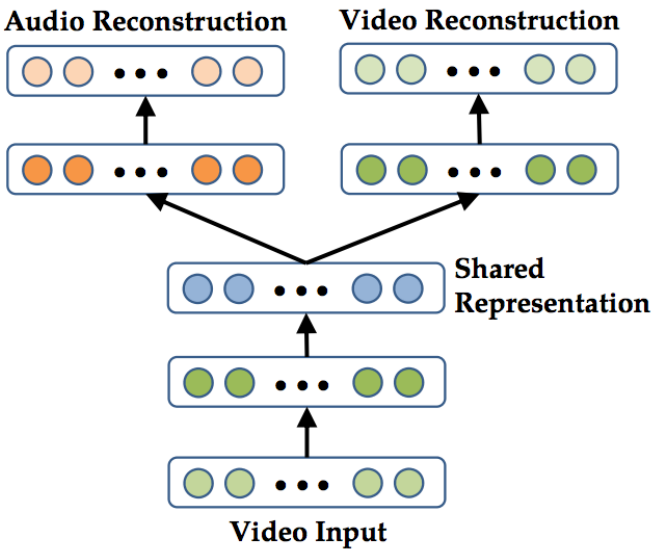


深度自编码器 -- SAE

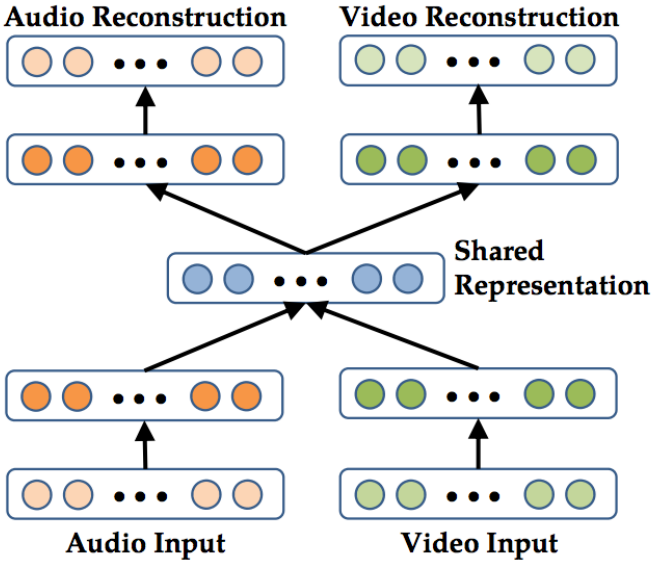
- 无论是展开形成多层的自编码器，还是接softmax用来分类，预训练后的网络参数往往表现不太理想，这时，需要进行参数微调 —— Fine Tuning.
- 参数微调，是指用BP算法对整个网络的参数作调整。
  - 预训练阶段，网络参数是分层训练的，直接整合起来效果不一定理想。
  - 参数微调阶段，是将网络里初始化后的参数作为整体进行调整，使最终参数适应所处理的问题。

多模态深度自编码器

- 多模态深度自编码器
  - 融合不同模态的数据，得到高级表示。
  - 可以看做是一种特征变换。



(a) Video-Only Deep Autoencoder

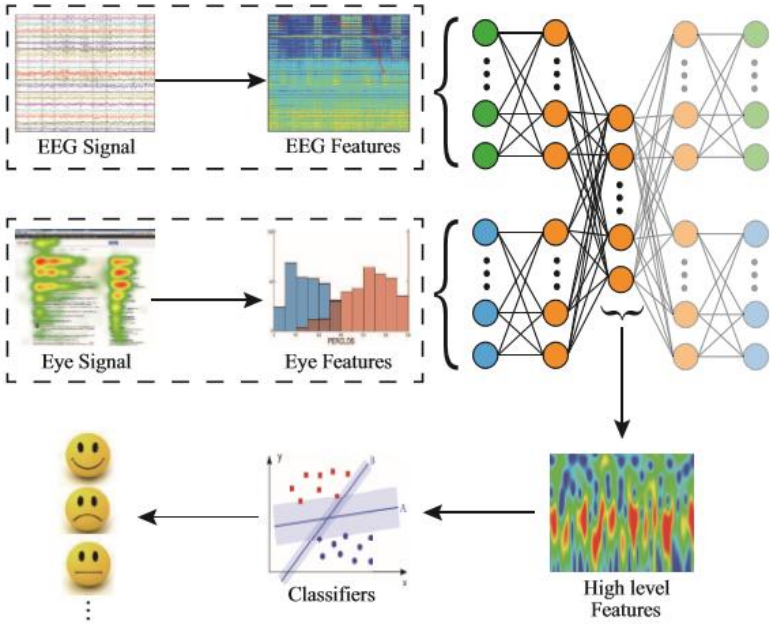


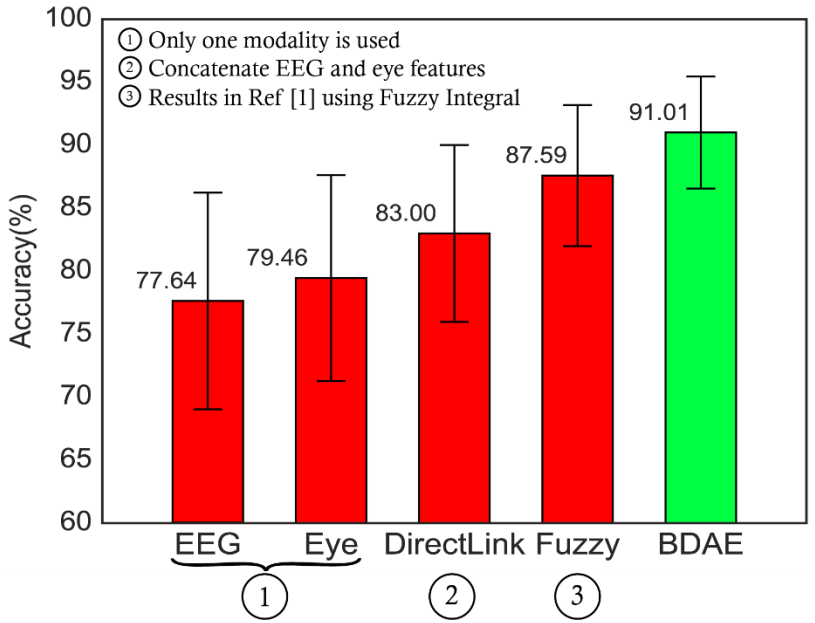
(b) Bimodal Deep Autoencoder

Ngiam, Jiquan, et al. "Multimodal deep learning." *Proceedings of the 28th international conference on machine learning (ICML-11)*. 2011.

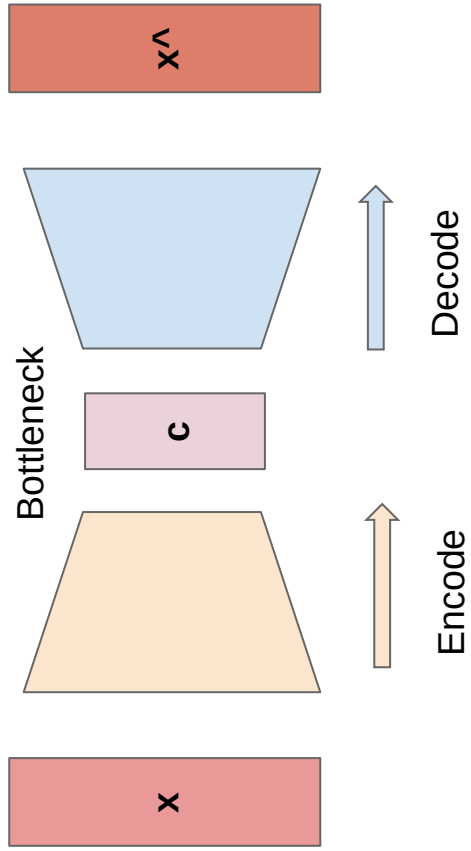
多模态深度自编码器的应用

- 使用多模态深度自编码器融合脑电和眼动特征
  - 无监督地融合脑电和眼动特征，得到融合后的高层次特征
  - 对融合后的高层次特征进行建模，进行情绪识别。
  - 相较于SVM，在SEED3类情绪数据集上取得了更好的结果。





