S. Solla and Y. Le Cun. Constrained neural networks for pattern recognition. In P. Antognetti and V. Milutinovic, editors, *Neural Networks: Concepts, Applications and Implementations.* Prentice Hall, 1991.

# Constrained Neural Networks for Pattern Recognition

Sara A. Solla
AT&T Bell Laboratories, Holmdel, NJ 07733, USA
and Nordita, Blegdamsvej 17, DK-2100 Copenhagen, Denmark

Yann le Cun
AT&T Bell Laboratories, Holmdel, NJ 07733, USA

## 1 Introduction

Layered neural networks are of interest as a tool to implement input-output maps. This work explores the ability of such architectures to perform pattern recognition tasks.

The ensemble of all possible network configurations compatible with a fixed architecture is explored to define a probability distribution over the space of input-output maps. Such distribution fully describes the functional capabilities of the chosen architecture. Its entropy measures the intrinsic functional diversity of the network ensemble.

Supervised learning is formulated as an optimization problem, resulting in a monotonic decrease of the effective volume of configuration space through the exponential elimination of network configurations incompatible with the examples of the desired map. Such contraction results in increased specificity in the functional capabilities of the ensemble, and a monotonic entropy reduction.

The residual entropy of the ensemble of trained networks monitors the emergence of generalization ability. As the residual entropy is decreased to zero, all ambiguity about the implemented map is eliminated. Only the desired map survives, and full generalization ability is achieved.

The number of examples needed to achieve an acceptable level of generalization ability is controlled by the intrinsic entropy of the chosen architecture, and can be decreased by reducing the number of independent parameters needed to specify the network configuration. Such reduction must be planned with care, not to destroy the ability to implement the desired input-output map.

The strategy is to increase the specificity towards the implementation of the desired task by incorporating prior knowledge about the task onto the architecture. Fully connected, unrestricted networks do not work well for the pattern recognition problem. Constrained networks better suited to this task use hidden units with locally connected receptive fields.

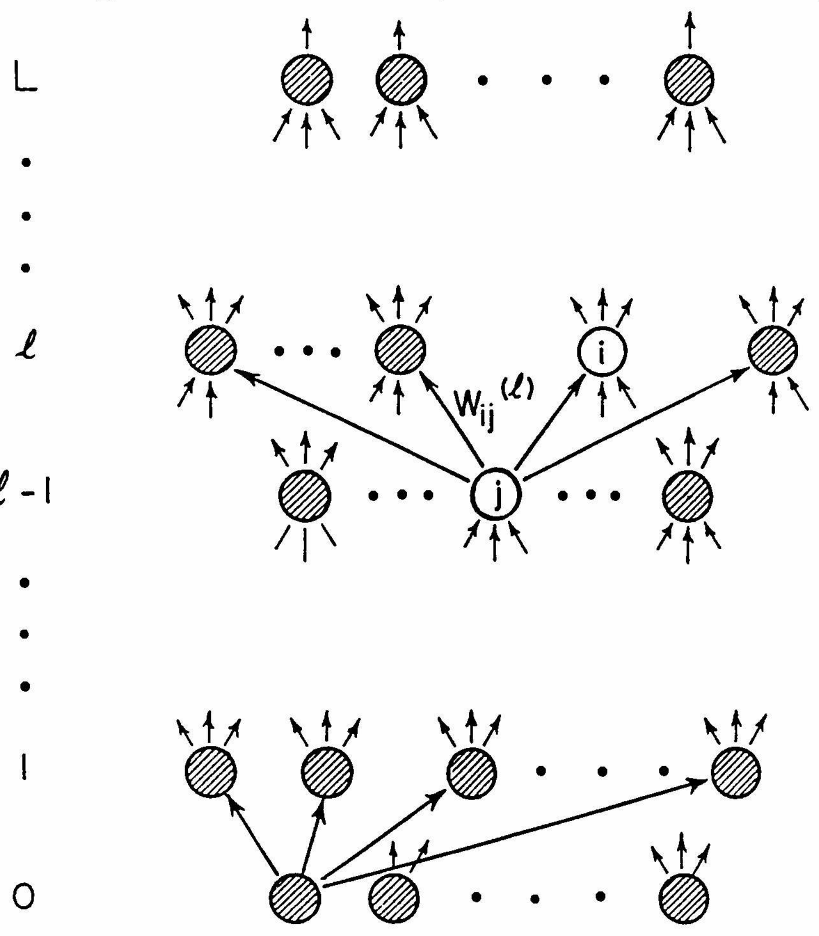

Figure 1: A layered feed-forward network with $L$ levels of processing.

deterministic parallel dynamics, as shown in Fig. 1. The network consists of $L+1$ layers providing $L$ levels of processing [Solla 89]. The first layer at $\ell = 0$ is the input field, and contains $N_0$ units. Subsequent layers are labeled $1 \leq \ell \leq L$; the $\ell$-th layer contains $N_\ell$ units. The $\ell$-th level of processing corresponds to determining the state of the units in layer $\ell$ according to the following deterministic and parallel rule:

$$\begin{aligned} U_i^{(\ell)} &= \sum_{j=1}^{N_{\ell-1}} W_{ij}^{(\ell)} V_j^{(\ell-1)} + W_i^{(\ell)}, \\ V_i^{(\ell)} &= g(U_i^{(\ell)}). \end{aligned} \tag{2.2}$$

The state $V_i^{(\ell)}$ of unit $i$ in layer $\ell$ is thus determined by the states $\{V_j^{(\ell-1)}\}$, $1 \leq j \leq N_{\ell-1}$, of units in the preceding layer.

The first layer receives input from the external world: a pattern is presented to the network by fixing the values of the state variables $\{V_i^{(0)}\}$, $1 \leq i \leq N_0$. The states of subsequent layers are determined consecutively according to Eq. (2.2). The state of the last layer at $\ell = L$ is the output. Given $\vec{x} = \vec{V}^{(0)}$ the network produces an output $\vec{y} = \vec{V}^{(L)}$, thus implementing an input-output map $\vec{y} = f(\vec{x})$.

It is precisely such ability to implement input-output maps that has triggered much recent interest in layered architectures [Lippmann 87]. Pattern recognition tasks require the identification of inputs as belonging to one out of a set of possible categories. Layered neural networks provide a simple representation for such classification tasks: the $j$-th output unit is devoted to a specific logical preposition $A_j$ about the input (the input belongs to category $j$, or possesses attribute $j$, or evokes memory $j$). The activity $y_j^\alpha$ of the $j$-th output unit under presentation of input $\vec{x}^\alpha$ is interpreted as the conditional probability that $A_j$ is true for $\vec{x}^\alpha$:

$$y_j^\alpha = \text{Prob}\{A_j = T|\vec{x}^\alpha\}. \tag{2.3}$$

This probabilistic interpretation of the activity of the output units provides a tool for the implementation of tasks such as categorization, pattern recognition, diagnosis, assignment of meaning, and associative memory.

