Neocognitron and the Map Transformation Cascade

Ângelo Cardoso, Andreas Wichert

INESC-ID Lisboa and Instituto Superior Técnico, Technical University of Lisbon
Av. Prof. Dr. Aníbal Cavaco Silva, 2744-016 Porto Salvo, Portugal

Abstract

Based on our observations of the working principles of the archetypical hierarchical neural network, Neocognitron, we propose a simplified model which we call the Map Transformation Cascade. The least complex Map Transformation Cascade can be understood as a sequence of filters, which maps and transforms the input pattern into a space where patterns in the same class are close. The output of the filters is then passed to a simple classifier, which yields a classification for the input pattern. Instead of a specifically crafted learning algorithm, the Map Transformation Cascade separates two different learning needs: Information reduction, where a clustering algorithm is more suitable (e.g., K-Means) and classification, where a supervised classifier is more suitable (e.g., nearest neighbor method). The performance of the proposed model is analyzed in handwriting recognition. The Map Transformation Cascade achieved performance similar to that of Neocognitron.

Keywords: Clustering, Hierarchical Neural Networks, Neocognitron, Invariant Pattern Recognition

1. Introduction

One approach to pattern classification with artificial neural networks is to use a hand-designed feature extractor to gather relevant information (LeCun and Bengio, 1998). A feature-extractor reduces the amount of information needed to describe a pattern while trying not to compromise the accuracy of the description. The gathered information is then fed into a trainable clas-

Email addresses: `angelo.cardoso@ist.utl.pt` (Ângelo Cardoso), `andreas.wichert@ist.utl.pt` (Andreas Wichert)

sifier, often a fully-connected multilayer neural network. The bottleneck in this approach is how to construct the feature extractor. Another approach is to feed the raw input and rely on the learning algorithm to produce the first layers, the feature extractor. This approach has had some success, but when the input has a topology (e.g., temporal, spatial), it ignores this information, thus partially compromising its results, i.e., the order of the input variables could be switched to any other fixed arbitrary order and the results would be the same. This approach has no built-in invariance under shifts and distortions of the inputs. To overcome this problem, it is necessary to preprocess the inputs, by normalizing their sizes and centering them. But preprocessing does not solve the problem completely, and a huge number of training patterns are necessary for the network to exhibit stronger invariance properties. To classify even a small image, a fully-connected network has a huge number of connections, which makes it susceptible to overfitting and computationally expensive (LeCun and Bengio, 1998).

Hubel and Wiesel's discoveries have inspired several models for pattern recognition (Hubel, 1988). In these models, the neural units have a local view unlike the common fully-connected networks. Neocognitron (Fukushima, 1980) was the first of these models. It has local receptive fields, which means that there is a local view in the network elements. It gradually reduces the information from the input layer through the output layer. This is done by integrating local features into more global features in sequential transformations. Its purpose is to classify topological data by gradually reducing the information from the input layer through the output layer. Each of these transformations is composed of two different steps. The first one reduces the information by representing it with previously learned templates represented by S-cells (resembling simple cells). The second step blurs the information with C-cells (resembling complex cells), in order to allow positional shifts, giving the model some invariance under shifts and distortions. However, Neocognitron has been called "complex in structure and parametrization" (Lovell et al., 1993). It is one of the most complicated neural networks.

We propose a new less complex description of the pattern recognition capabilities of the Neocognitron through a new model, the Map Transformation Cascade. By this simplified model we try to understand the nature of the preformed classification and to answer the following question: What is a good choice of parameters?

The Map Transformation Cascade is a hierarchical neural network. The information is processed sequentially. Each layer only processes information after the previous layer is finished. The input is tiled with a squared mask, where each sub-pattern is replaced by a number indicating a corresponding class. By doing so, we get a representation of the pattern in the class space. The mask has the same behavior in all different positions, resembling the weight-sharing mechanism in Neocognitron and Convolutional Neural Networks. In the following step, the class representation of the pattern is transformed by losing the exact positional information. The corresponding representation of the pattern in the class space is tiled with a squared mask, eliminating the positional information of each class inside the mask. The layers of a Map Transformation Cascade can be seen as filters, since they have a clear and interpretable output, which is a modification of the input information. Several filters transform and map the input pattern into a space where patterns of the same class are close. The output of the filters is then passed to a simple classifier, which produces a classification for the input pattern. This classification is performed by a supervised classifier, which is in our case the k-Nearest Neighbor algorithm.

The performance of the proposed model is analyzed in handwritten character recognition on the ETL1 database [1]. Handwritten character recognition consists of determining the character represented by a given input image. Although the problem of printed character recognition is considered solved, handwritten character recognition is still an unresolved problem. Neural networks deliver state-of-the-art performance in this task (Sinha, 1999; Simard et al., 2003). Because of the wide scope of applications for handwritten text recognition, any slight improvement has significant economic impact, which makes this field very active.

Besides comparing networks with different characteristics, our main purpose is to understand the validity of the model in terms of performance, i.e., how well can the model perform with the right parameters on this data set, and how well that performance compares to Neocognitron's. To find the right choice of parameters to describe our architecture, we used a simple random search optimization algorithm. The Map Transformation Cascade and the Neocognitron achieved similar performance.

[1]ETL1 database, http://www.etl.go.jp/ etlcdb/index.htm.