深度自编码器Pytorch实现

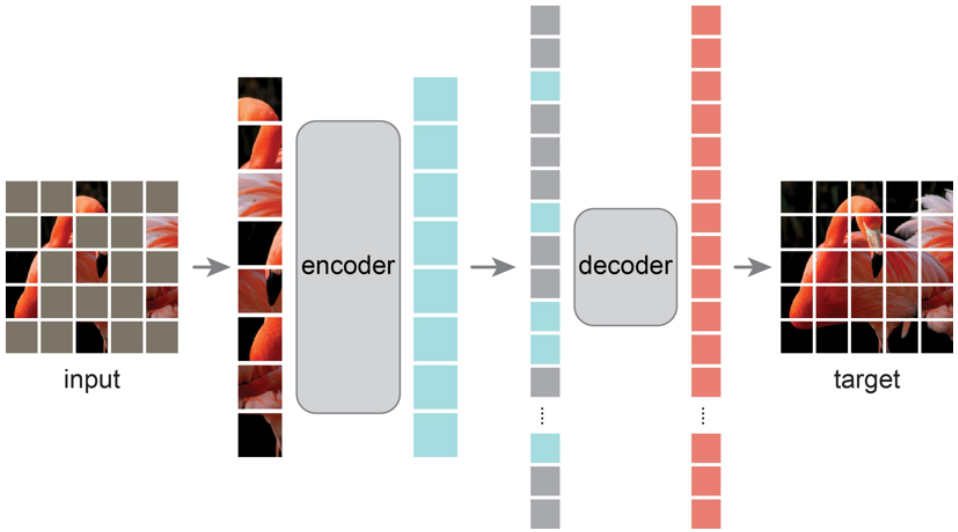



```python
class AE(nn.Module):
    def __init__(self, num_inputs=784):
        super().__init__()
        self.encoder = nn.Sequential(
            nn.Linear(num_inputs, 400),
            nn.ReLU(inplace=True),
            nn.Linear(400, 400),
            nn.ReLU(inplace=True),
            nn.Linear(400, 20)
        )
        self.decoder = nn.Sequential(
            nn.Linear(20, 400),
            nn.ReLU(inplace=True),
            nn.Linear(400, 400),
            nn.ReLU(inplace=True),
            nn.Linear(400, num_inputs)
        )

    def forward(self, x):
        return self.decoder(self.encoder(x))


ae = AE(784)
x = torch.randn(10, 784)
print('Input tensor x size: ', x.size())
y = ae(x)
print('Output tensor y size: ', y.size())
```

```
Input tensor x size:  torch.Size([10, 784])
Output tensor y size:  torch.Size([10, 784])
```

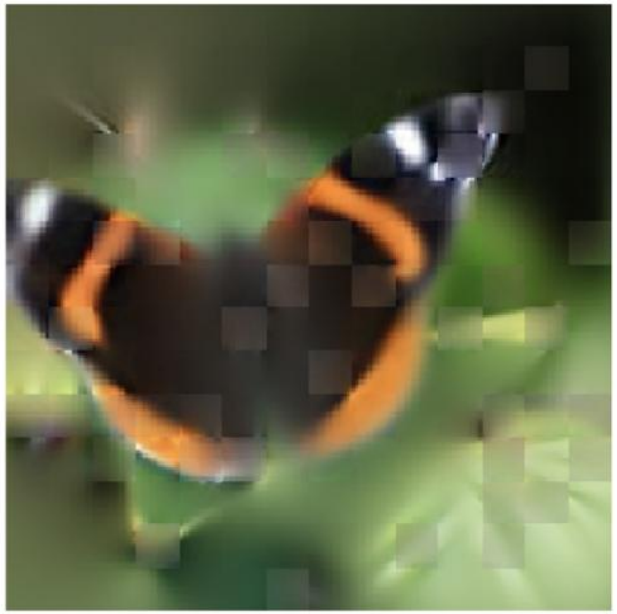
MAE

- **Masked Auto-Encoders as Scalable Vision Learners (MAE) (Kaiming et al. 2022).**
- **What is MAE?**
  - **Very simple method, but highly effective.**
  - **BERT-like algorithm, but with crucial design changes for vision.**
  - **Intriguing properties – better scalability and more from analysis.**

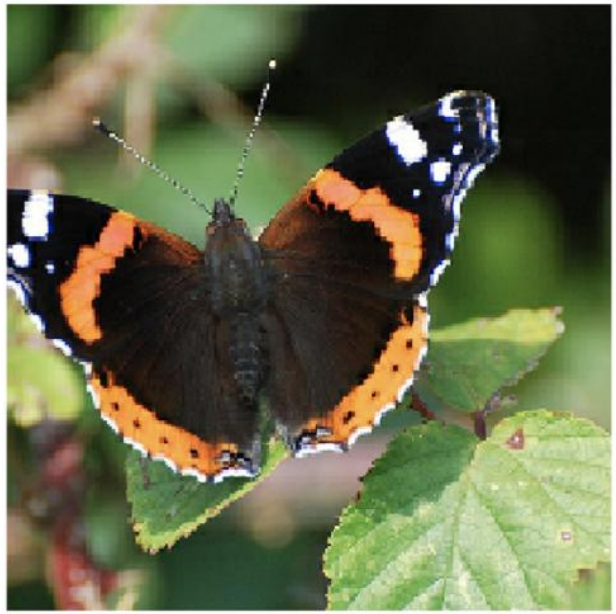


MAE Reconstruct Example

Masked input: 80%

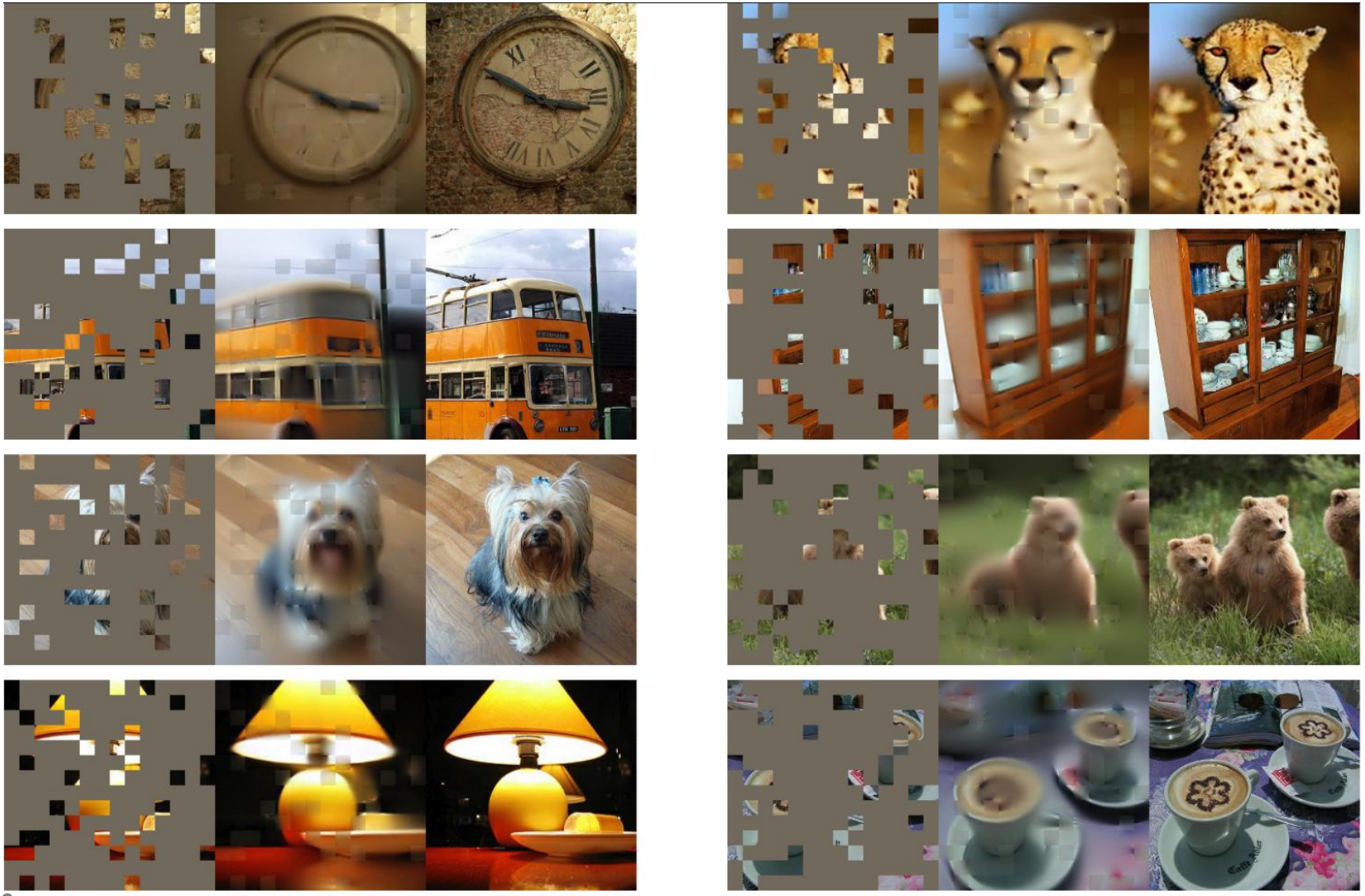


MAE's guess



Groud truth

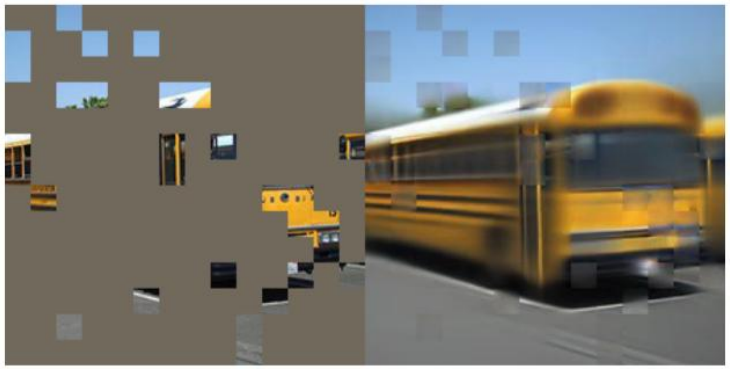
ImageNet val set (unseen)