A layered architecture is specified by the number $\{N_\ell\}, 0 \leq \ell \leq L$ of units per layer. The parameters of the network are the couplings $\{W_{ij}^{(\ell)}\}$, $1 \leq i \leq N_\ell$, $1 \leq j \leq N_{\ell-1}$, $1 \leq \ell \leq L$, and the biases $\{W_i^{(\ell)}\}$, $1 \leq i \leq N_\ell$, $1 \leq \ell \leq L$, corresponding to a point $\vec{W}$ in a configuration space of dimension

$$D_{\vec{W}} = \sum_{\ell=1}^{L} N_\ell(1 + N_{\ell-1}). \tag{2.4}$$

The dimensionality $D_{\vec{W}}$ thus counts the number of independent parameters needed to specify the network. It follows from Eq. (2.4) that $D_{\vec{W}} \sim L < N_\ell^2 >$, where $L$ is the depth of the network and $< N_\ell >$ is a characteristic layer width. Thus $D_{\vec{W}}$ is typically a large number, and it scales as $D_{\vec{W}} \sim N^2/L$, where $N = \sum_{\ell=1}^{L} N_\ell$ is the total number of processing units.

The configuration space $\{\vec{W}\}$ describes the ensemble of all possible networks that can be constructed within the constraint of the specified architecture. Every point $\vec{W}$ in configuration space represents a specific network design and corresponds to the realization of a specific input-output map $f_{\vec{W}}$. A question that arises is that of characterizing the class of maps $\vec{y} = f(\vec{x})$ that can be implemented by a given network architecture. Such characterization requires a full exploration of configuration space.

Among the various functions that are implementable by the chosen architecture, there is a desired map $\tilde{f}$ to be realized. The process of learning refers to a guided exploration of configuration space so as to determine values of the couplings $\{W_{ij}^{(\ell)}\}$ and biases $\{W_i^{(\ell)}\}$ for which $f_{\vec{W}} = \tilde{f}$. Such search for configurations $\vec{W}$ which implement the map $\tilde{f}$ is guided by whatever information is available on the desired map. Supervised learning requires the availability of examples: input-output pairs $(\vec{x}, \vec{y})$ for which $\vec{y} = \tilde{f}(\vec{x})$.

## 3 Functional Capabilities

Every point $\vec{W}$ in configuration space selects a specific network design which implements the map $f_{\vec{W}}$. It is convenient to partition configuration space according to the functional capabilities of the network ensemble. Regions corresponding to the implementation of a specific map $f$ are selected by a masking function

$$\Theta_f(\vec{W}) = \begin{cases} 1 & \text{if } f_{\vec{W}} = f \\ 0 & \text{if } f_{\vec{W}} \neq f \end{cases}$$

Given a prior density $\rho_0(\vec{W})$ which constrains the effective volume of configuration space to $\int d\vec{W} \rho_0(\vec{W}) = 1$, the fractional volume occupied by configurations which implement $f$ is given by

$$P_0(f) = \int d\vec{W} \rho_0(\vec{W}) \Theta_f(\vec{W}). \tag{3.1}$$

The functional capabilities of the network ensemble described by the density $\rho_0(\vec{W})$ are quantitatively specified by the probability distribution $P_0(f)$ on the space of functions [Solla 89]. The class of functions implementable by the chosen architecture is

$$\mathcal{F}_0 = \{f | P_0(f) \neq 0\}. \tag{3.2}$$

The realizability of the desired map $\tilde{f}$ corresponds to the requirement $P_0(\tilde{f}) \neq 0$.

It is useful to consider the entropy [Denker 87]

$$S_0 = -\sum_{\{f\}} P_0(f) \ln P_0(f) \tag{3.3}$$

of the prior distribution. Only functions $f \in \mathcal{F}_0$ contribute to $S_0$. The optimal case of an ensemble devoted to the unique implementation of the desired map $\tilde{f}$ corresponds to $P_0(\tilde{f}) = 1$ and $P_0(f) = 0$ for all $f \neq \tilde{f}$, and results in $S_0 = 0$. But in a typical case the set $\mathcal{F}_0$ of realizable functions contains many maps $f$ besides $\tilde{f}$, and $S_0 > 0$.

Learning refers to a systematic modification of the network ensemble towards specificity in the implementation of the desired map $\tilde{f}$. The starting point is an ensemble defined by the density $\rho_0(\vec{W})$ and characterized by the prior entropy $S_0$; learning results in a systematic entropy reduction towards the goal $S = 0$.

The prior entropy $S_0$ is an intrinsic property of the chosen network architecture: the ensemble of possible network configurations described by the density $\rho_0(\vec{W})$ fully determines the distribution $P_0(f)$ and its associated entropy. This intrinsic entropy is bounded by

$$S_0 \leq \ln n_0, \tag{3.4}$$

where $n_0$ is the number of implementable functions which define the class $\mathcal{F}_0$, Eq. (3.2). The upper bound is attained when all realizable functions are equally likely, and corresponds to a uniform distribution, $P_0(f) = 1/n_0$ for all $f \in \mathcal{F}_0$. This bound is rarely achieved; the distribution $P_0(f)$ is typically nonuniform since the prior ensemble is biased towards the implementation of a subset of functions within $\mathcal{F}_0$.

The characterization of the functional capabilities of the network ensemble through the probability distribution $P_0(f)$ provides two criteria for the appropriate choice of network architecture. The first one is the *realizability* of the desired map, $P_0(\tilde{f}) \neq 0$. The second one is *specificity*: $P_0(\tilde{f}) >> P_0(f)$ for most other $f \in \mathcal{F}_0$, which results in a small intrinsic entropy $S_0$ and facilitates the achievement of $S = 0$ through learning.

## 4 Supervised Learning

Supervised learning requires examples of the desired map $\tilde{f}$. The training set contains $m$ input-output pairs $(\vec{x}^\alpha, \vec{y}^\alpha)$, $1 \leq \alpha \leq m$, for which $\vec{y}^\alpha = \tilde{f}(\vec{x}^\alpha)$. Learning the training set is posed as an optimization problem [Solla 88] by introducing a measure of quality: to which extent does the map $f_{\vec{W}}$ realized by network $\vec{W}$ coincide with the desired map $\tilde{f}$? Given the input $\vec{x}^\alpha$, network $\vec{W}$ associates to it the output $f_{\vec{W}}(\vec{x}^\alpha)$. The distance

$$\varepsilon^\alpha(\vec{W}) = d(\vec{y}^\alpha, f_{\vec{W}}(\vec{x}^\alpha)) \tag{4.1}$$

between the target $\vec{y}^\alpha$ and the actual output $f_{\vec{W}}(\vec{x}^\alpha)$ measures the error made by the network on the $\alpha$-th example.