2. Related work

In each layer of most artificial neural networks every unit receives input from all the units in the previous layer. This results in a global view in each of the units. Hierarchical networks work differently: Each unit only receives input from a localized subset of units from the previous layer, making its view local. Each unit only deals with a smaller localized part of the information. The global view is constructed as we ascend in the layers toward the output layer. This approach has two major benefits (Hecht-Nielsen, 1989): The units in each layer have simpler problems to consider (only part of the input), and these networks can work with many fewer units. The two most popular hierarchical neural networks are the Neocognitron and the Convolutional neural networks. Both networks share three key principles: Local receptive fields, shared weights and subsampling. Local receptive fields and shared weights reduce computational costs by decreasing the number of connections, neural units and parameters. Shared weights also improve generalization abilities by reducing the machine capacity (LeCun and Bengio, 1998). Subsampling improves shift and distortion invariance capabilities (LeCun and Bengio, 1998).

2.1. Neocognitron

Neocognitron (Fukushima, 1980, 1988) is a neural model mainly used for vision, which can perform unsupervised learning. It is an evolution of a previously proposed model, the Cognitron (Fukushima, 1975). Neocognitron has good generalization capabilities, as it is able to learn a pattern from only a few typical examples. For successful learning, it is not necessary to present all the deformed versions of patterns that might appear in the future (Fukushima, 1998). The Neocognitron has been successfully applied to handwritten text recognition (Fukushima, 2003).

The Neocognitron has two main types of cells: S-cells, which resemble simple cells, and C-cells, which resemble complex cells. A stage is a sequence of two layers of different types, where the first is an S-cell layer and the latter is a C-cell layer. Networks can have one or more stages, e.g., the network on Figure 1 has three stages. Cells in higher stages (closer to the output) tend to have larger receptive fields and to be less sensitive to the position of the stimulus. In a recent version of Neocognitron (Fukushima, 2003), there is also a contrast extraction layer between the input layer and the first S-cell Layer.

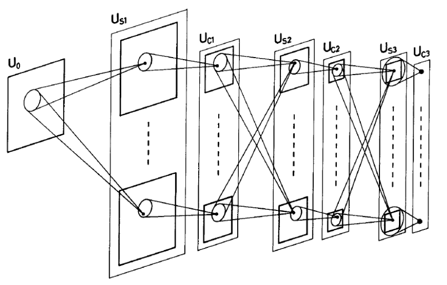

Figure 1: Neocognitron architecture (Fukushima, 1998)

The network outline can be seen in Figure 1. Each square represents a matrix of cells, called a cell-plane. The connections entering each of the cells in a cell-plane are homogeneous and topographically ordered (Fukushima, 1998). The number of stages depends on the data to be classified. If the data has high complexity, the number of stages has to be larger (Fukushima and Wake, 1991). The number of cell-planes in each stage also has to increase as the number of classes to classify increases(Fukushima and Wake, 1991). Other parameters that depend heavily on the data are the size and overlap of the receptive fields. The choice of all these parameters has a strong influence on the results achieved, and at this point we have only a few hints on how best to choose them (Barnard and Casasent, 1990).

For more information about the Neocognitron, consult the appendix.

2.2. Convolutional Neural Networks

Convolutional neural networks (CNNs), like the Neocognitron, are inspired by the classical hypothesis of Hubel and Wiesel, and have some built-in shift and distortion invariance.

The weights of several units are synchronized, so they all represent the same feature (e.g., a vertical line) in different positions of the layer, so that each feature can be detected anywhere in the layer. By making several units represent the same feature at different positions, the output set of units can be seen as a feature map. The feature map can be sequentially implemented by scanning the input image with a template, identifying the positions in which the feature is present. The weight sharing improves the generalization ability by reducing the number of free parameters (LeCun and Bengio, 1998). The idea is that the features can be detected independently of their position. Shifting the input will not change the detected features, only their positions would be shifted as in the input.

Each convolutional layer is followed by another layer that performs a local averaging and subsampling. The local averaging and subsampling layer reduces the resolution of the feature map of the previous layer. These layers are responsible for reducing the importance of the position of the features, allowing some degree of shift and distortion. This idea is also present in the Neocognitron, and is inspired by the classical hypothesis. The convolutional layer resembles a layer of simple cells, and the local averaging and subsampling layer resembles a layer of complex cells. The number of feature maps increases from the input layer towards the output layer as the meaning of each feature becomes more complex. CNNs with a fixed size that share weights along a single temporal dimension are called time-delay neural networks (TDNNs). CNNs for composite object recognition, such as patterns representing words (the number of letters varies), are called space displacement neural networks (SDNNs) and consist of replicated CNNs.

The key difference between the Neocognitron and CNNs is the learning method. The Neocognitron is trained with a crafted algorithm, which is classified under the competitive learning paradigm. CNNs are trained with the backpropagation algorithm, which is a form of gradient descent. This training algorithm is susceptible to local minimal problems. However, in CNNs this problem is slightly less prominent than in fully-connected networks, since the number of free parameters is reduced by weight sharing.

3. Map Transformation Cascade

The Map Transformation Cascade model is also composed of two types of cells, simple and complex. A layer is a set of cells of the same type. A stage is a sequence of two layers of different types, where the first is an S-Layer and the latter is a C-Layer. The S-Layer corresponds to a layer of simple cells in the visual cortex. It maps the input into the class space. The input is tiled with a squared mask, where each sub-pattern is replaced by a number indicating a corresponding class. The masks that tile the pattern may overlap, see Figure 2.