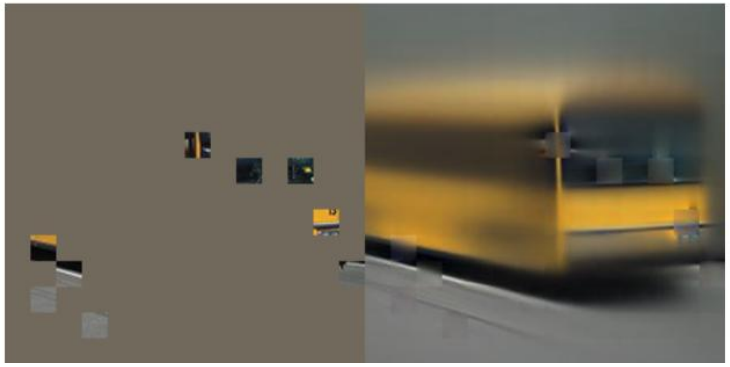
MAE can generalize

original

75% mask



85% mask



95% mask

MAE can generalize

original

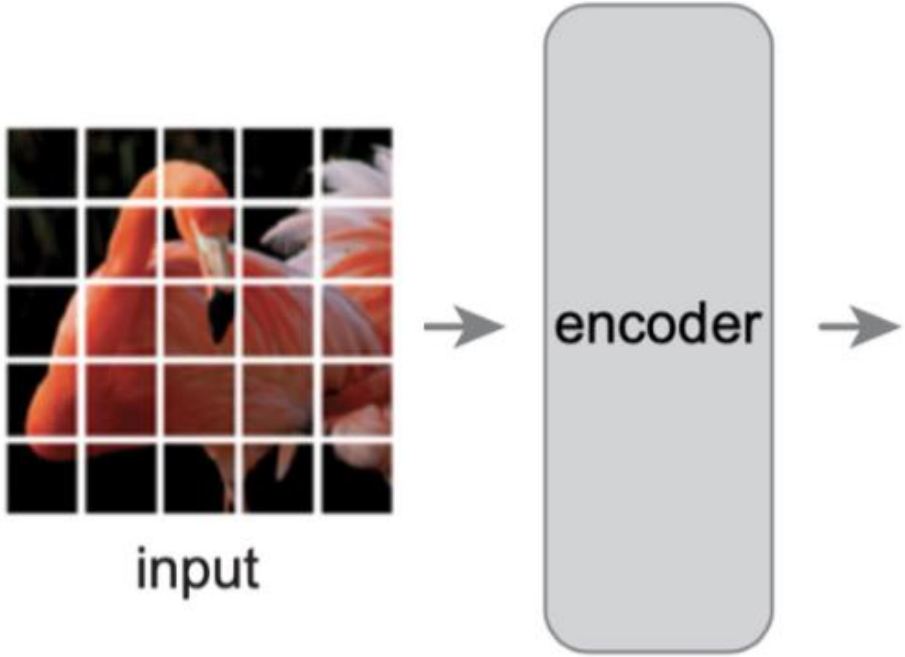
75% mask

85% mask

95% mask

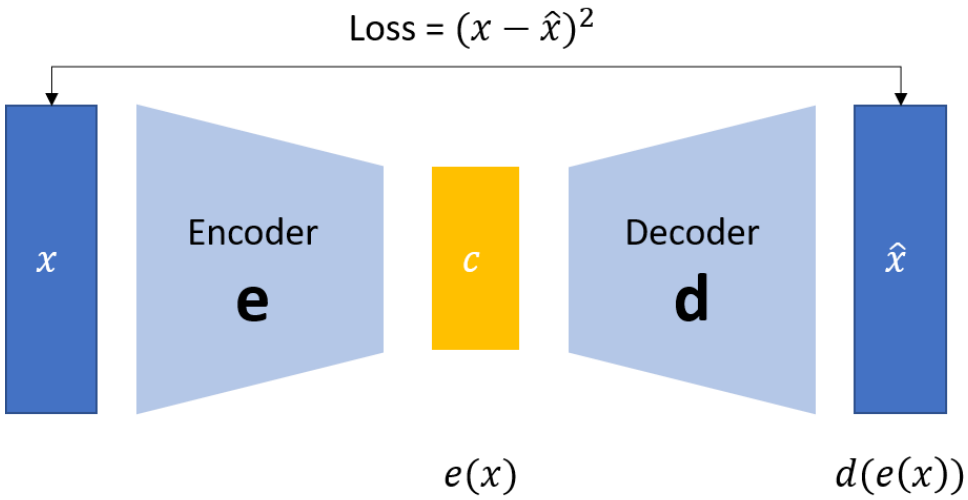
MAE for Downstream Tasks

- After MAE pre-training, just throw away the decoder
- Encoder is used for representations with full-sequence input

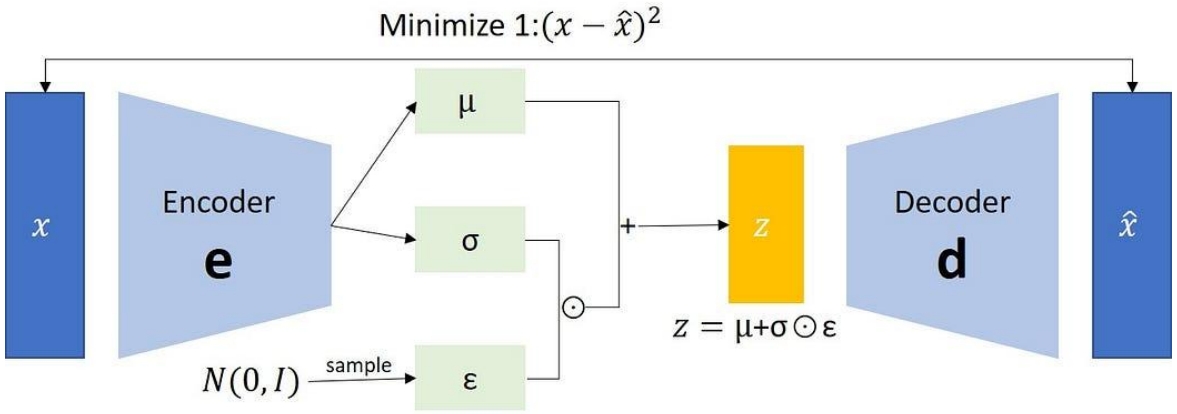


Variational Autoencoder (VAE)

Autoencoder



VAE



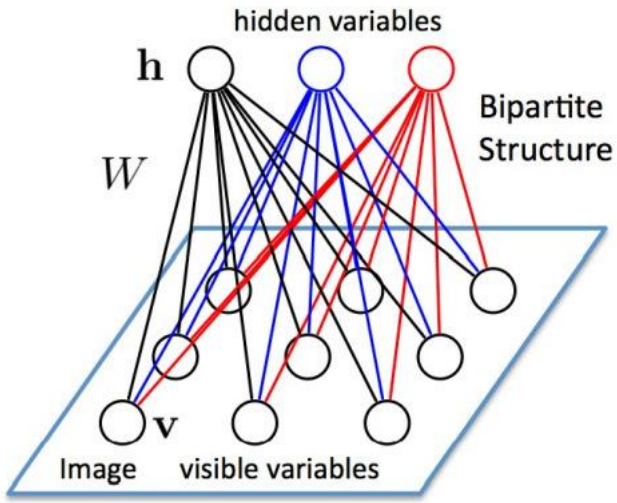
Minimize 2: $\frac{1}{2}\sum_{i=1}^{N}(\exp(\sigma_i) - (1+\sigma_i) + \mu_i{}^2)$

Deep Belief Network

Renaissance of Neural Network--- “Deep Learning,” 2006

- Geoff Hinton invented Deep Belief Networks (DBN) to make neural net learning fast and effective; *Science, 2006*
  - Pre-train each layer from bottom up
  - Each pair of layers is an Restricted Boltzmann Machine (RBM)
  - Jointly fine-tune all layers using back-propagation