The total error

$$E_m(\vec{W}) = \sum_{\alpha=1}^{m} \varepsilon^\alpha(\vec{W}) \tag{4.2}$$

measures the dissimilarity between $f_{\vec{W}}$ and $\tilde{f}$ on the restricted domain $\{\vec{x}^\alpha\}$, $1 \leq \alpha \leq m$, of input space. This measure is not just a counting of the number of errors. It is based on a distance in output space, and thus carries information on the magnitude of the error incurred on, if any. A configuration satisfying

$$E_m(\vec{W}) = 0 \tag{4.3}$$

is that of a network which produces the correct output $\vec{y}^\alpha$ for every input $\vec{x}^\alpha$ in the training set.

Learning refers to the search for global minima of $E_m(\vec{W})$. A variety of techniques can be applied to this optimization problem. Gradient descent results in the back-propagation algorithm [Rumelhart 86]. Although the task is rendered difficult by the high dimensionality $D_{\vec{W}}$ of configuration space (Eq. (2.4)), and the roughness of the surface defined by $E_m(\vec{W})$ [Solla 88], it is somewhat simplified by knowing that solutions correspond to $E_m(\vec{W}) = 0$.

The search for network configurations $\vec{W}$ which satisfy the $E_m(\vec{W}) = 0$ condition results in a modification of the network ensemble. The untrained networks are described by the probability $\rho_0(\vec{W})$ of finding network $\vec{W}$ within the ensemble. The probability $\rho_m(\vec{W})$ of finding network $\vec{W}$ within the ensemble of trained networks includes a survival probability, $\exp\{-\beta E_m(\vec{W})\}$. After appropriate normalization,

$$\rho_m(\vec{W}) = \frac{1}{Z_m}\rho_0(\vec{W})e^{-\beta E_m(\vec{W})}, \tag{4.4}$$

with

$$Z_m = \int d\vec{W}\rho_0(\vec{W})e^{-\beta E_m(\vec{W})}. \tag{4.5}$$

The parameter $\beta$ controls the error sensitivity. The limit $\beta \to \infty$ corresponds to error-free learning, since network $\vec{W}$ can only survive the training if it produces the correct output for every input in the training set: as $\beta \to \infty$, $E_m(\vec{W}) \neq 0$ results in $\rho_m(\vec{W}) = 0$. For finite $\beta$, $\rho_0(\vec{W}) \neq 0$ guarantees $\rho_m(\vec{W}) \neq 0$. All networks present in the prior ensemble are still represented in the trained ensemble, but with a probability that is reduced exponentially with the error of the network on the training set.

The normalization integral $Z_m$ of Eq. (4.5) measures the effective volume of configuration space occupied by the ensemble of trained networks, and guarantees $\int d\vec{W}\rho_m(\vec{W}) = 1$. Since $E_m(\vec{W}) \geq E_{m-1}(\vec{W})$ for all networks $\vec{W}$, then $Z_m \leq Z_{m-1}$. Learning thus results in a monotonic contraction of the effective volume in configuration space as the size $m$ of the training set is increased.

The ensemble of trained networks described by the density $\rho_m(\vec{W})$ of Eq. (4.4) is a Gibbs canonical ensemble [Tishby 89], with the error sensitivity parameter $\beta$ playing the role of an inverse temperature. The properties of the ensemble can be investigated using standard statistical physics techniques. The normalization integral $Z_m$ of Eq. (4.5) is the partition function, from which the average learning error, the entropy, and the prediction error can be obtained through appropriate derivatives [Tishby 89,Levin 89].

## 5 Generalization Ability

Although supervised learning is formulated as a search for networks which minimize the learning error $E_m(\vec{W})$ on the training set, the actual goal is to produce networks with good generalization ability, i. e. a high probability of producing the correct output for inputs not in the training set. The generalization ability $G_m$ is defined as the probability that a trained network, chosen according to the probability density $\rho_m(\vec{W})$, will produce the correct output for an arbitrary test input $\vec{x}$, distinct from the $m$ training inputs. The problem is thus one of extending the domain of a function: given the partial information of knowing $\bar{f}$ only on the points $\{\vec{x}^\alpha\}$, $1 \leq \alpha \leq m$ of the training set, is it possible to obtain a network which realizes the target function $\bar{f}$, and thus produces $\bar{f}(\vec{x})$ for every point in input space $\{\vec{x}\}$?

A learning session starts with a network chosen from the prior ensemble according to the probability density $\rho_0(\vec{W})$, and produces through the modification of the couplings and biases a network belonging to the trained ensemble, as described by the probability density $\rho_m(\vec{W})$. A meaningful measure of the emerging generalization ability can only be obtained by considering the statistical properties of the ensemble of trained networks.

It is straightforward to extend the analysis of functional capabilities of Sec. 3 to the ensemble of trained networks described by the probability density $\rho_m(\vec{W})$. The probability of finding networks which implement a specific map $f$ is given by

$$P_m(f) = \int d\vec{W} \rho_m(\vec{W}) \Theta_f(\vec{W}). \tag{5.1}$$

All networks $\vec{W}$ for which $\Theta_f(\vec{W}) = 1$ share a common value of the learning error $E_m(\vec{W})$. Then

$$\frac{P_m(f)}{P_0(f)} = \frac{\rho_m(\vec{W})}{\rho_0(\vec{W})} \tag{5.2}$$

for all $\vec{W}$ such that $\Theta_f(\vec{W}) = 1$. The equality (5.2) proves that the partition of configuration space according to its functional capabilities is *sufficient* in the statistical sense [Levin 89], in that it implies no loss of information.