The S-Layer learning is performed by a clustering algorithm. We used K-Means, but it is possible to use other algorithms like self organizing maps (SOM) for this task (Wichert, 1993; Kemke and Wichert, 1993).

The C-Layer, which corresponds to a layer of complex cells in the visual cortex, transforms the input it receives from the S-Layer. Its purpose is

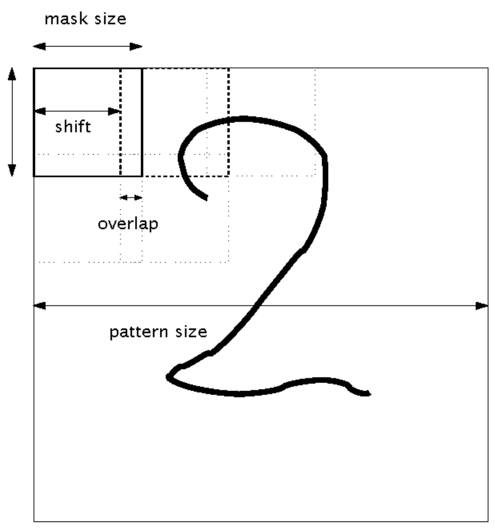

Figure 2: The pattern is tiled with a squared mask. The masks that tile the pattern may overlap. A mask is defined by the mask size. The tiling is defined by the pattern size, the mask size and the overlap. Shift describes how much a mask is shifted in abscissa direction or ordinate direction, with no overlap shift is equal to the mask size.

to allow positional shifts, thus giving the model shift invariance. The class representation of a pattern is tiled with a squared mask, eliminating the positional information of each class inside the mask. The transformation performed by the C-Layer is determined from the start, and is not a result of any learning.

The two layers of a stage are highly related, since the output of the C-Layer is fuzzy version of the S-Layer output. There are two different categories of stages, the input stage and hidden stage, which differ in their input type. In the input stage, the input is a binary pattern; in the hidden stage, it is the representation of the pattern in the class space. The stages of a Map Transformation Cascade can be seen as filters, since they have clear and interpretable output that is a transformation of the input information. Several filters map and transform the input pattern into a space where patterns of the same class are close. The output of the filters is then passed to a simple classifier, which produces a classification for the input pattern. This classification is performed by a supervised classifier, which in our case is the k-Nearest Neighbor. The operation of this network during classification is graphically represented in Figure 3.

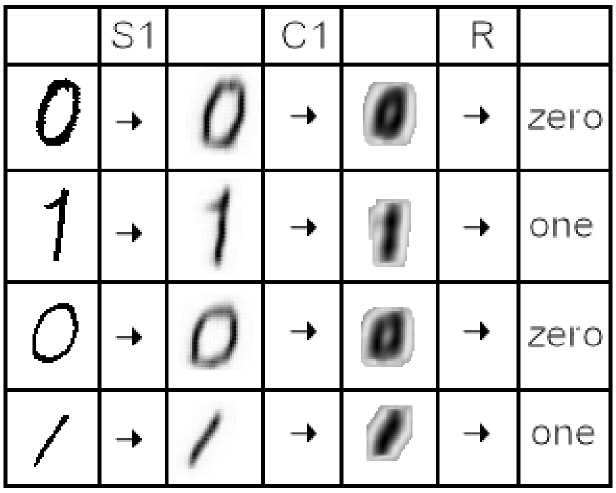

Figure 3: Network operation during classification for four input patterns with one stage.

3.1. Input Stage

The classification of our model begins with the binary input pattern of the S-Layer of the first stage, the input stage. Figure 4 shows an example of such a binary input pattern. The binary input pattern is tiled with a squared mask

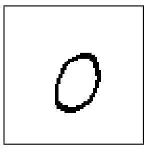

Figure 4: Example of a binary input pattern from the ETL1 data set. The pattern is represented by a vector in which each component has a value of 0 ("zero") or 255 ("one").

M of size $j \times j$ in which a corresponding class is determined, see Figure 2. The class is determined through the use of the elements in each squared mask. Each of the corresponding n sub-patterns $\vec{x}_h$, with $h \in \{1, 2, ..., n\}$, is mapped into a corresponding class represented by a number. Each sub-pattern in each mask corresponds to a $(j \times j)$-dimensional vector. The corresponding classes are learned by K-Means clustering.

3.1.1. Learning phase

During the learning phase of the S-Layer, binary input training patterns are presented to the input stage. They are tiled with a squared mask M of size

$j \times j$. In each position, a training sub-pattern is determined. All sub-patterns representing background, in which all components are zero, are filtered out. As a result, we obtain a training set of n sub-patterns represented by vectors $\vec{x}_h$, with $h \in \{1, 2, ..., n\}$, of dimension $j \times j$. We group the n sub-patterns into clusters represented by the cluster centers c_v. After the clustering, we have cluster centers $\vec{c}_1, \vec{c}_2, \vec{c}_3, ..., \vec{c}_k$ of the clusters $C_1, C_2, C_3, ..., C_k$, with

$$C_v = \{\vec{x}_h | d(\vec{x}_h, \vec{c}_v) = \min_i d(\vec{x}_h, \vec{c}_i)\}, \tag{1}$$

$$c_v = \frac{1}{|C_v|} \sum_{\vec{x}_h \in C_v} \vec{x}_h. \tag{2}$$

The clustering algorithm uses random initialization with numbers from the interval 0 to 255 for each component. The number of clusters is determined experimentally. In Figure 5, we see an example of seven learned cluster centers representing seven different classes, numbered one to seven, and a background class, numbered zero. There are two distinct categories of classes: Classes determined by K-Means clustering, and a special class corresponding to the background information. By convention, a "one" in the input binary pattern represents information, and a "zero," the background of the binary pattern (no information). The solution of the K-Means clustering strongly depends on the initialization of the cluster centers. With random initialization, K-Means clustering on the same data can lead to quite different partition results. This problem may be solved by the adaptive initialization (Wichert et al., 2002). However we opted for a random initialization to improve the random search in the following experiments.