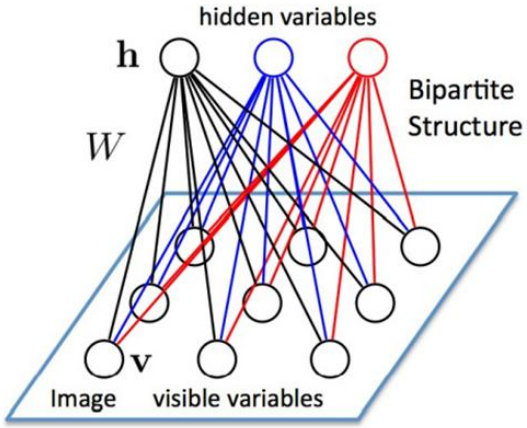
受限玻尔兹曼机（RBM）



Ludwig Edward Boltzmann 1844.2.20-1906.9.5
奥地利物理学家，热力学和统计物理学的奠基人之一

- 假设有一个二分图，每一层的节点之间没有连接，一层是可视层，即输入数据层（v），一层是隐藏层（h）。
- 如果假设所有的节点都是随机二进制变量节点（只能0或1），同时假设全概率分布p(v, h)满足Boltzmann分布，我们称这个模型为RBM。

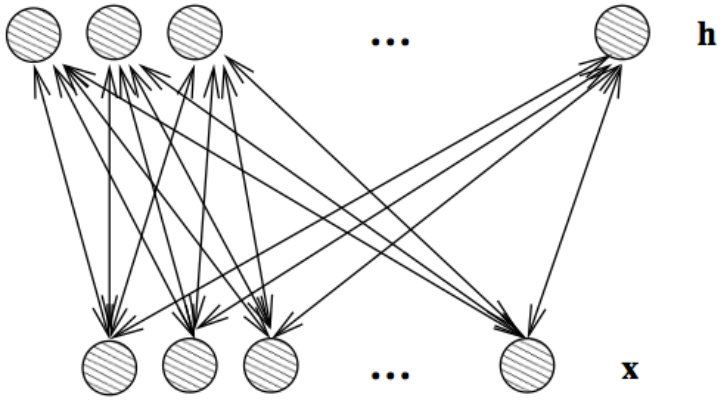
受限玻尔兹曼机（RBM）



- RBM有n个可视节点和m个隐藏节点，其中每个可视节点只和m个隐藏节点相关，与其他可视节点独立，对于每个隐藏节点也是只受n个可视节点的影响。
- RBM的可调节参数：可视层与隐藏层之间的权重矩阵 W_{mxn}，可视节点的偏移量 $b=(b_1, b_2, \cdots, b_n)$，隐藏节点的偏移量 $c=(c_1, c_2, \cdots, c_n)$，它们决定了RBM将一个n维样本编码成一个什么样的m维样本。

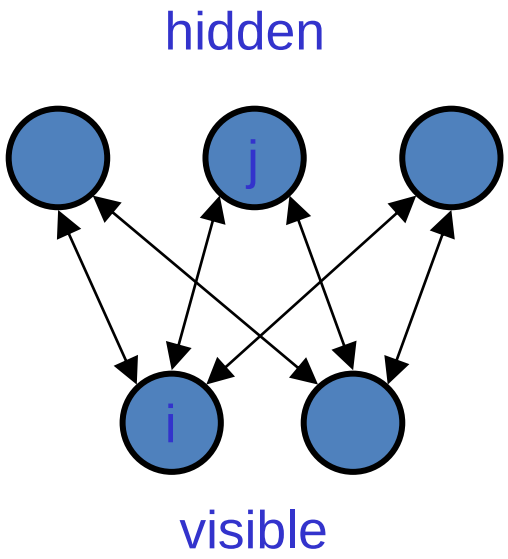
Deep Belief Networks (DBN)

- Geoff Hinton (2006)
- Uses Greedy layer-wise training but each layer is an RBM (Restricted Boltzmann Machine)
- RBM is a constrained Boltzmann machine with
  - No lateral connections between hidden (h) and visible (x) nodes
  - Symmetric weights



Restricted Boltzmann Machines (RBM)

- We restrict the connectivity to make learning easier.
  - Only one layer of hidden units.
  - No connections between hidden units.
- In an RBM, the hidden units are conditionally independent given the visible states.
- So we can quickly get an unbiased sample from the posterior distribution when given a data-vector.



RBM: Weights → Energies → Probabilities

- Joint distribution $\mathrm{p}(\mathbf{v}, \mathbf{h}; \theta)$ is defined in terms of an energy function $\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta)$

$$\mathrm{p}(\mathbf{v}, \mathbf{h}; \theta) = \frac{exp\left(-\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta)\right)}{Z}$$

- For a Bernoulli-Bernoulli RBM

$$\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}\, v_i h_j - \sum_{i=1}^{V} b_i v_i - \sum_{j=1}^{H} a_j h_j$$

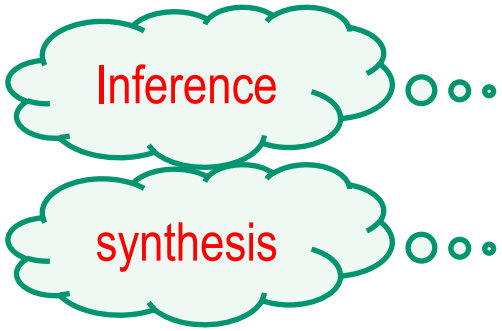
- For a Gaussian-Bernoulli RBM

$$\mathrm{E}(\mathbf{v}, \mathbf{h}; \theta) = -\sum_{i=1}^{V}\sum_{j=1}^{H} w_{ij}\, v_i h_j + \frac{1}{2}\sum_{i=1}^{V} (v_i - b_i)^2 - \sum_{j=1}^{H} a_j h_j$$

- $\mathrm{p}(\mathbf{v}, \mathbf{h}; \theta)$ → generative model!

Restricted Boltzmann Machines (RBM)

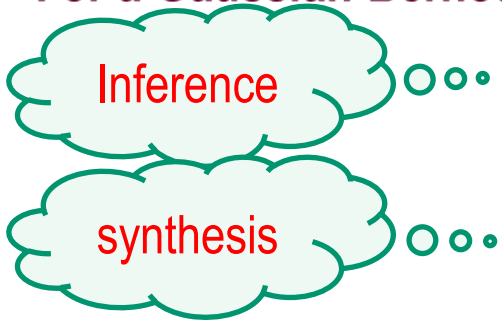
- Conditional probabilities are very easy to calculate
- For a Bernoulli-Bernoulli RBM



$$p(h_j = 1|\mathbf{v};\theta) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right)$$

$$p(v_i = 1|\mathbf{h};\theta) = \sigma\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i\right)$$

- For a Gaussian-Bernoulli RBM



$$p(h_j = 1|\mathbf{v};\theta) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right)$$

$$p(v_i|\mathbf{h};\theta) = N\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i, 1\right)$$

Restricted Boltzmann Machines (RBM)

从联合分布中导出条件分布是直观的：

$$P(\boldsymbol{h} \mid \boldsymbol{v}) = \frac{P(\boldsymbol{h}, \boldsymbol{v})}{P(\boldsymbol{v})} \tag{20.7}$$

$$= \frac{1}{P(\boldsymbol{v})} \frac{1}{Z} \exp\left\{\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h}\right\} \tag{20.8}$$

$$= \frac{1}{Z'} \exp\left\{\boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W}\boldsymbol{h}\right\} \tag{20.9}$$

$$= \frac{1}{Z'} \exp\left\{\sum_{j=1}^{n_h} \boldsymbol{c}_j^\top \boldsymbol{h}_j + \sum_{n_h}^{j=1} \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j\right\} \tag{20.10}$$

$$= \frac{1}{Z'} \prod_{j=1}^{n_h} \exp\left\{\boldsymbol{c}_j^\top \boldsymbol{h}_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j\right\}. \tag{20.11}$$

- 从无向图的性质可知.
  - 在给定可观测变量时，隐变量之间互相条件独立.
  - 在给定隐变量时，可观测变量之间也互相条件独立.

