[illegible] bing CNN + Caffe
zhe

Neural Networks

1-Hidden Layer Neural Networks

Rosenblatt

"Shallow Learning" G. Hinton

- A two-stage regression or classification model

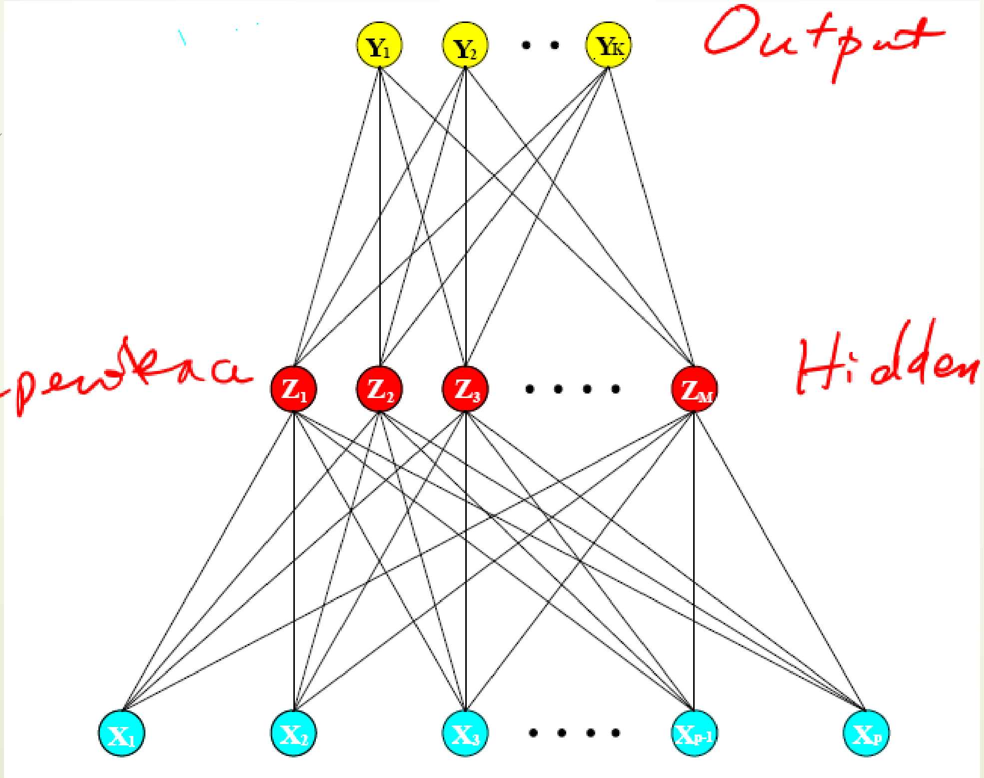
Yann LeCun

杨立昆

"MNIST"

Bengio

McGill



"ICLR"