It follows from Eqs. (4.4) and (5.2) that

$$\frac{P_m(f)}{P_0(f)} = \frac{1}{Z_m} e^{-\beta E_m(f)}, \tag{5.3}$$

with $Z_m \leq 1$ (since $Z_m \leq Z_0$, and $Z_0 = 1$), and $E_m(f)$ standing for the common value of $E_m(\vec{W})$ for all networks $\vec{W}$ for which $\Theta_f(\vec{W}) = 1$. The target function is emphasized, since $E_m(\tilde{f}) = 0$, and

$$\frac{P_m(\tilde{f})}{P_0(\tilde{f})} = \frac{1}{Z_m} \geq 1. \tag{5.4}$$

The same amplification factor $(Z_m)^{-1}$ affects all functions $f$ sufficiently similar to the target function $\tilde{f}$ to agree with it on all $m$ training inputs. Functions for which $E_m(f) > 0$ differ from $\tilde{f}$ even on the restricted set of inputs $\{\vec{x}^\alpha\}$, $1 \leq \alpha \leq m$, and are exponentially deemphasized according to Eq. (5.3). Such sharpening of the probability distribution on the space of functions results in a decrease of the entropy

$$S_m = -\sum_{\{f\}} P_m(f) \ln P_m(f) \tag{5.5}$$

with respect to the intrinsic entropy $S_0$. The monotonic decrease of the entropy $S_m$ with increasing training set size $m$ reflects an increasing specificity. The network ensemble becomes narrowly focused on the implementation of the target function $\tilde{f}$ and a class of functions increasingly similar to $\tilde{f}$.

Consider the class of functions

$$\mathcal{F}_m = \{f | P_m(f) \neq 0\} \tag{5.6}$$

implementable by the ensemble of trained networks. Only functions $f \in \mathcal{F}_m$ contribute to $S_m$. It follows from Eq. (5.4) that $P_0(\tilde{f}) \neq 0$ guarantees $P_m(\tilde{f}) \neq 0$. But $P_m(f) \neq 0$ for all

$f \in \mathcal{F}_m$, not only $\tilde{f}$. It is precisely such residual diversity in the functional capabilities of the trained ensemble which conspires against the achievement of perfect generalization ability. Training does not necessarily result in a network $\vec{W}$ which implements $\tilde{f}$, but in a network $\vec{W}$ which implements any function $f \in \mathcal{F}_m$ with probability $P_m(f)$. It is only through the systematic decrease of the residual entropy $S_m$ that generalization ability emerges.

The efficiency of the $m$-th example in reducing the ensemble entropy,

$$\eta_m = S_{m-1} - S_m, \tag{5.7}$$

measures the information content of the example. The average efficiency of a training set of size $m$,

$$\bar{\eta} = \frac{1}{m} \sum_{i=1}^{m} \eta_i, \tag{5.8}$$

is given by the total entropy decrease,

$$\bar{\eta} = \frac{S_0 - S_m}{m}. \tag{5.9}$$

The goal $S_m = 0$ thus requires a training set of size

$$m^\star = \frac{S_0}{\bar{\eta}}. \tag{5.10}$$

The number of examples $m^\star$ needed to restrict the functionality of the ensemble to the implementation of $\tilde{f}$ is proportional to $S_0$, the intrinsic entropy of the chosen architecture. The bound of Eq. (3.4) implies

$$m^\star \leq \frac{1}{\bar{\eta}} \ln n_0. \tag{5.11}$$

Meaningful generalization ability can be extracted from a reasonably small training set only through constraints in the intrinsic functional capabilities of the chosen architecture. Consider the implementation of Boolean functions from $N_0$ inputs into $N_L = 1$ output. There are $2^{2^{N_0}}$ such functions. If the chosen architecture is so general as to be able to implement all of them, then $\ln n_0 = 2^{N_0}$, and the desired map $\tilde{f}$ cannot be extracted until the training set size reaches $m^\star \sim 2^{N_0}$. Such regime requiring a number of examples comparable to the total number of points in input space is of little interest.

The number $n_0$ of functions implementable by the chosen architecture can be estimated by considering that it cannot exceed the total number of distinct networks within the prior ensemble. If the couplings and biases are specified with $b$ bits of precision, then $\ln n_0 \leq b D_{\vec{W}}$, where $D_{\vec{W}}$ is the dimensionality of configuration space. Since $D_{\vec{W}} \sim N^2/L$, where $N$ is the total number of neurons and $L$ is the depth of the network, the number of examples needed to extract the desired map $\tilde{f}$ is bounded by $m^\star \sim bN^2/L$, a number which can be kept reasonably small by a judicious choice of network architecture.

## 6 The Contiguity Problem

The use of layered neural networks for classification, categorization, or diagnosis, discussed in Sec. 2, relies on the interpretation of the activity $y_j^\alpha$ of the $j$-th output unit under presentation of the $\alpha$-th input (Eq. (2.3)) as the probability that the input belongs to category $j$ or possesses attribute $j$.

A specific classification task, the contiguity problem [Denker 87,Solla 88,Solla 89] is used here to illustrate issues of learning and generalization. The contiguity problem is a classification of binary input patterns $\vec{x} = (x_1, ..., x_{N_0})$, $x_i = 0, 1$ for all $1 \leq i \leq N_0$, into classes according to the number $k$ of blocks of +1's in the pattern. For example, for $N_0 = 10$, $\vec{x} = (0110011100)$ corresponds to $k = 2$, while $\vec{x} = (0101101111)$ corresponds to $k = 3$. This classification leads to $(1 + k_{max})$ categories corresponding to $0 \leq k \leq k_{max}$, with $k_{max} = \lceil \frac{N_0}{2} \rceil$. (For any real number $u$ in the interval $(n-1) < u \leq n$, $\lceil u \rceil = n$ is the upper integer part of $u$).

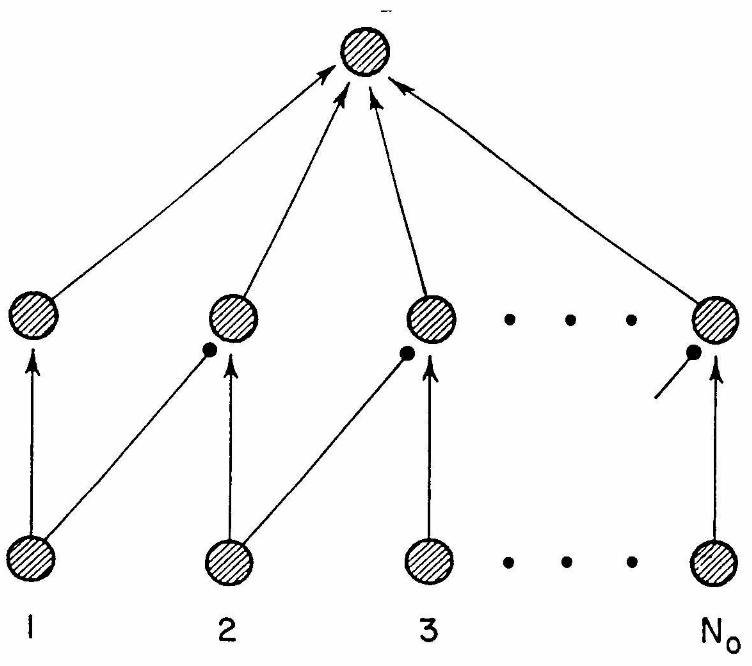

Figure 2: A network solution to the contiguity problem with $L = 2$. All couplings $\{W_{ij}\}$ have absolute value of unity. Excitatory ($W_{ij} = +1$) and inhibitory ($W_{ij} = -1$) couplings are indicated by $\rightarrow$ and $-\!\bullet$, respectively. Intermediate units are biased by $W_i^{(1)} = -0.5$; the output unit is biased by $W_i^{(2)} = -(k_0 + 0.5)$.