Restricted Boltzmann Machines (RBM)

从联合分布中导出条件分布是直观的：

$$P(\boldsymbol{h} \mid \boldsymbol{v}) = \frac{P(\boldsymbol{h}, \boldsymbol{v})}{P(\boldsymbol{v})} \tag{20.7}$$

$$= \frac{1}{P(\boldsymbol{v})} \frac{1}{Z} \exp\left\{\boldsymbol{b}^\top \boldsymbol{v} + \boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}\right\} \tag{20.8}$$

$$= \frac{1}{Z'} \exp\left\{\boldsymbol{c}^\top \boldsymbol{h} + \boldsymbol{v}^\top \boldsymbol{W} \boldsymbol{h}\right\} \tag{20.9}$$

$$= \frac{1}{Z'} \exp\left\{\sum_{j=1}^{n_h} \boldsymbol{c}_j^\top \boldsymbol{h}_j + \sum_{n_h}^{j=1} \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j\right\} \tag{20.10}$$

$$= \frac{1}{Z'} \prod_{j=1}^{n_h} \exp\left\{\boldsymbol{c}_j^\top \boldsymbol{h}_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j} \boldsymbol{h}_j\right\}. \tag{20.11}$$

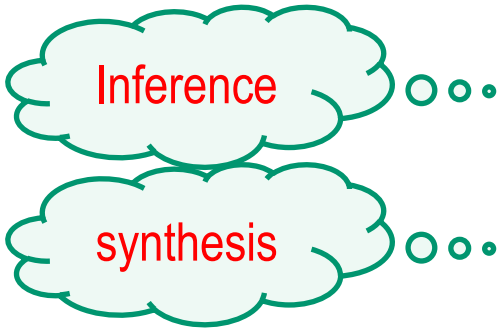
$$P(h_j = 1 \mid \boldsymbol{v}) = \frac{\tilde{P}(h_j = 1 \mid \boldsymbol{v})}{\tilde{P}(h_j = 0 \mid \boldsymbol{v}) + \tilde{P}(h_j = 1 \mid \boldsymbol{v})} \tag{20.12}$$

$$= \frac{\exp\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\}}{\exp\{0\} + \exp\{c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}\}} \tag{20.13}$$

$$= \sigma(c_j + \boldsymbol{v}^\top \boldsymbol{W}_{:,j}). \tag{20.14}$$

Restricted Boltzmann Machines (RBM)

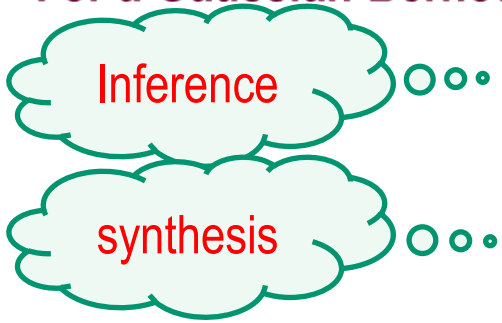
- Conditional probabilities are very easy to calculate
- For a Bernoulli-Bernoulli RBM



$$p(h_j = 1|\mathbf{v};\theta) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right)$$

$$p(v_i = 1|\mathbf{h};\theta) = \sigma\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i\right)$$

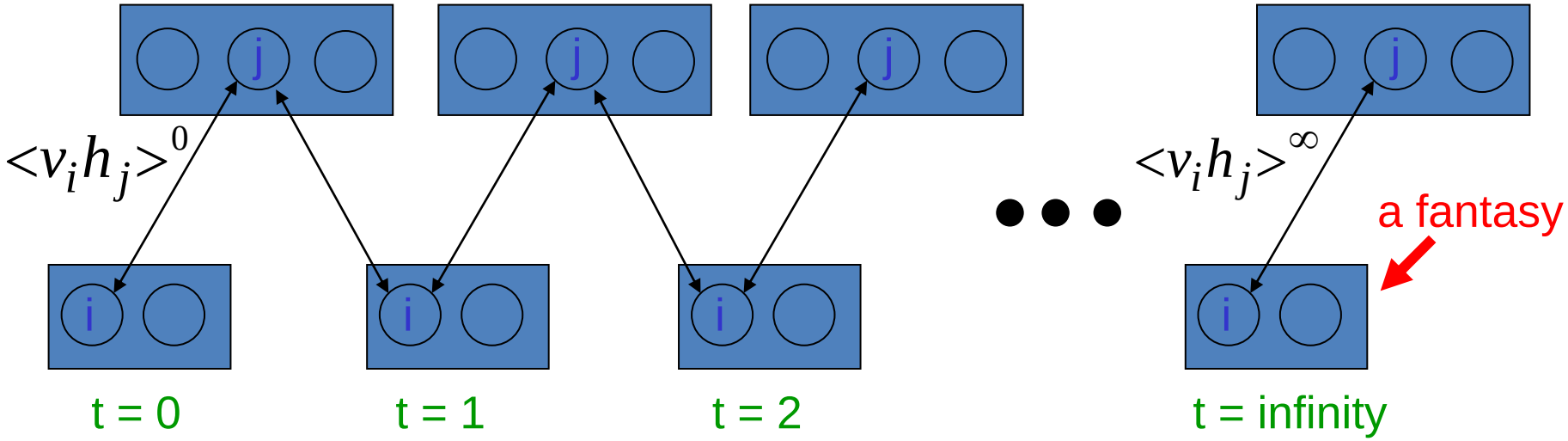
- For a Gaussian-Bernoulli RBM



$$p(h_j = 1|\mathbf{v};\theta) = \sigma\left(\sum_{i=1}^{V} w_{ij}\, v_i + a_j\right)$$

$$p(v_i|\mathbf{h};\theta) = N\left(\sum_{j=1}^{H} w_{ij}\, h_j + b_i, 1\right)$$

Maximum likelihood learning for RBM



Start with a training vector on the visible units.

Then alternate between updating all the hidden units in parallel and updating all the visible units in parallel.

$$\frac{\partial \log p(v)}{\partial w_{ij}} = <v_i h_j>^0 - <v_i h_j>^\infty$$

Maximum likelihood learning for RBM

$$\ln P(\mathbf{v}) = \ln\left(\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\right) - \ln\left(\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\right)$$

$$\frac{\partial \log P(\mathbf{v})}{\partial \boldsymbol{\theta}} = \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right)}{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} - \frac{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}\left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right)}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}$$

$$= \sum_{\mathbf{h}}\left(P(\mathbf{h}|\mathbf{v})\left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right)\right) - \sum_{\mathbf{v},\mathbf{h}}\left(P(\mathbf{v},\mathbf{h})\left(-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right)\right)$$

$$= \mathbb{E}_{P(\mathbf{h}|\mathbf{v})}\left[-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right] - \mathbb{E}_{P(\mathbf{h},\mathbf{v})}\left[-\frac{\partial E(\mathbf{v},\mathbf{h})}{\partial \boldsymbol{\theta}}\right]$$

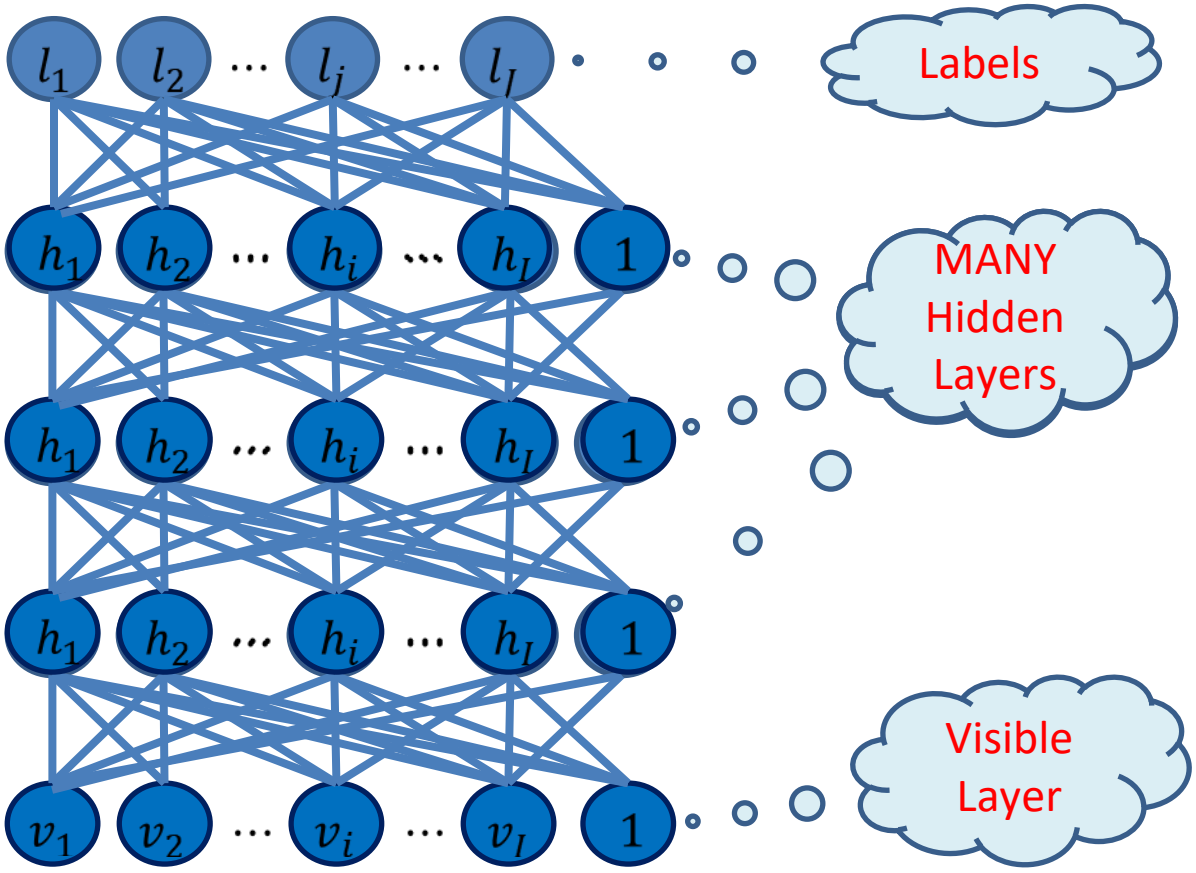
Training RBMs

- $\Delta w_{ij} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}$
- Approximate $\langle v_i h_j \rangle_{model}$
  i. Initialize $\mathbf{v_0}$ at data
  ii. Sample $\mathbf{h_0} \sim p(\mathbf{h}|\mathbf{v_0})$
  iii. Sample $\mathbf{v_1} \sim p(\mathbf{v}|\mathbf{h_0})$
  iv. Sample $\mathbf{h_1} \sim p(\mathbf{h}|\mathbf{v_1})$
  v. Call $(\mathbf{v_1}, \mathbf{h_1})$ a sample from the model.
- $(\mathbf{v}_\infty, \mathbf{h}_\infty)$ is a true sample from the model. $(\mathbf{v_1}, \mathbf{h_1})$ is a very rough estimate but worked
- Contrastive divergence algorithm (CD)