Single-Hidden Layer NN

- Z称为导出特征，在神经网络中也成为隐藏层。先由输入的线性组合创建Z，再以Y为目标用Z的线性组合建立模型

x, Y   x→z   z→Y

M = A B

α   β

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T X),\ m = 1, \ldots, M,$$

$$T_k = \beta_{0k} + \beta_k^T Z,\ k = 1, \ldots, K,$$

$Y_k \leftarrow$

$$f_k(X) = g_k(T),\ k = 1, \ldots, K,$$

$$Z = (Z_1, Z_2, \ldots, Z_M), \text{ and } T = (T_1, T_2, \ldots, T_K).$$

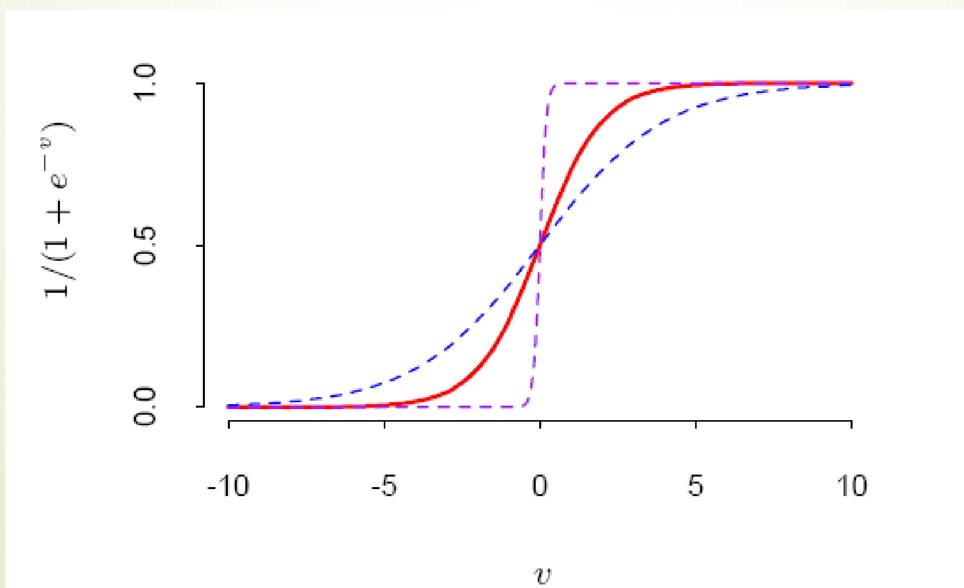
Matrix Factorization

Sigmoid Hidden Layer

cy

logit

- 激活函数$\sigma(\nu)=\exp(y)/(1+\exp(y))=1/(1+\exp(-y))$
- 神经网络源于人脑开发模型，神经元接收到的信号超过阀值时被激活。由于梯度下降法训练需要光滑的性质，阶梯函数被光滑阀函数取代。



S-shape

Softmax Output

- 输出函数 $g_k(T)$ 是对于向量T的最终变换，早期的K分类使用的是恒等函数，后来被softmax函数所取代，因其可以产生和为1的正估计。

$$g_k(T) = \frac{e^{T_k}}{\sum_{\ell=1}^{K} e^{T_\ell}}.$$

Projection Pursuit Regression
投影寻踪模型

- Ridge function $g_m(\langle w,X\rangle)$: 先将X投影于某一方向，再用得到的标量进行回归
- Universal approximation: M任意大时可以任意好的逼近空间中的任意连续函数

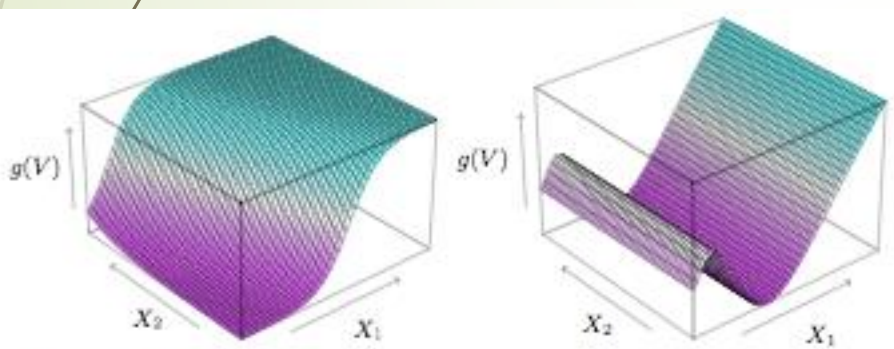


FIGURE 11.1. *Perspective plots of two ridge functions.*
(Left:) $g(V) = 1/[1+\exp(-5(V-0.5))]$, *where* $V = (X_1 + X_2)/\sqrt{2}$.
(Right:) $g(V) = (V+0.1)\sin(1/(V/3+0.1))$, *where* $V = X_1$.

$$f(X) = \sum_{m=1}^{M} g_m(\omega_m^T X).$$

generalized
Lin. model

Least Square Fitting for PPR

- 如何拟合投影寻踪模型
- 目标：误差函数的近似极小值

$$\sum_{i=1}^{N}\left[y_i-\sum_{m=1}^{M}g_m(\omega_m^T x_i)\right]^2$$

- 为避免过分拟合，对于输出函数g需要限制
- M的值通常作为前向分布策略的一部分来估计， 也可以由交叉验证来估计。

Weighted Least Square

- M=1时，首先给定一个投影方向的初值，通过光滑样条估计g
- 给定g， 在误差函数上对投影方向做极小化

- 舍弃了二阶导数之后，再带入误差函数得

$$g(\omega^T x_i) \approx g(\omega_{\text{old}}^T x_i) + g'(\omega_{\text{old}}^T x_i)(\omega - \omega_{\text{old}})^T x_i$$