There are

$$\mathcal{N}(k) = \begin{pmatrix} N_0 + 1 \\ 2k \end{pmatrix} \tag{6.1}$$

input patterns in the $k$-th category. A simpler classification task investigated here is the dichotomy into two classes corresponding to $k \leq k_0$ and $k > k_0$. This problem can be solved [Solla 89] by an $L = 2$ layered network with $N_0$ input units, $N_1 = N_0$ intermediate units, and $N_2 = 1$ output unit. The architecture is shown in Fig. 2. The first level of processing detects subsequent 01 pairs corresponding to the left edge of a block, and the second level of processing counts the number of such edges to determine the value of $k$.

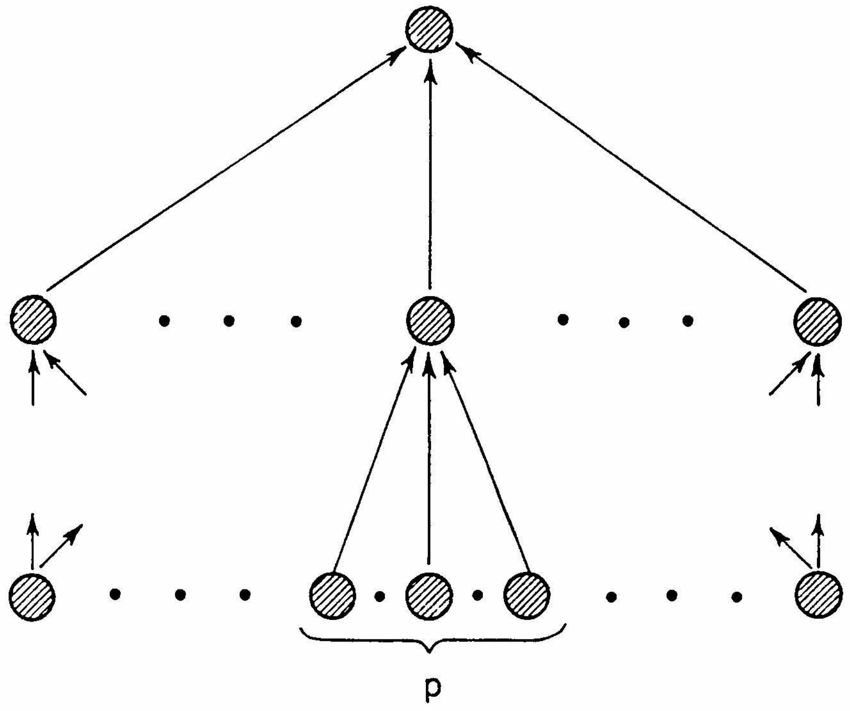

Figure 3: Network architecture for learning contiguity, with $L = 2$, $N_0 = N_1$, $N_2 = 1$, and a receptive field of size $p$.

Knowing that such solution exists, we choose for our learning experiments the network architecture shown in Fig. 3, with $N_0 = N_1$ and $N_2 = 1$. The network is not fully connected: each unit in the $\ell = 1$ layer receives input from only the $p$ subsequent units just below it in the $\ell = 0$ layer. The parameter $p$ can be interpreted as the width of a *receptive field.* Note that $p \leq N_0$, and that for $p < N_0$ the network is not fully connected.

The results reported here are for $N_0 = 10$, $k_0 = 2$, and $2 \leq p \leq 10$. The total domain of $2^{N_0} = 1024$ input patterns is restricted to the union of $\mathcal{N}(2) = 330$ and $\mathcal{N}(3) = 462$ corresponding to $k = 2$ and $k = 3$ respectively. Out of these $\mathcal{N}(2) + \mathcal{N}(3) = 792$ input patterns, a training set of $m = 100$ patterns is randomly selected, with $m(2) = m(3) = 50$ examples of each category.

The search for appropriate network configurations $\vec{W}$ is performed using the back-propagation algorithm [Rumelhart 86] and the quadratic error function [Solla 88]

$$E_m(\vec{W}) = \sum_{\alpha=1}^{m}(y^\alpha - f_{\vec{W}}(\vec{x}^\alpha))^2. \tag{6.2}$$

The starting point $\vec{W}_o$ for the gradient descent algorithm is chosen at random from a normal distribution. The step size for the downhill search is kept constant during each run.

After each learning iteration consisting of a full presentation of the training set ($\Delta t = 1$), the network is checked for learning and generalization abilities by separately counting the fraction of correct classifications for patterns within ($\%L$) and not included ($\%G$) in the training set, respectively. The classification of the $\alpha$-th input pattern is considered correct if $|y^\alpha - f_{\vec{W}}(\vec{x}^\alpha)| \leq \Delta$, with $\Delta = 0.1$.

Both the learning $\%L$ and generalization $\%G$ abilities of the network are monitored as a function of time $t$. The learning process is terminated after $\tau$ presentations of the training set. The stopping criterion is that the network performance on the training set, as measured by $E_m$ defined in Eq. (6.2), satisfies $E_m \leq 0.005$. For a training set of size $m = 100$ and the network architecture of Fig. 3, learning is always successful for $p \geq 3$.

| $p$ | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 |
|---|---|---|---|---|---|---|---|---|
| $\%G$ | 55 | 59 | 61 | 63 | 68 | 73 | 90 | 95 |

Table I. Results for learning contiguity with varying receptive fields $p$, averaged over successful training runs ($\%L = 100$).

Results summarized in Table I are averages over several runs with different starting points $\vec{W}_o$ for each value of $p$. All simulations achieved $\%L = 100$. Note that the generalization ability of the resulting networks increases monotonically with decreasing $p$. For $p = 3$, $\%G = 100$ is sometimes obtained. For $p = 2$, the gradient descent algorithm often gets trapped in local minima with large $E_m$, and $\%L = 100$ cannot be achieved. But when a $p = 2$ network is successfully trained to $\%L = 100$, it also exhibits full generalization ability $\%G = 100$. It is of interest to examine the network configurations which exhibit good generalization ability. The $p = 2$ and 3 networks with $\%G = 100$ correspond to organization into edge detectors, as shown in Fig. 2.