Building a Deep Network

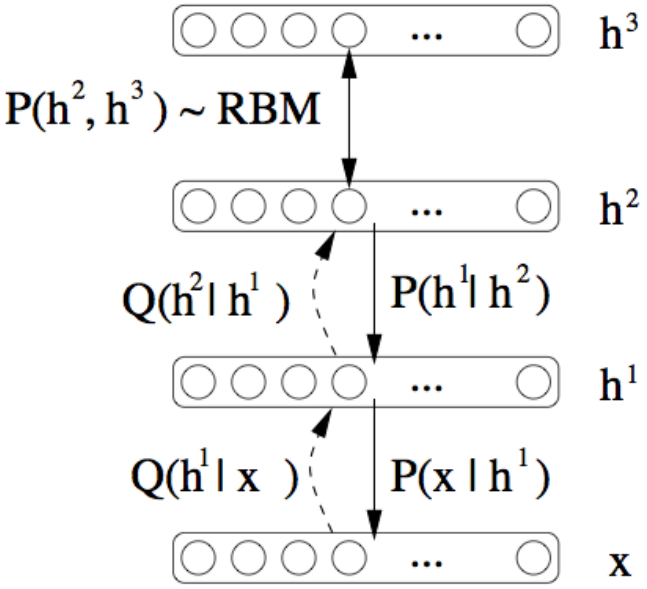
- This is the main reason why RBMs are interesting (as a building block)
- First train a layer of hidden units that receive input directly from the data (image, speech, coded text, etc).
- Then treat the activations of hidden units (the trained "features") as if they were "data" and learn features of features in a second hidden layer.
- It can be proved that each time we add another layer of features we improve a variational lower bound on the log probability of the training data.
  - The proof is complicated (Hinton et al, 2006)
  - Based on an equivalence between an RBM and a deep directed model

Deep Belief Network



Deep Belief Network Training

- Same Greedy Layer-wise approach
- First train lowest RBM ($h^0 - h^1$) using RBM update algorithm
- Freeze weights and train subsequent RBM layers
- Then connect final outputs to a supervised model and train that model
- Finally, unfreeze all weights, and train full DBN with supervised model to fine-tune weights



Fine-tuning for discrimination

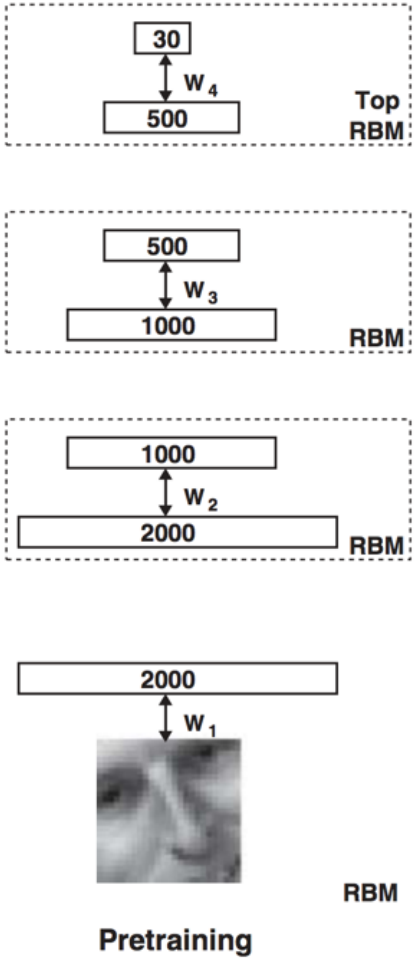
- First learn one layer at a time greedily.
- Then treat this as “pre-training” that finds a good initial set of weights which can be fine-tuned by a local search procedure.
- Backpropagation can be used to fine-tune the model for better discrimination.

Fine Tuning DNN after pre-training: Optimization view

- Stacking RBMs one layer at a time scales well to really big networks
- Do not start back-propagation until sensible feature detectors are found by RBM pre-training that should already be very helpful for the discrimination task.
- Back-propagation only needs to perform a local search from a sensible starting point.

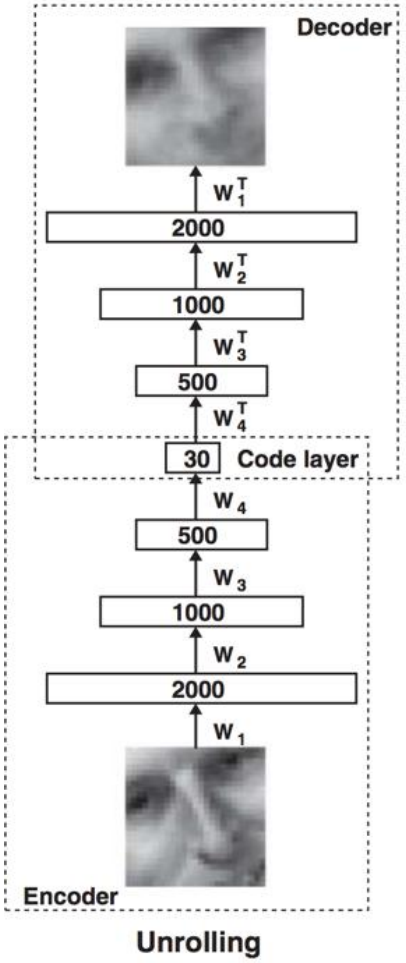
深度自编码器 - DAE

- 如图所示，深度自编码器利用RBM对每一层进行预训练。
- 输入图像首先转换成一维的向量，作为第一个RBM的可见层，其隐层需要人为指定，图中为2000个节点。
- 第一个RBM训练好之后，其隐含层作为第二个RBM的可见层，训练新的RBM。图中，第二个RBM的隐含层有1000个节点。
- 然后以同样的方式继续训练两个新的RBM。



深度自编码器 - DAE

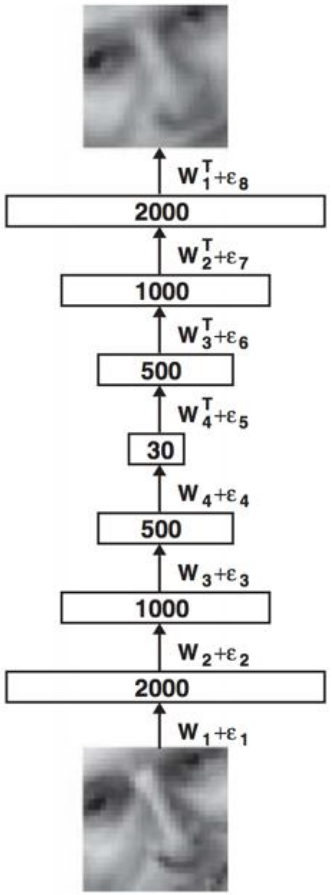
- 有了训练好的RBM之后，我们将这些RBM堆叠起来，作为编码部分，即Encoder。
- Decoder的构造比较简单，将Encoder的结构对称一下，系数使用RBM训练好的系数的转置作为初始值即可。
- 由于每一个RBM训练都会学到一些信息，并且权重使用相同的权重，因此网络可以实现重建输入的功能。
- 这个形成深度自编码器的过程，叫做 Unrolling 或者 Unfolding。



Unrolling

深度自编码器 - DAE

- 直接使用转置的权重初始化后得到的深度自编码器，虽然能够实现一些重建输入数据的功能，但是效果并不理想。
- 需要使用BP算法对网络的参数进行整体的调整，减小重建图像和原始图像之间的误差。
- 这种参数的微调成为 Fine-tuning.