3.1.2. Mapping

During mapping, the corresponding sub-patterns of an input pattern are mapped into the corresponding classes. The binary input pattern is tiled with a squared mask M of size $j \times j$. The tiling can be generated dynamically by scanning a mask over the input pattern. In each position, a mask is applied to the input, and the sub-pattern is compared with the previously learned classes (see Figure 6). For each position, the input is mapped into the most similar class represented by the index i (between 0 and k). The index 0 represents the background, by convention, and the indices 1 to k represent other classes. An index i between 1 to k is the same as the cell-plane index i in the conventional Neocognitron. The index 0 corresponds to the inactivation of cells in all cell planes at a certain position in the conventional Neocognitron.

During mapping, for each sub-pattern $\vec{x}$ the most similar class i is determined according to the Euclidean distance:

$$i = \min_{l} d(\vec{x}, \vec{c_l}), l \in \{0, 1, ..., k\}. \tag{3}$$

In contrast to a conventional Neocognitron, we get a reduced representation of the pattern in the class space. The topological relation between the pattern and its representation in the class space is guaranteed by the background information 0 and the binary nature of the input pattern. In Figure 7 we see the representation of Figure 4 in the class space.

3.1.3. Transformation

The class representation of a pattern is tiled m times with a squared mask M of size $l \times l$. In each position, a vector $\vec{c_h}$, with $h \in \{1, 2, ..., m\}$, of dimensions $l \times l$, is determined. The vector $\vec{c_h}$ describes the presence of some classes inside the mask M. There are two categories of classes: The classes that describe the presence of some features, and the background class represented by zero. The transformation function in the first step eliminates zeros from the corresponding vector. After the first step, the vector represents the classes without exact positional information. In the next step, there are several possible alternatives. We could apply the mode, defined in statistics. The mode is the value that occurs most frequently. The mode is not unique, since the same maximum frequency may be attained at different values. For example the mode of $\{1, 6, 3, 6, 3, 3, 3\}$ is $\{3\}$, but the mode of $\{1, 6, 3, 6, 3\}$ is $\{3, 6\}$. However the mode eliminates too much important information; for example the vector $\{1, 6, 3, 6, 3, 3, 3\}$ indicates the presence of several classes, 1, 3 and 6. Instead of using the mode, we eliminate the frequency information for each present class. As a result the position and frequency information of the classes inside the mask are discarded. The same operation is performed by the C-cells of Neocognitron. The elimination of frequency represents shift invariant operator based on the idea that only changes in a shape matter. A line or a curve is a line or a curve independent of its length. However by gaining shift invariance we lose some discrimination power, both prominent features and small noisy features may be treated equivalently. Figure 8 shows the input into a C-Layer mask. The output of the C-Layer mask for this position will be the set of all present classes. The background and the frequency information is discarded by the C-Layer.

The output of a mask of a C-Layer is represented by a binary vector. A 'one' represents a class at the corresponding position of a binary vector; its absence is denoted by a 'zero'. The class set $\{1, 2, 3, 4, 5, 6, 7\}$ is represented by a binary vector of dimension 7. The presence of the classes $\{1, 6, 3\}$ is represented by the binary vector $\vec{u} = [1\ 0\ 1\ 0\ 0\ 1\ 0]$, with ones in the corresponding positions 1, 3, and 6. We call this vector the uncertainty vector. The class representation of a pattern is transformed into an uncertain class representation. For p classes, we get a uncertainty vector $\vec{u}$ of dimension p. The more ones are present in the uncertainty vector $\vec{u}$, the more information is lost. The result of a transformation of m squared masks M covering a class pattern is a $(m \times p)$-dimensional binary uncertainty class vector $\vec{U}$. The index 1 to m concerns the position of the mask M. This binary vector is composed of m uncertainty vectors

$$\vec{U} = [\vec{u}_1, \vec{u}_2, ..., \vec{u}_m].$$

3.2. Hidden Stage

The uncertainty class vector represents the input to S-Layer of the hidden stage. The uncertainty class vector is tiled with a squared mask M of size $j \times j$ in which a corresponding class is determined. The class is determined through the use of the elements in each squared mask. The elements of each squared mask are represented by binary sub-vectors of dimension $j \times j \times p$. Each of the $j \times j$ elements inside the mask corresponds to a p dimensional uncertainty vector. The corresponding classes are learned by a clustering algorithm.

3.2.1. Learning phase

During the learning phase of the S-Layer, the uncertainty class vectors that were determined by some previous stage are presented to the hidden stage. They are tiled with a squared mask M of size $j \times j$. In each position, a training sub-pattern is determined. All sub-patterns representing the background, in which all components are zero, are filtered out. The elements of a squared mask are represented by binary sub-vectors of the dimension $j \times j \times p$.