The monotonic improvement of the generalization ability with decreasing receptive field $p$, illustrated for the contiguity problem in Table I, is a consequence of the reduction of the dimensionality $D_{\vec{W}}$ of configuration space and the concurrent reduction in the number $n_0$ of implementable functions.

The effect is quite general, and is described as follows. The dimensionality $D_{\vec{W}}$ of Eq. (2.4) can be written as

$$D_{\vec{W}} = \sum_{\ell=1}^{L} d_\ell, \tag{6.3}$$

where $d_\ell$ counts the number of couplings and biases incoming to the units in layer $\ell$. For a fully connected network as shown in Fig. 1, each of the $N_\ell$ units in layer $\ell$ receives a bias, and input from all $N_{\ell-1}$ units in layer $(\ell - 1)$. Thus

$$d_\ell = N_\ell(1 + N_{\ell-1}), \tag{6.4}$$

and Eq. (2.4) is recovered. For a network with a receptive field organization, each of the $N_\ell$ units in layer $\ell$ receives a bias, and input from the $p$ units in layer $(\ell - 1)$ within its receptive field. Thus

$$d_\ell = N_\ell(1 + p), \tag{6.5}$$

which results in a dimensionality reduction for $p < N_\ell$.

For the contiguity problem, the network architecture of Fig. 3 corresponds to $d_1 = N_0(1+p)$, $d_2 = (1+N_0)$, and $D_{\vec{W}} = N_0(p+2)+1$. Full connectivity is recovered for $p = N_0$, and results in $D_{\vec{W}} = (N_0+1)^2$. Note the difference between $D_{\vec{W}} \sim N_0^2$ for the fully connected network, and the linear $D_{\vec{W}} \sim pN_0$ for the constrained network. Dimensionality reduction limits the functional capabilities of the chosen architecture, and results in better generalization ability even for a small number of training examples. Specificity is lost by increasing $p$; the resulting increase in the number $n_0$ of implementable functions requires an increase in the number of training examples if good generalization ability is to be achieved.

Specificity due to limited receptive fields stimulates the emergence of feature extractors such as edge detectors. This method for dimensionality reduction is thus particularly well suited for networks intended for pattern recognition tasks. Further reduction in the number of independent parameters needed to specify the network can be achieved through weight-sharing techniques [Rumelhart 86,le Cun 89]. The basic idea is to incorporate symmetries in the form of equality constraints which reduce $D_{\vec{W}}$ without modifying the actual size of the network. In the case of pattern recognition, these additional constraints can be used to generate shift-invariant feature extractors.

As a simple illustration of the weight-sharing techniques, consider once more the network architecture of Fig. 3. For $p = 2$, $D_{\vec{W}} = 4N_0 + 1$ independent parameters are needed to specify the network. Since the solution of Fig. 2 is based on a unique type of feature extractor, namely edge detectors, further constraints can be introduced by requiring

$$W^{(1)}_{i,i-1} = W_L \tag{6.6a}$$

and

$$W^{(1)}_{i,i} = W_R \tag{6.6b}$$

for all $1 \le i \le N_0$. Such constraint guarantees the equality of all weights coming from left and right onto all units in the $\ell = 1$ layer, and produces a shift-invariant feature extractor (Except for the $i = 1$ unit which receives input only from the unit below it, as shown in Fig. 2). The dimensionality of configuration space is thus reduced to $D_{\vec{W}} = 2N_0 + 3$, without any decrease in the size of the network.

It might be argued that the $p = 2$ network is already quite constrained, and that to impose the additional weight-sharing equalities of Eq. (6.6) amounts to giving too much of an architectural hint about the task to be performed. Such criticism can only be overcome by demonstrating the power of the techniques when applied to larger problems, as the one considered in the following Section.

## 7 The Digit Recognition Problem

The use of limited receptive fields in combination with weight-sharing techniques stimulates the emergence of shift-invariant feature extractors and generates networks with good generalization ability for pattern recognition tasks. A specific classification task based on the recognition of handwritten digits [le Cun 89] is used here to illustrate such strategy.

The problem is the classification of numerals. Inputs consist of 16x16 images of binary pixels, handwritten by a single person using a mouse. Of the 48 examples generated for each class, 32 are randomly selected to form a training set of size $m = 320$. The remaining 160 examples are used to evaluate the generalization ability of the trained networks. Some of the training examples are shown in Fig. 4.

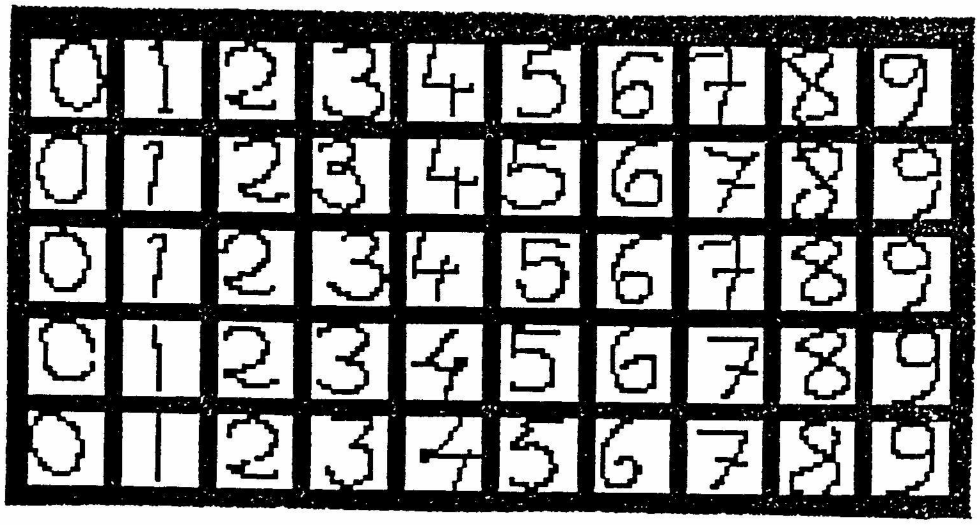

Figure 4: Some examples of input patterns.

Learning experiments have been performed on a variety of network architectures. In all cases the input field contains $N_0$ =16x16=256 units, and $N_L$ = 10 output units are used to represent the 10 distinct categories. The search for appropriate network configurations $\vec{W}$ is performed using the back-propagation algorithm. The starting point $\vec{W}_o$ for the gradient descent algorithm is chosen at random from a uniform distribution, scaled for each component according to the size of the receptive field it contributes to. The step size for the downhill search is adjusted according to the curvature of the error function [le Cun 89]. The learning process is terminated after $\tau = 30$ presentations of the training set. The output is considered correct if the most active unit signals the correct category.