- 对于右端进行最小二乘方回归，得到投影方向的新估计值，重复以上步骤得到

$$\sum_{i=1}^{N} \left[y_i - g(\omega^T x_i)\right]^2 \approx \sum_{i=1}^{N} g'(\omega_{\text{old}}^T x_i)^2 \left[\left(\omega_{\text{old}}^T x_i + \frac{y_i - g(\omega_{\text{old}}^T x_i)}{g'(\omega_{\text{old}}^T x_i)}\right) - \omega^T x_i\right]^2$$

$$(\omega_m, g_m)$$

Training of Neural Networks

- 未知参数称为权，用θ表示权的全集
- 对于回归和分类问题，我们分别使用误差的平方和，平方误差或互熵（离散）作为拟合的度量

$$\begin{aligned}\{\alpha_{0m},\alpha_m;\ m=1,2,\ldots,M\}&\quad M(p+1)\ \text{weights},\\ \{\beta_{0k},\beta_k;\ k=1,2,\ldots,K\}&\quad K(M+1)\ \text{weights}.\end{aligned}$$

θ

$$R(\theta)=\sum_{k=1}^{K}\sum_{i=1}^{N}(y_{ik}-f_k(x_i))^2.$$

$$R(\theta)=-\sum_{i=1}^{N}\sum_{k=1}^{K}y_{ik}\log f_k(x_i),$$

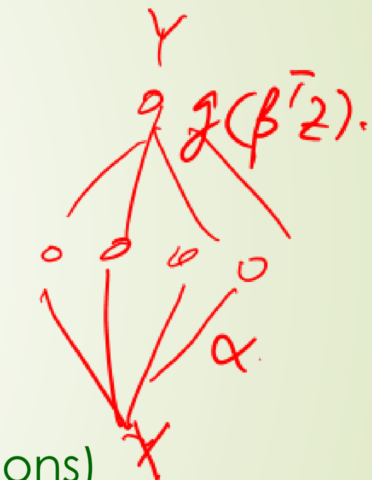
Least Square Fitting: Backpropogation (BP)

- 平方误差损失的反向传播细节

$$
\begin{aligned}
R(\theta) &\equiv \sum_{i=1}^{N} R_i \\
&= \sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))^2,
\end{aligned}
$$

$z = \sigma(\alpha^T x)$

$g_k(\beta^T z)$.



- 具有导数（Chain Rule for Composite Functions)

$$
\begin{aligned}
\frac{\partial R_i}{\partial \beta_{km}} &= -2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi}, \\
\frac{\partial R_i}{\partial \alpha_{m\ell}} &= -\sum_{k=1}^{K} 2(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{i\ell}.
\end{aligned}
$$

z'_α

神经网络的拟合

- 使用梯度下降法迭代，在第（r+1）次时有如下公式

$$\beta_{km}^{(r+1)} = \underline{\beta_{km}^{(r)}} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \beta_{km}^{(r)}},$$

$$\alpha_{m\ell}^{(r+1)} = \underline{\alpha_{m\ell}^{(r)}} - \gamma_r \sum_{i=1}^{N} \frac{\partial R_i}{\partial \alpha_{m\ell}^{(r)}},$$

步长

神经网络的拟合

- 如果将迭代前的公式写成如下形式

$2\left(y_{ik}-f_k(\beta^T z)\right)$

$$\frac{\partial R_i}{\partial \beta_{km}} = \delta_{ki} z_{mi},$$
$$\frac{\partial R_i}{\partial \alpha_{m\ell}} = s_{mi} x_{i\ell}.$$

- 其中 δ_{ki} 和 s_{mi} 分别是当前模型输出层，隐藏层的“误差”，并且满足(Backpropogation Equation)

$$s_{mi} = \sigma'(\alpha_m^T x_i) \sum_{k=1}^{K} \beta_{km} \delta_{ki},$$

Foreward and Backward

- 上面的关系称作反向传播方程 (BP Equation)
- 向前传递时固定当前权值，计算预测值 $\hat{f}_k(x_i)$
- 向后传递是计算误差 δ_{ki} ，进而又得到 s_{mi}
- 最后使用更新的误差值计算更新的梯度
- 反向传播方法具有简单性和局部特性，每个隐藏单元只传递信息