As a result, we obtain a training set of n sub-patterns represented by vectors $\vec{U}_h$, with $h \in \{1, 2, ..., n\}$. We group the n sub-patterns into clusters represented by the cluster centers c_v. After the clustering, we have cluster centers $\vec{c}_1, \vec{c}_2, \vec{c}_3, ..., \vec{c}_k$ for the clusters $C_1, C_2, C_3, ..., C_k$, with

$$C_v = \{\vec{U}_h | d(\vec{U}_h, \vec{c}_v) = \min_i d(\vec{U}_h, \vec{c}_i)\}, \tag{4}$$

$$c_v = \frac{1}{|C_v|} \sum_{\vec{U}_h \in C_v} \vec{U}_h. \tag{5}$$

The clustering algorithm uses random initialization with numbers from the interval 0 to 1 for each component. The number of clusters is determined experimentally. As in the input stage, there are two distinct categories of classes: Classes determined by clustering, and a special class corresponding to the background information.

3.2.2. Mapping

The uncertainty class vector is tiled with a squared mask M of size $j \times j$, in which a corresponding class is determined. The class is determined through the use of the elements in each squared mask. The elements of a squared mask are represented by binary sub-vectors of dimension $j \times j \times p$. In each position, each mask is compared with the previously learned classes.

For each position, the input is mapped to the most similar class, according to the Euclidean distance represented by the index $i \in \{0, \cdots, k\}$. The index 0 represents the background by convention, and the indices 1 to k represent other classes.

$$i = \min_l d(\vec{U}_h, \vec{c}_l), l \in \{0, 1, ..., k\} \tag{6}$$

In the corresponding stages, the process is repeated. Each stage acts as a filter that reduces the information. Finally, the last layer corresponds to a classifier, which in our case is realized by a simple nearest neighbor.

3.2.3. Transformation

The transformation in the hidden stage is the same as in the input stage. The class representation of a pattern is transformed into a uncertain class representation. The exact positional and frequency information for each class present inside a mask are eliminated. The result of the transformation is a binary uncertainty class vector $\vec{U}$.

3.3. Recognition Layer

In the R-Layer, the binary uncertainty class vector is classified by a supervised classifier. We use the simplest of all machine learning algorithms, the nearest neighbor method. It is the k-nearest neighbor method with $k = 1$. The distance between the binary uncertainty class vector and previous learned vectors is calculated by the Euclidean distance. The vector is assigned to the most similar class of the previously learned vectors.

3.4. Principles

The Map Transformation Cascade resembles Neocognitron in its key principles. It also has two types of layers, which represent simple and complex cell-layers. However, instead of having several outputs for each position in the input, namely one output for each cell-plane, each layer has only one output for each position.

In Neocognitron the output indicates how strongly each feature is present in each position. Each S-cell-plane learns to recognize a different feature, and the number of cell-planes depends on the selectivity threshold during the learning phase (Fukushima and Tanigawa, 1996). During recognition, the threshold should be set lower to improve generalization (Fukushima and Tanigawa, 1996).

The Map Transformation Cascade implements the winner-takes-all principle during the learning and recognition phases. We used the most simple, unsupervised learning method, the K-Means clustering method. Instead of measuring the cosine similarity between the input vector and the weight vector, as is done in Neocognitron (Fukushima, 1989), we measure the Euclidean distance between the two vectors. During recognition only the most commonly present feature is chosen. By doing so, we get a representation of the pattern in the class space.

The topological relation between the pattern and its representation in the class space is guaranteed by the background information. Without the background, all topological information about the original image is lost. The loss of information is not provoked by the transformation operation (C-cell layer in the Neocognitron). This becomes clear by performing repeated map transformations. Already in the second stage, all topological information about the input pattern is lost. The second stage uses the class space as the input, and the class space has no topological relation with the original space, the input space. A number that indicates a corresponding class replaces each sub-pattern in a mask, and the number has no relation to the corresponding sub-pattern of the input space itself. In the class space 1 and 2 are as similar as 1 and 6. Since the classes are unary encoded forming the binary uncertainty vector, all different classes are equally distant according to the euclidean distance. The same principle holds in Neocognitron. This doesnt mean that the corresponding sub-patterns have the same relation, see for example Figure 5 (b), (c) and (g). The figure 7 represents the pattern of Figure 4 in the class space, the class 1 is as similar to 2 as 1 is to 6, however the corresponding sub-patterns of the class 1 and 6 are more similar to each other than 1 and 2). This error grows with corresponding layers recursively. This loss could be compensated to some extend by using Self-Organizing Maps (SOM) (Wichert, 1993; Kemke and Wichert, 1993). The essential background information is also represented by Neocognitron. It corresponds to the inactivation of cells in all cell planes at a certain position.

The recognition layer was introduced. Even though the conventional Neocognitron does not have this additional layer, the highest stage resembles this behavior. A similar function as well was already introduced in (Shouno et al., 1999) for evaluation. In Neocognitron, the learning process for the highest stage obeys different rules, and simple cells, unlike in the previous layers, have a label indicating which class the cell represents. The purpose of clearly separating the recognition layer is to allow it to function in different ways, using well known classification algorithms such as KNN, RBF.

The choice of masks to ensure shift invariance and discrimination is a difficult process that has to be made with care. The choice leads either to an architecture that is reasonably invariant to shifts at the expense of losing a certain amount of discrimination power, or to sharp discriminatory power by sacrificing shift invariance (Barnard and Casasent, 1990). We solve this problem through a random search over the sizes of different masks.