Three different network architectures investigated for this problem are described and compared below.

## 7.1 Network # 1

This is an $L = 2$ fully connected network, as shown in Fig. 5. The intermediate $\ell = 1$ layer contains $N_1 = 12$ hidden units. Since the network is fully connected, $d_1 = N_1(1 + N_0)$=12x257=3084, and $d_2 = N_2(1+N_1)$=10x13=130, resulting in $D_{\vec{W}} = 3214$ independent parameters. This network achieves a generalization ability of $\%G = 87$.

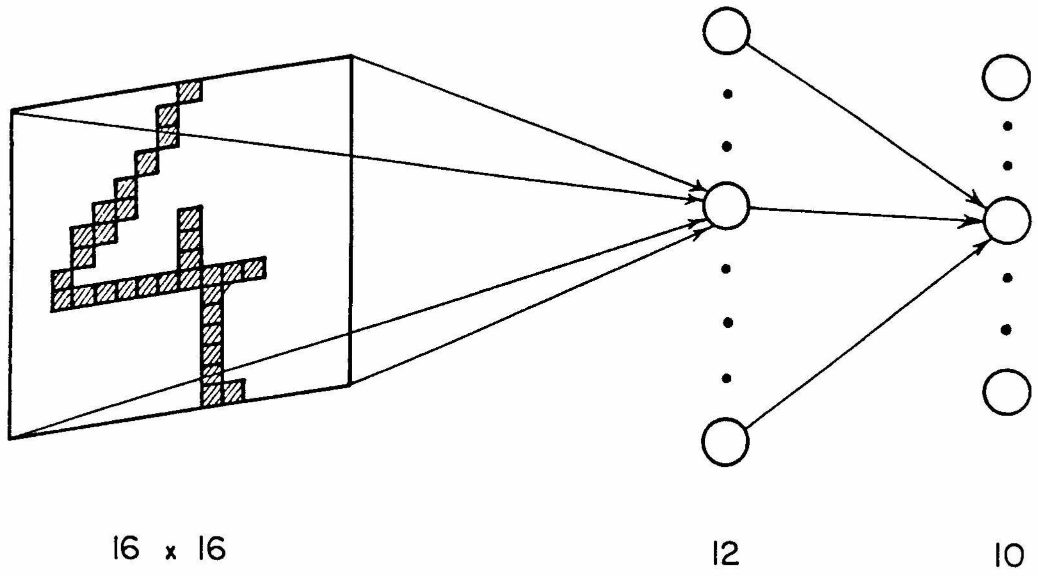

Figure 5: Network # 1, an $L = 2$ network with $N_0$ =16x16=256, $N_1 = 12$, and $N_2 = 10$.

The generalization ability exhibits large fluctuations for networks trained from different starting points $\vec{W}_o$. A large variance with respect to the mean $\%G$ indicates that the available data is not sufficient to specify the configuration of the chosen architecture. The trained ensemble is too diverse, and the many configurations $\vec{W}$ which are compatible with the training set vary widely in their generalization ability. This network architecture has too many independent parameters.

## 7.2 Network # 2

This is an $L = 3$ network with limited receptive fields, as shown in Fig. 6. The $\ell = 1$ layer consists of a square array of $N_1$=8x8=64 units, each one of them receiving input from a receptive field of size $p$=3x3=9, resulting in $d_1 = N_1(1+p)$=64x10=640. The $\ell = 2$ layer consists of a smaller square array of $N_2$=4x4=16 units, each of them receiving input from a receptive field of size $p$=5x5=25, resulting in $d_2 = N_2(1+p)$=16x26=416. The $\ell = 2$ layer is fully connected to the output at $\ell = 3$, and $d_3 = N_3(1+N_2)$=10x17=170. The total number of independent parameters is $D_{\vec{W}} = 1226$.

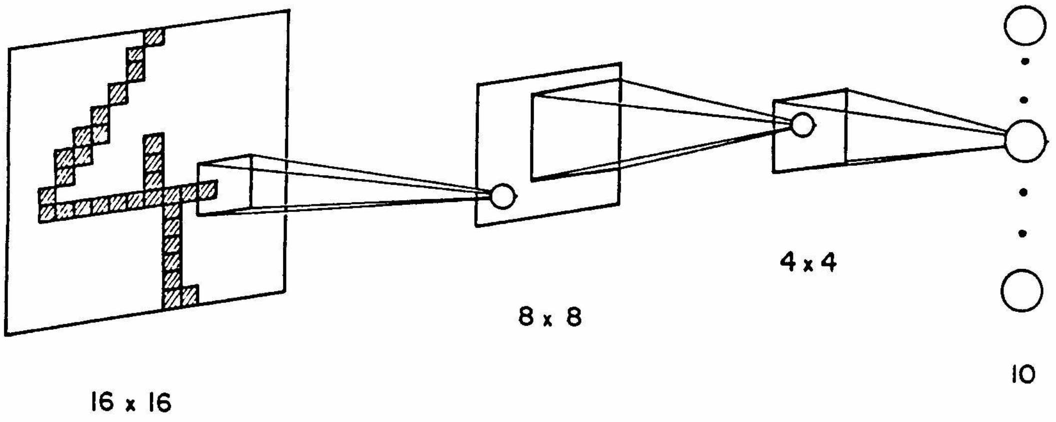

Figure 6: Network # 2, an $L = 3$ network with $N_0$=16x16=256, $N_1$=8x8=64, $N_2$=4x4=16, and $N_3 = 10$.

This network achieves a generalization ability of $\%G = 89.5$. The performance is better than that of the preceding network: a better mean value for the generalization ability together with a smaller variance indicate that the ensemble of trained networks is more narrowly focused in its functional capabilities. Since both networks have been trained with the same training set of size $m = 320$, a smaller residual entropy $S_m$ reflects a smaller intrinsic entropy $S_0$ for the constrained network.

## 7.3 Network # 3

This is an $L = 3$ network with limited receptive fields and shared weights, as shown in Fig. 7. The $\ell = 1$ layer consists of *two* square arrays of 8x8=64 units each, with a total of $N_1 = 128$ units. Each one of these arrays is a feature map: all units within a given array share the same weights, and therefore detect the same feature at different locations of the input field. As in the preceding network, every unit in the $\ell = 1$ layer receives input from a receptive field of size $p$=3x3=9. But in contrast to network # 2, all units within a feature map share the same set of 9 couplings, although they can have different biases. Thus 64 biases and 9 coupling suffice to specify all connections into each of the feature maps, and $d_1$=2x(64+9)= 146.