"SGD"

Stochastic Gradient Descent"

11.4 神经网络的拟合

- 迭代公式中的γ称为学习率(Learning rates)，此种迭代更新称为批学习(Batch learning)
- 对于批学习，学习率通常取常数，也可以在每次更新的时候通过极小化误差函数的线搜索来优化
- 使用在线学习(Online learning)，学习率: (1)随迭代次数递减到零，保证收敛；(2) 常数，跟踪环境变化

神经网络训练的一些问题

- 初始值？
  - 如果权值接近于0，则S型函数的运算大多是线性的，并且随着权值的增加变成非线性的
  - 权值恰为0导致0导数和良好的对称性，且算法永远不会前进，而以大权值开始常常导致很差的解
  - 初始化训练？
- 局部最优解
  - 多次随机初始值

神经网络训练的一些问题

- 过分拟合？
  - 权衰减是一种更加直接的正则化方法
  - 将惩罚项加入误差函数得到

$$R(\theta) + \lambda J(\theta)$$

$$J(\theta) = \sum_{km} \beta_{km}^2 + \sum_{m\ell} \alpha_{m\ell}^2$$

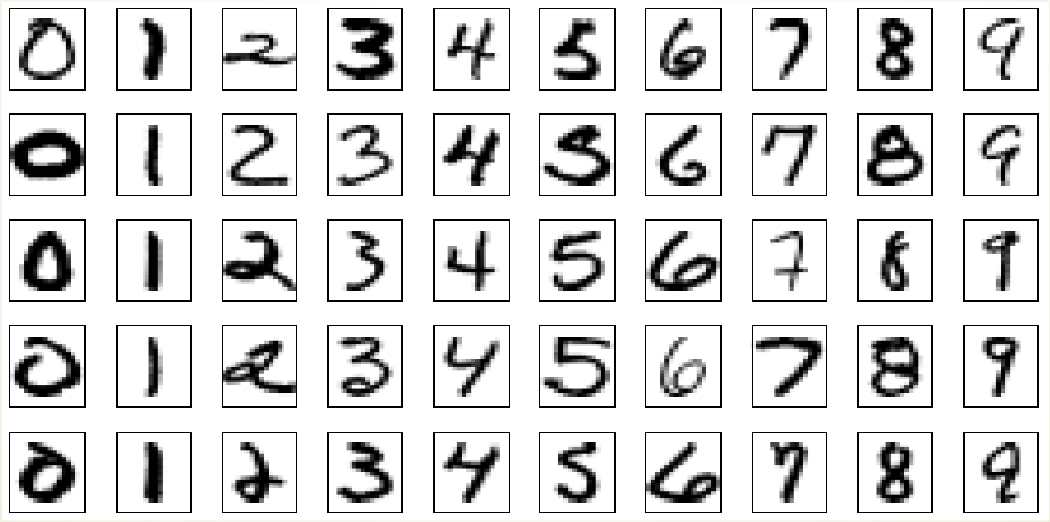
  - Ridge/LASSO etc.
  - λ是大于0的调整参数，较大的值使权值向0收缩。 值由交叉验证估计
  - 还可以weight sharing (restricted)

"Dropout."