4. Experiments

The experiments are conducted on the ETL1 data set of handwritten characters [2]. The data set contains handwritten letters and digits. Each pattern is a 64×63 image (*horizontal* $\times$ *vertical*). All the patterns are labeled. The digits which total 14450 patterns, are used in the experiments. The dimensions images are 64×63, and they are in gray scale. The train

[2]ETL1 database, http://www.etl.go.jp/ etlcdb/index.htm.

and validation sets are generated randomly from the data set, either for each generation or for each experiment. The images were preprocessed using thrno [3], a binarization program based on the discriminant criterion (Otsu, 1979). In a recent version of Neocognitron, there is a special layer of cells that is responsible for contrast-extraction. The thrno software is used to achieve a similar effect without adding extra complexity to the model, and keeping it focused on the key principles. An additional background column is added to the bottom of the patterns to make them square (64×64). All the masks in the experiments are also square.

4.1. Parametrization

Like the previous models, the proposed model is highly susceptible to the network parameters (Barnard and Casasent, 1990). Several networks with incorrect parameters for this data set performed as badly as randomly classifying the input, i.e., picking a random number between 0 and 9.

The necessary parameters to define a layer are size, shift, frame, and for the S-layers, number of classes. As for the Recognitron Layer, size derives from the parameters of the previous layers. Its size is always the size of the output of the last stage.

The S-Layer can be predetermined or not. A predetermined S-Layer is a layer of the first stage with a predetermined number of classes, each of which represents a straight line with a specific orientation. The concept is the same as for Neocognitron, and was described in Section Appendix A.3. The R-Layer should always have the same size of the output of the last stage. The overlap of a mask is defined by the following equation (see Figure 2):

$$overlap = size - shift, \tag{7}$$

and the number of cells per dimension, or equivalently the number of mask positions per dimension, is defined by the following expression:

$$mask\ positions\ per\ dimension = \lfloor \frac{pattern\ size - size}{shift} \rfloor + 1. \tag{8}$$

For $overlap = 0$ it is simply

$$mask\ positions\ per\ dimension = \lfloor \frac{pattern\ size}{size} \rfloor. \tag{9}$$

[3]Taiichi Saito, http://www.is.aist.go.jp/etlcdb/util/thrno.htm.

Additionally we introduce the frame size. The frame artificially enlarges the size of the input pattern with background information around the original pattern. Its purpose is to define the input information for the parts of a mask that lie outside the original pattern. With frame the number of cells per dimension, or equivalently the number of mask positions per dimension, is

$$mask\ positions\ per\ dimension = \lfloor \frac{pattern\ size + frame \times 2 - size}{shift} \rfloor + 1. \tag{10}$$

4.2. Parameter optimization

The model has high sensitivity to the parametrization. Genetic algorithms usually have two genetic operators: Crossover and mutation (Srinivas and Patnaik, 1994). The crossover operator's purpose is commonly defined as to find good solutions by combining individuals. The mutation operator is commonly considered less important and its purpose is to introduce random changes to avoid local minimums (Tomassini, 1995). A random search algorithm with a mutation operator and a fitness function is used to solve this problem. These changes in individuals are followed by the selection of the fittest ones. No crossover was used in order to avoid additional complexity in defining the operator, and also because mutation is more useful when the population is converging (Beasley et al., 1993; Davis et al., 1991). The optimization takes place through the selection and mutation of individuals.

In each experiment the population has a fixed size. In each generation the current members are mutated and added to the population. Afterwards, the already existing members and the new ones are evaluated according to a fitness function. The population is then trimmed to its defined size by keeping the best individuals, which can be both current members and new ones.

Besides comparing networks with different characteristics, the main purpose is to understand the validity of the model in terms of performance, i.e., how well the model can perform with the right parameters on this data set, and how good that performance is relative to Neocognitron's. Due to the large amount of time required by some of the optimizations, no predetermined stopping criteria is established unless explicitly mentioned. Two stopping criteria for the random search are empirically used: Classification rate stabilization and population age increase. The first one indicates that the networks in the population are not getting better in terms of performance, even though they may be changing. The second one indicates that fewer individuals with improved performance over the existing ones are being generated, and therefore the population is stabilizing.

The clustering algorithm used in the experiments has a random initialization, which often makes the results differ for the same network on the same data sets. This characteristic intuitively reduces local minima problems and improves diversity, although it also affects the convergence of the optimization. For simplicity all networks use nearest the neighbor method in the Recognition Layer; this value was not subject to any optimization. Some of the optimizations used the results of previously realized optimizations. The best individual after optimization for each of the populations of the experiments is shown in Table 4 and their performance on a test set is shown in Table 5.

The training classification rate for the best individuals of all experiments is often 100%. Because of this, the training set classification rate are not used to analyze the relative performance of the networks, since it offers no guidance. It would be possible for the model to reject a pattern, either by a minimum similarity threshold or by not choosing a class when two or more classes are equally probable. However, to facilitate the comparison with Neocognitrons results and to limit the complexity of the experiments, rejection will not be considered. The validation set classification rate will serve as a basis for the parameter optimization analysis, and will also be referred to as classification rate.

The optimization will be performed primarily on the cell size. The number of cell classes, and in some experiments the cell frame, will also be optimized.

Early experiments, which tried to optimize the cell shift, proved to be useless given the great changes they made on the output of each layer. An increase in the cell shift from 1 to 2 reduces the size of the output almost by half. These drastic changes in the output size make it difficult for the following layers to process the output with good results. A more complex mutation is probably needed to tackle this problem successfully. Because of this, the optimization of the cell shift was abandoned.