Fine-tuning

深度自编码器降维

- 数据降维
  - A图是PCA对MNIST降维结果
  - B图是Deep Auto-Encoeder的降维结果
  - B图降维效果好于A图

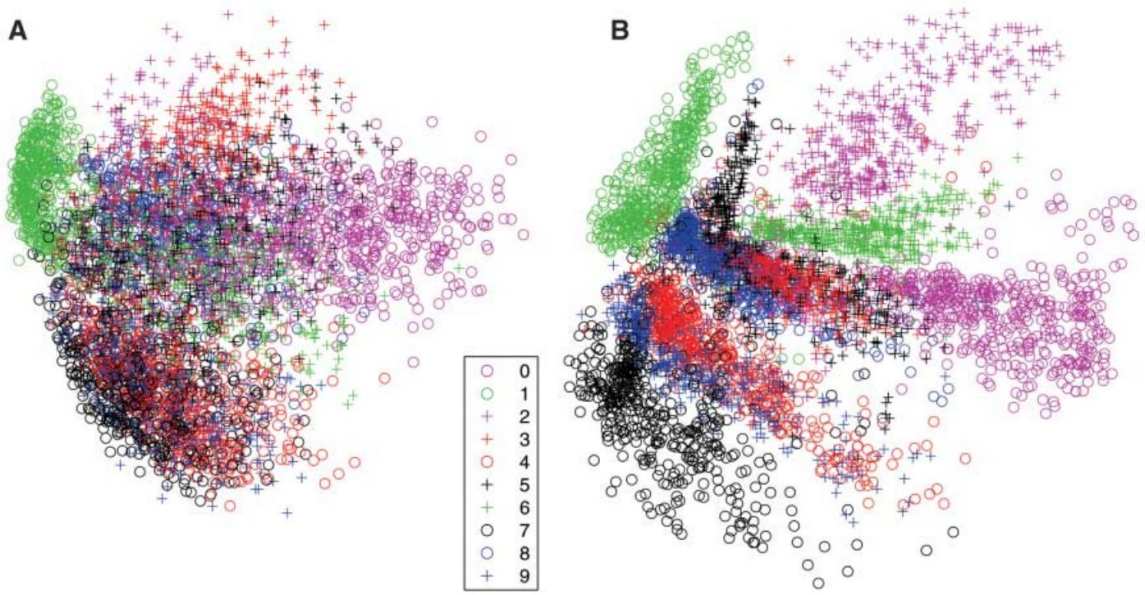
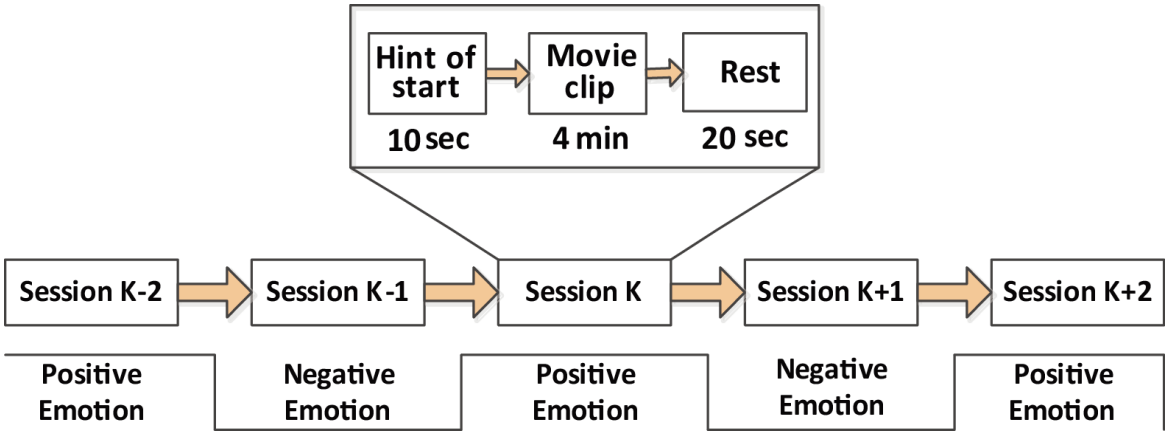


Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.

Applications to EEG-based Emotion Recognition

Emotion Experiments



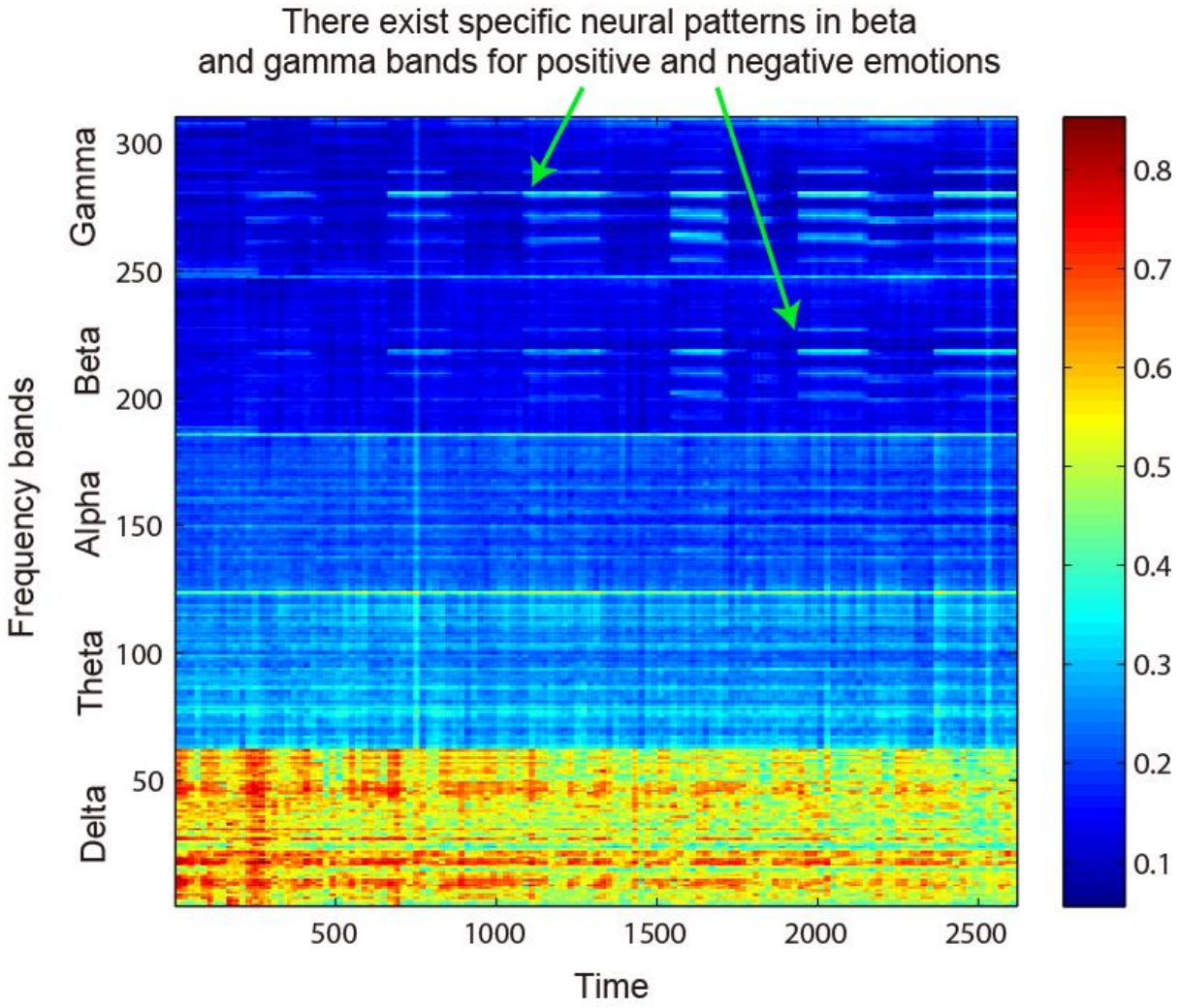
About 15 movie clips were used in each experiment, approximately 5 clips for each emotion (positive, neutral and negative).