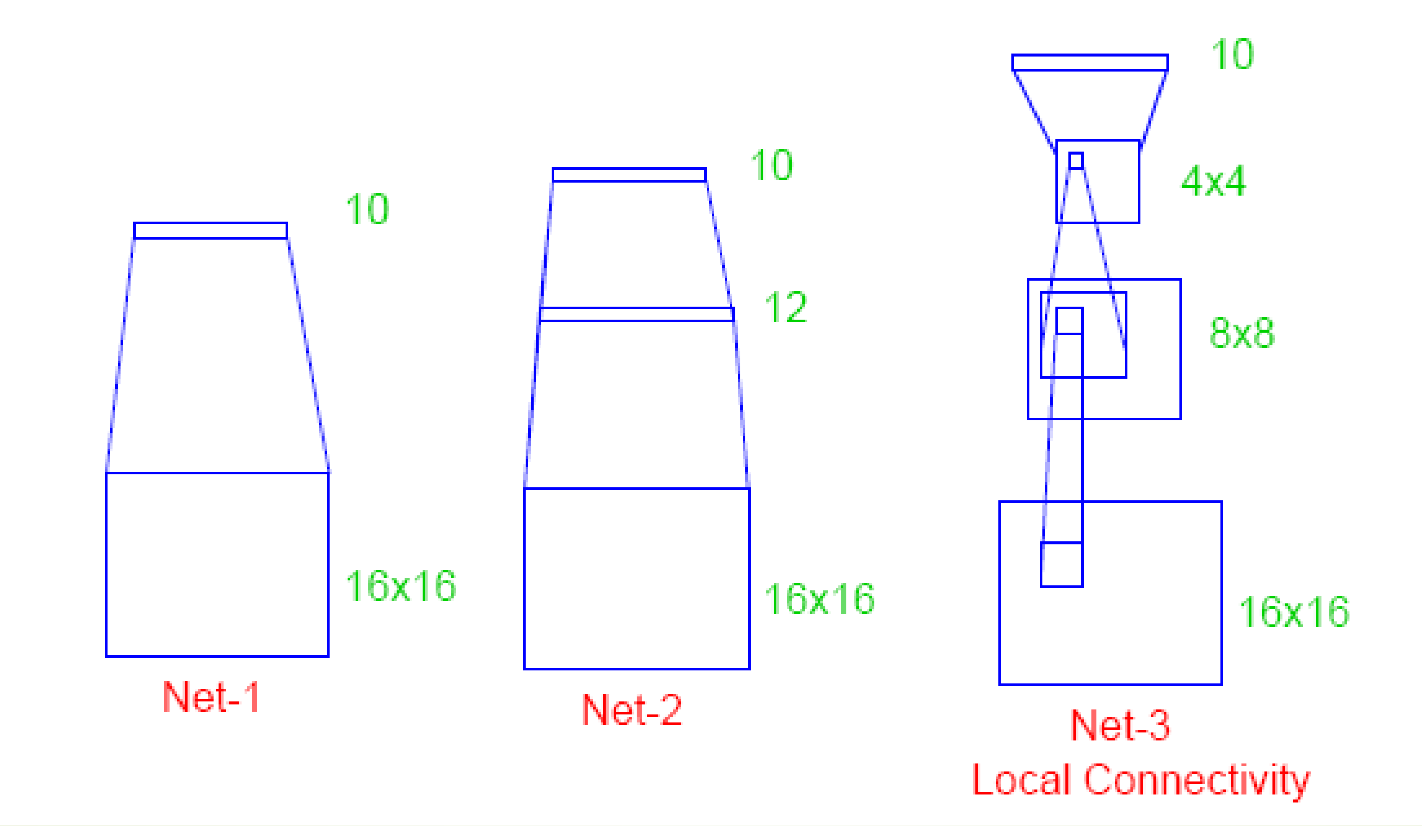
神经网络训练的一些问题

- 输入的scale对于结果的影响？
  - 最好对于所有的输入都进行标准化，这个可以保证在正则化过程中平等的对数据进行处理，而且为随机初值的选择提供一个有意义的值域
  - 一般在[-0.7, 0.7]上面随机选取均匀的权值
- 隐藏单元和层的数目：隐藏单元过少则模型可能不具备足够的灵活性，如果隐藏单元过多，则多余的收缩到0. 一般来说隐藏单元的数量在5到100之间，可以取合理大的数量，在用正则化加以训练，使得多余的变作0.
  - Deep networks?