Early experiments with an optimization of the number of stages, besides the cell size and the cell classes, were conducted. Their strong tendency to produce one-stage networks made their usefulness unclear. This tendency could be due to a real better performance of one-stage networks or to an advantage of the one-stage networks in the optimization process given their smaller number of parameters. The optimization of the number of stages in a single population was abandoned.

Outline of Experiments. The goal of the experiments is to find a parametrization of the Map Transformation Cascade that can achieve good results on the ETL1 data set. Besides this goal, the experiments will also serve to analyze different parameterizations. The optimizations maximize the following fitness function:

$$f(\text{parametrization}) = \frac{\#\text{ validation patterns correctly classified}}{\#\text{ validation patterns}}, \qquad (11)$$

i.e., the optimizations maximize the validation set classification rate. Each experiment is run with a population of size 3, and 2 additional children per individual of the population. Each child is a copy of one of the members of the population and is then mutated according to Table 1, using the following expression [4]:

$$\textit{new value} = \begin{cases} \textit{old value} + \lceil \textit{old value} \times \textit{mutation} \rceil & \text{if } \textit{mutation} \geq 0 \\ \textit{old value} + \lfloor \textit{old value} \times \textit{mutation} \rfloor & \text{if } \textit{mutation} < 0 \end{cases}. \qquad (12)$$

In each iteration of an optimization, 2 children are generated for each of the existing 3 members, resulting in 9 members, from which the best 3 become the next population. In the legend of the charts,“per. Mov. Avg.” refers to “period Moving Average”.

Cell Property	Mutation Applied
Classes	$[-30\%, +30\%]$
Size	$[-10\%, +10\%]$
Shift	0%
Frame	$[-10\%, +10\%]$

Table 1: Mutations applied to the existing parameterizations.

The first experiment, Thinning-Out (see Sec. 4.2.1), analyzes two populations with different thinning-out methods: One that performs the thinning-out from the S-Layer to the C-Layer and another one that performs it from

[4]Each parameter is represented by an integer with no specific number of bits assigned.

the C-Layer to the S-Layer. This experiment is used to choose a thinning-out method for the remaining experiments.

A second experiment, Stages (see Sec. 4.2.2), analyzes three populations with different numbers of stages and a predetermined S-Layer. The first population is composed of networks with three stages, the second has networks with two stages and the third has networks with one stage. The results of the optimizations of this experiment are used as seeds for another experiment without a predetermined S-Layer.

The third experiment, No predetermined S-Layer (see Sec. 4.2.3), also analyzes three populations with different numbers of stages, but without a predetermined S-Layer. This experiment allows a comparison between networks with different number of stages without a predetermined S-Layer. It also allows a comparison between networks with or without a predetermined S-Layer. The best individual from all the previous experiments will be used in a final optimization.

The fourth and final experiment, Large Set (see Sec. 4.2.4), analyzes a single population whose seed individual is the best individual so far from the previous experiments. The optimization is performed with a larger validation set which is randomly generated for each generation. The increased size of the validation set results in an improved level of stability of the optimization. The best individual of this population is finally tested on different size test sets.

4.2.1. Thinning-out

The purpose of this experiment is to evaluate the performance of the proposed model with different thinning-out methods. Thinning-out refers to a reduction in the number of cells in two contiguous layers. Two thinning-out methods will be considered, from the S-Layer to the C-Layer and from the C-Layer to the S-Layer.

This experiment is performed on networks with one stage because they have fewer parameters than networks with more stages. With fewer parameters, the optimizations can progress more quickly toward a higher classification rate. The seed individuals for both populations have minimal size cells (2×2) and have no frame (see Table 3). In the experiment, the stopping criterion is a maximum number of generations (360). This criterion is used because the seed individuals of both populations were not derived from any previous optimization, which allowed for a more faithful comparison.

The training and validation data sets with 20 patterns were generated each time a new generation started. These small sets were used to reduce the time needed to compute each generation. The size of the sets makes the margin of error of the classification rate considerable. This margin of error sometimes makes the selection of the best individuals wrong, affecting the convergence of the optimization although the purpose of this experiment is just to make a relative comparison.

The mutations used in all experiments are shown in Table 1.

There is no evidence that one of the thinning-out methods can outperform the other for this data set (see Figure 9 and 10). The C-Layer to S-Layer thinning-out and the S-Layer to C-Layer thinning-out show similar performance. Both thinning-out options empirically appear to be valid. The C to S-Layer thinning-out, which is uncommon in related models, seems less susceptible to the remaining parametrization of the network given its early classification rate burst. This thinning-out method is used in further experiments, because of its less common usage and suggested similar performance.

4.2.2. Stages

The purpose of this experiment is to evaluate the performance of the proposed model using different numbers of stages. All networks have a predetermined S-Layer, as is reported in a Neocognitron version for the same task (Fukushima, 2003). The predetermined S-Layer was previously discussed in Section Appendix A.3.

The seed individual for the 3-stage, predetermined-S-Layer population was based on a Neocognitron parametrization for the same task (Fukushima, 2003), although the thinning-out takes place from the C to the S-Layer as discussed in Section 4.2.1.