Comparisons of SVM and DBN

THE MEAN ACCURACIES AND STANDARD DEVIATIONS (%) OF SVM AND DNN FOR DIFFERENT KINDS OF FEATURES

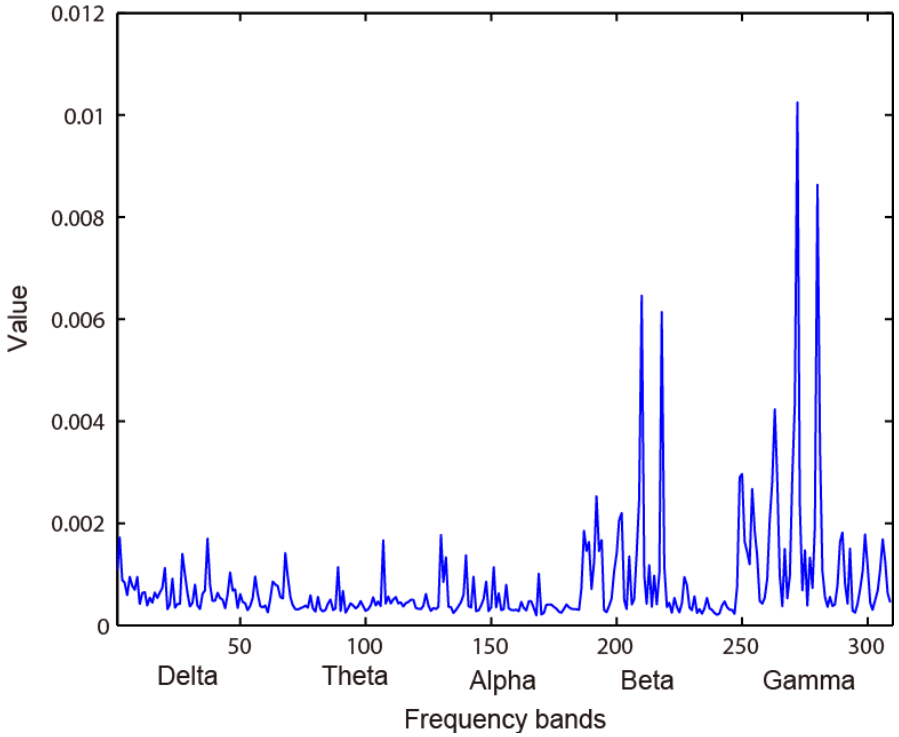
Feature	Classifier	Delta	Theta	Alpha	Beta	Gamma	Total
PSD	SVM	58.03/15.39	57.26/15.09	59.04/15.75	**73.34/15.20**	71.24/16.38	59.60/15.93
	DNN	60.05/16.66	55.03/13.88	52.79/15.38	60.68/21.31	**63.42/19.66**	61.90/16.65
DE	SVM	60.50/14.14	60.95/10.20	66.64/14.41	80.76/11.56	79.56/11.38	**83.99/09.72**
	DNN	64.32/12.45	60.77/10.42	64.01/15.97	78.92/12.48	79.19/14.58	**86.08/08.34**
DASM	SVM	48.87/10.49	53.02/12.76	59.81/14.67	**75.03/15.72**	73.59/16.57	72.81/16.57
	DNN	48.79/09.62	51.59/13.98	54.03/17.05	69.51/15.22	70.06/18.14	**72.73/15.93**
RASM	SVM	47.75/10.59	51.40/12.53	60.71/14.57	74.59/16.18	74.61/15.57	**74.74/14.79**
	DNN	48.05/10.37	50.62/14.02	56.15/15.28	70.31/15.62	68.22/18.09	**71.30/16.16**
DCAU	SVM	55.92/14.62	57.16/10.77	61.37/15.97	75.17/15.58	76.44/15.41	**77.38/11.98**
	DNN	54.58/12.81	56.94/12.54	57.62/13.58	70.70/16.33	72.27/16.12	**77.20/14.24**

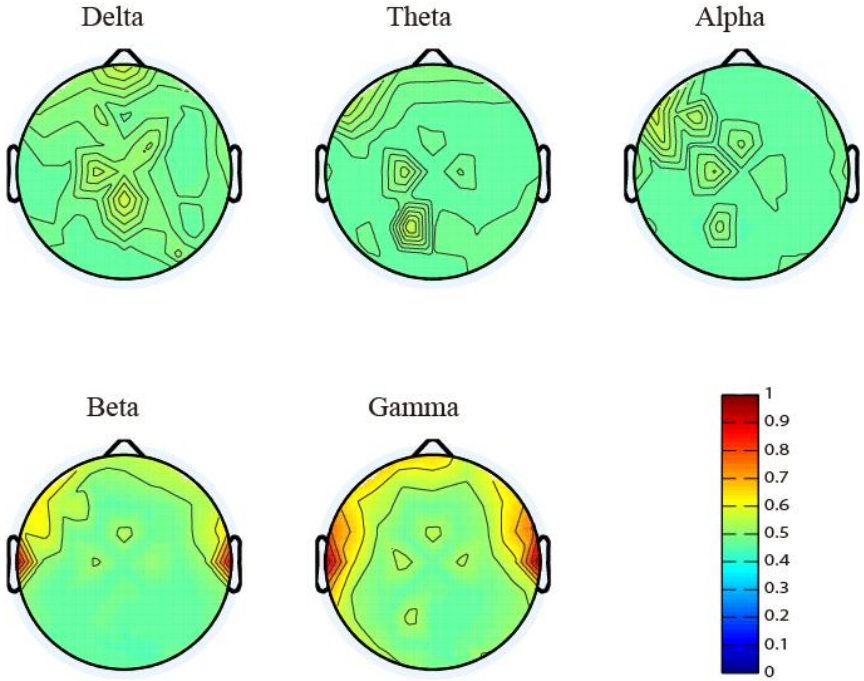
Neural Patterns in beta and gamma bands



Wei-Long Zheng, Jia-Yi Zhu, Yong Peng and Bao-Liang Lu, " EEG-based Emotion Classification using Deep Belief Networks''. in IEEE ICME2014

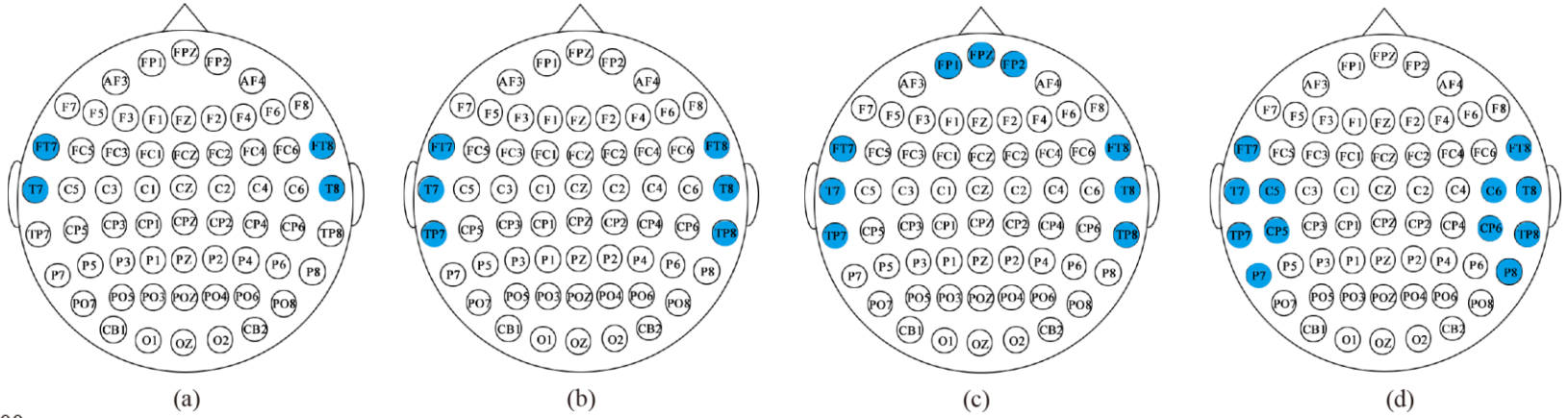
Feature Selection

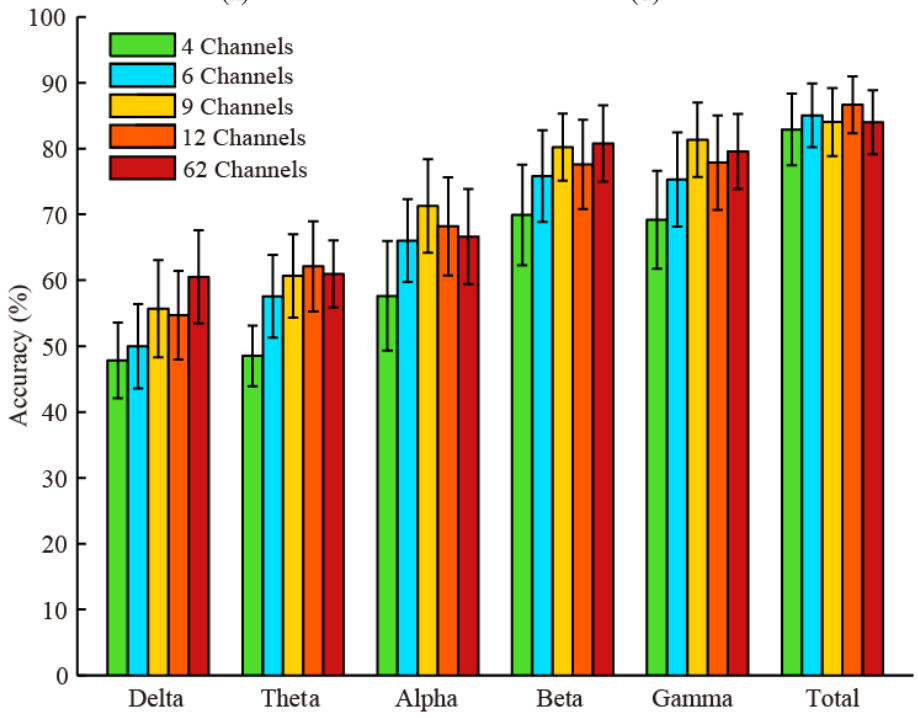




The mean weight distribution of trained deep belief networks

Critical Channels





We design four different profiles of selected electrodes placements according to features of high peaks in the weight distribution and asymmetric properties in emotion processing. Profiles of 6, 9 and 12 channels achieve better performance than 62 channels.

Thanks!