Example: Hand-written Digits

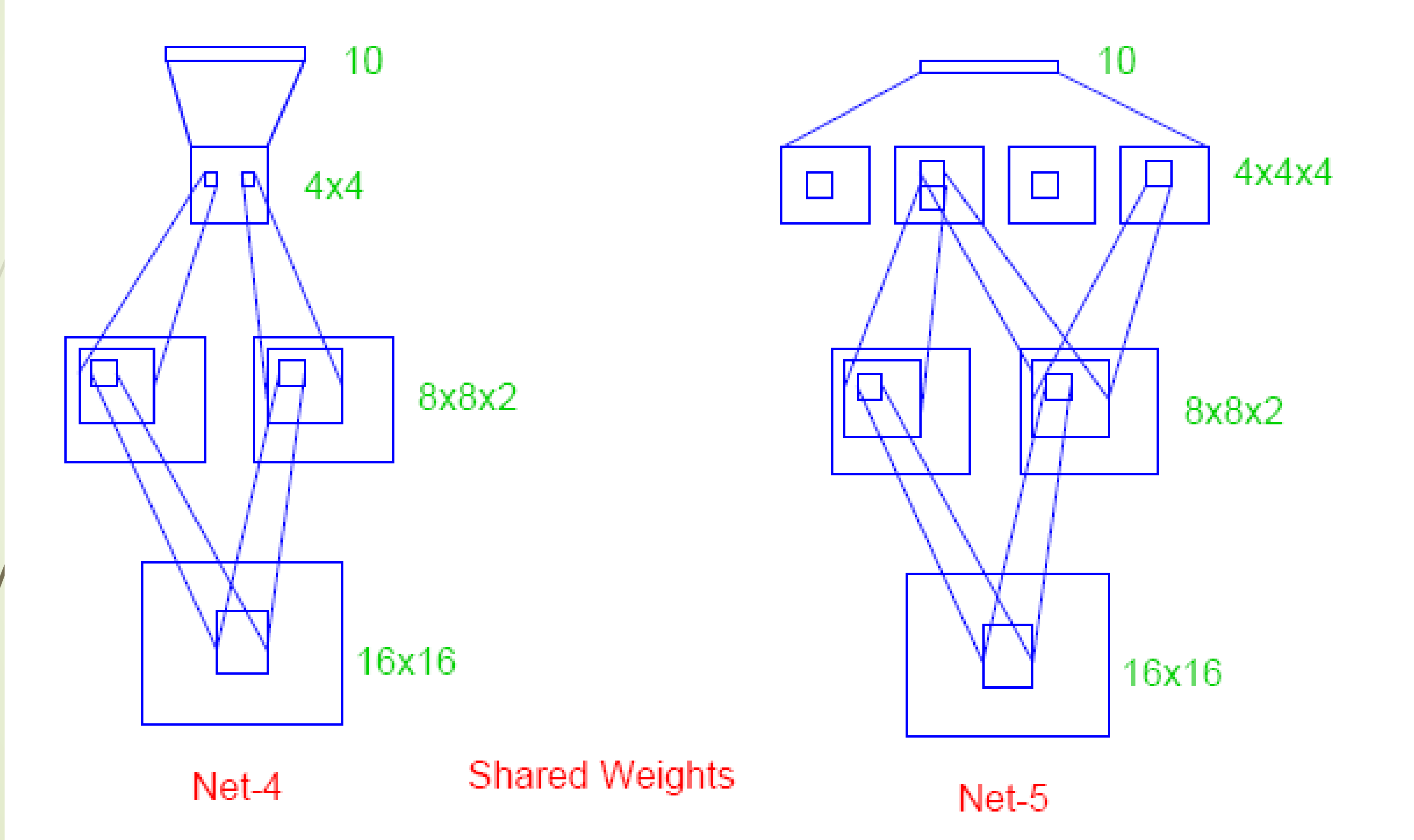


五种Network

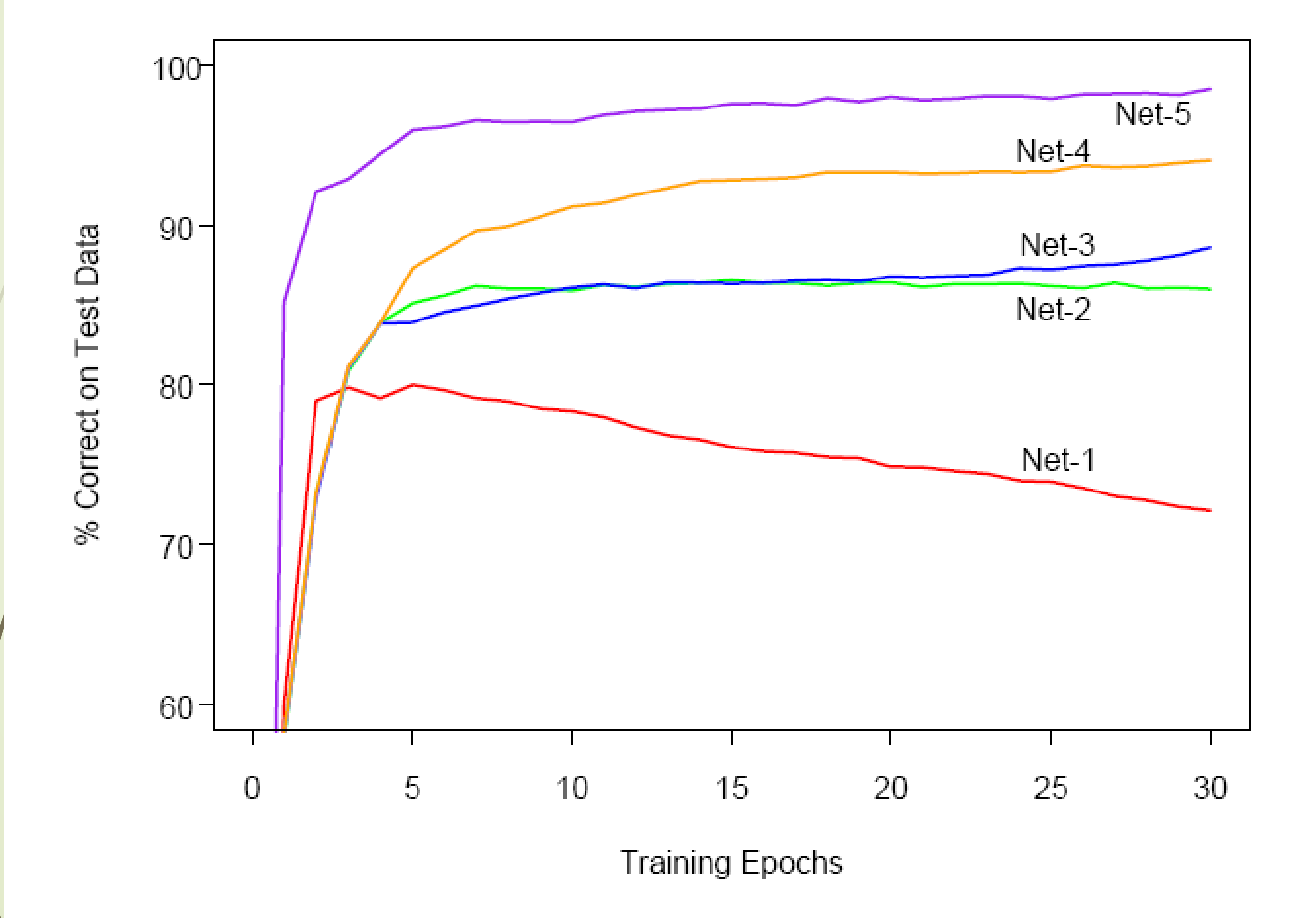


10
16x16
Net-1
10
12
16x16
Net-2
10
4x4
8x8
16x16
Net-3
Local Connectivity

五种Network Structures



Test Performance



Test Performance

	Network Architecture	Links	Weights	% Correct
Net-1:	Single layer network	2570	2570	80.0%
Net-2:	Two layer network	3214	3214	87.0%
Net-3:	Locally connected	1226	1226	88.5%
Net-4:	Constrained network 1	2266	1132	94.0%
Net-5:	Constrained network 2	5194	1060	98.4%

Deep Learning?

- 多层神经网络？Hierarchical Features?
- IPAM-UCLA 2012 Summer School: Deep Learning, Feature Learning:http://www.ipam.ucla.edu/programs/gss2012/ contains various video streams and slides
- Andrew Ng's course at Stanford, Unsupervised Feature Learning and Deep Learning: http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial

Deep vs. Shallow Learning

$$X \cong BC$$

- Shallow Learning as Matrix Factorization:
  - SVD: orthogonal B, C
  - Topic models: nonnegative B, C
  - Sparse coding/dictionary learning: B is dictionary (basis, frame, etc.), C is sparse
  - Conditional independence:

$$X = \sum_i b_i c_i^T, \quad b = [b_i], \quad C^T = [c_i^T]$$

$$b_{ik} \perp c_{ik} \mid k$$