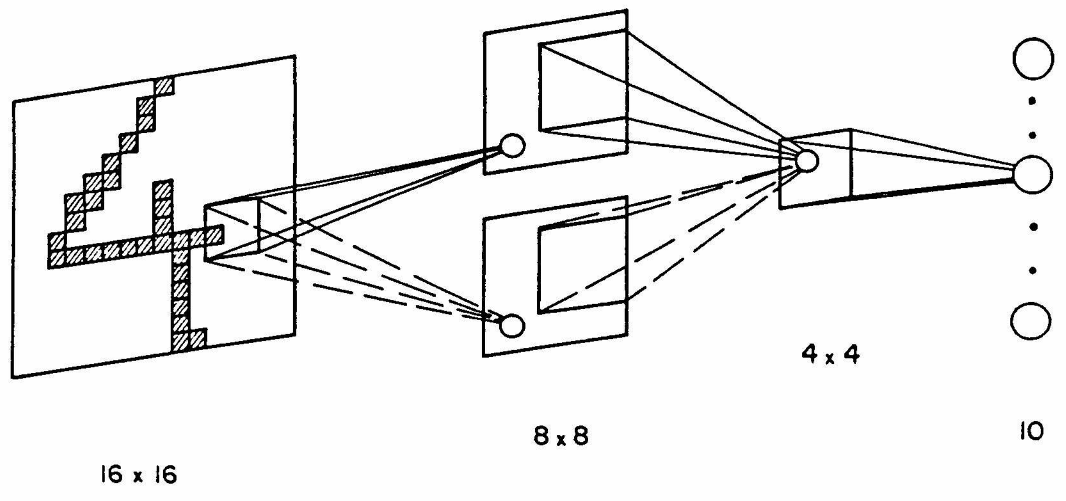

Figure 7: Network #3, an $L = 3$ network with $N_0$=16x16=256, $N_1$=2x8x8=128, $N_2$=4x4=16, and $N_3$ = 10.

The $\ell = 2$ layer consists of a smaller square array of $N_2$= 4x4=16 units. Each receives input from *two* receptive fields of size $p$=5x5=25, located in equivalent positions in both feature maps. There is no weight-sharing, and $d_2 = N_2(1 + 2p)$=16x51=816. The $\ell = 2$ layer is fully connected to the output at $\ell = 3$, and $d_3 = N_3(1 + N_2)$=10x17=170. The total number of independent parameters for this network is $D_{\vec{W}}$=1132.

This network achieves a generalization ability of $\%G = 94$. Shift-invariant feature extraction is a useful tool for the digit recognition task!

## 7.4 Comparative Remarks

The systematic improvement of the generalization ability achieved through the successive training of networks # 1, # 2, and # 3, illustrates the advantages of reducing the functional capabilities of the chosen architecture. A systematic reduction in the number $D_{\vec{W}}$ of free parameters needed to specify the network results in a decrease of the number $n_0$ of implementable functions and of the intrinsic entropy $S_0$. Such loss of generality should not undermine the network's ability to implement the desired function: the specific strategy adopted to achieve the dimensionality reduction should be tailored to the task.

The strategy is to incorporate prior knowledge about the desired task onto the choice of network architecture. Although specifying such knowledge might be difficult in a general case, it appears feasible in highly regular tasks such as pattern recognition. Solutions to this problem rely on extracting local features, and combining them into higher order features.

Fully connected networks such as network # 1 do not work very well for this problem. The use of limited receptive fields in networks # 2 and # 3 forces the hidden units to combine only local sources of information, and stimulates the emergence of feature extractors.

For both networks # 2 and # 3, units in layer $\ell = 1$ receive information from 9 units in the $\ell = 0$ input layer, arranged in a 3x3 square. Contiguous receptive fields are displaced by two units in the $\ell = 0$ plane, so that the input fields of adjacent units in the $\ell = 1$ layer share either a row or a column. The contraction from 16x16 to 8x8 thus achieved in going from $\ell = 0$ to $\ell = 1$ results in a loss of spatial resolution. Information on the location of a detected feature is kept at a coarser level: it is not the precise location of a feature that is relevant to the classification process, but its approximate location in relation to the approximate location of other features also present in the image.

Units in the $\ell = 1$ layer of network # 2 develop their couplings independently during the training process. Different units thus specialize in the detection of different features within their specific receptive fields. Since distinctive features of a digit can appear at several different locations, it is useful to develop feature extractors capable of detecting a feature anywhere in the input field. It is precisely such shift-invariant feature extractors which emerge as a consequence of the weight-sharing constraint imposed on the $\ell = 1$ layer of network # 3. Each one of the $\ell = 1$ planes is devoted to the detection of a unique feature, everywhere in the input field.

Units in the $\ell = 2$ layer of networks # 2 and # 3 receive information from receptive fields containing 25 units in layer $\ell = 1$, arranged in a 5x5 square. Contiguous receptive fields are displaced by one unit in the $\ell = 1$ plane. The further reduction from 8x8 to 4x4 units increases the compression of the feature map. In network # 3, each unit in the $\ell = 2$ layer receives information from *two* receptive fields, one from each $\ell = 1$ feature map. Such units capture correlations between the different features detected by the $\ell = 1$ layer. The considerably larger generalization ability of network # 3 is due to the emergence of such higher order feature extractors.

## 8 Conclusions

For a given training set of size $m$, the generalization ability $G_m$ measures the probability of obtaining a correct output when an input not in the training set is presented to a trained network.

The number $m^*$ of examples needed to obtain good generalization ability is controlled by the intrinsic entropy $S_0$ of the chosen architecture, and can be decreased by reducing the number $D_{\vec{W}}$ of independent parameters needed to specify the network configuration.

The reduction of the dimensionality $D_{\vec{W}}$ results in a decrease in the number $n_0$ of functions implementable by the chosen architecture. Such reduction must be planned with care, so as not to destroy the ability to implement the desired input-output map.

[Rumelhart 86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams: Learning Internal Representations by Error Propagation: Parallel Distributed Processing, MIT Press, Cambridge (1986), pp. 318-362.

[Solla 88] Sara A. Solla, Esther Levin, and Michael Fleisher: Accelerated Learning in Layered Neural Networks: Complex Systems 2 (1988), pp. 625-639.

[Solla 89] Sara A. Solla: Learning and Generalization in Layered Neural Networks: the Contiguity Problem: Neural Networks from Models to Applications, ed. by L. Personnaz and G. Dreyfus, IDSET, Paris (1989), pp. 168-177.

[Tishby 89] Naftali Tishby, Esther Levin, and Sara A. Solla: Consistent Inference of Probabilities in Layered Networks: Predictions and Generalization: Proceedings of the International Joint Conference on Neural Networks, IEEE, New York (1989), Vol. II, pp. 403-410.