The first optimization has 3 stages with a predetermined S-Layer. The best individual of this population after optimization, was used as a seed for the 2-stage, predetermined-S-Layer, with the last stage removed. The results of the 2 stages with a predetermined S-Layer were used in the same way for the 1-stage, predetermined-S-Layer optimization.

The training and validation data sets with 200 patterns were generated at the beginning of the experiment and were used throughout the experiment.

The 1-stage, predetermined-S-Layer achieved a significantly poor validation classification rate (77% see Figure 13) when compared with the 2-stage, predermined-S-Layer population (89,8% see Figure 12) and the 3-stage, predetermined-S-Layer population (87,7% see Figure 11).

The 1-stage, predetermined-S-Layer population classification rate exhibits monotonic growth (see Figure 13) because there is no random initialization in its networks. This means that the 1-stage predetermined-S-Layer population has no way of adapting besides its parametrization, which empirically explains its relatively poor performance. The performance of the predetermined S-Layer appears to be poor when no other S-Layer exists.

The 2-stage, predetermined-S-Layer population seems to perform better than the 3-stage, predetermined-S-Layer population. An explanation for this is that 2 stages fit the data set better than 3 stages. However, to some extent, this can also be explained by the smaller number of parameters in the 2-stage networks.

The predetermined S-Layer is based on Neocognitron (Fukushima, 2003). The next experiment analyzes the relative performance of this characteristic for the proposed model. Three optimizations, which are similar to the ones in this experiment but without the predetermined S-Layer, are analyzed in the next experiment to analyze the influence of the predetermined S-Layer on the performance of the model.

4.2.3. No Predetermined S-Layer

The purpose of this experiment is to understand the influence of the predetermined S-Layer on performance by comparing similar networks with or without a predetermined S-Layer. The best individuals from Section 4.2.2 are used as seeds for the three populations without a predetermined S-Layer (see Table 3).

The 3-stage population achieved an 88,4% (see Figure 14) classification rate on the validation set, compared to 87,7% for the 3-stage, predetermined-S-Layer one (see Figure 11). The 2-stage population achieved an 89,5% classification rate on the validation set (see Figure 15), compared to 89,8% for the 2-stage, predetermined-S-Layer one (see Figure 12). The 1-stage population achieved the best results, with an 91,1% classification rate on the validation set (see Figure 16), a great improvement when compared with 77,0% for the 1-stage, predetermined-S-Layer population (see Figure 13).

The networks without a predetermined S-layer achieved a similar or better validation classification rate for this data set. This is empirically explained by their higher adaption capacity, since the first S-Layer classes can represent any kind of feature. As for the predetermined S-Layer, its classes represent straight lines of different orientations.

4.2.4. Large Set

The purpose of this experiment is to carry out a final evaluation of the best network so far, taken from the 1-stage population. The training set size remains 200 and the validation set size is increased to 500. Finally, the best individual is tested with different size test sets.

Test Set Size	Classification Rate
200	93,00%
500	92,00%
1000	94,20%
3000	93,33%
10000	92,91%

Table 2: Test set classification rates of the best individual of the Large Set population for different test sets.

The Large Set population (LS) achieved a significant increase in the classification rate over the 1 stage population (1S), which served as the seed for this optimization. The increased test set size seems to have an important impact on the results, given the small number of generations and the significant increase in the classification rate. The 1S test classification rate is stable for 15 generations (see Figure 16), and the LS validation classification rate improved 1,8% (see Figure 17) in just 11 generations. This suggests that to optimize the populations above an already high level of fitness, it is necessary to increase the validation set size. Empirically, this is explained by the reduction in the margin of error of the fitness function, which may be particularly significant for these individuals due to their random initialization.

The experiments with the best individual from the LS population show a good performance of the model on this data set with 200 training patterns (see Table 2). The training set is not increased, to prevent over-fitting due to the use of the nearest neighbor classifier.

4.3. Summary of the results

The seed individual of each optimization is shown in Table 3, the best individual of each population is shown in Table 4 and the results of the best individuals in a 200 patterns test set are shown in Table 5.

It is interesting that the networks with only 1 stage and no predetermined S-Layer were the ones that achieved the best performance, 92,91% (see Table

2), a result which is slightly better than Neocognitron's 92,8% test classification rate, also with 200 training patterns for this same data set (Fukushima, 2003) and a 3000-pattern test set.

This result shows a slightly improved performance of the Map Transformation Cascade compared to Neocognitron. The network parametrization, is however, quite different from the reported parametrization of Neognitron (Fukushima, 2003), which has 4 stages (in Neocognitron the purpose of the last stage is similar to that of the Map Transformation Cascade's recognition layer). In addition, the best parametrization found in the experiments only has one stage and a recognition layer (corresponding to a Neocognitron with 2 stages). The results show that it is possible to achieve the same performance with a much simpler model. The result also suggest that despite the background information, too much information is lost by the class space representation of the pattern.

5. Conclusion

A new hierarchical network model is proposed through the analysis of the key principles of Neocognitron. The purpose of this model is to achieve the same key principles in a simpler way. The main contribution for the simplicity of the Map Transformation Cascade is the learning. It can be performed by well-known learning algorithms, avoiding the additional complexity of a specifically designed learning algorithm.

The cell-planes are not present in the description of our proposed model. In each S-Layer (simple layer), the input is scanned with a mask in all the different positions for all the different classes. The mask has the same behavior in all the different positions, resembling the weight-sharing mechanism and cell-plane representation in Neocognitron. However, only one class can be present at each position inside the mask. This behavior is equivalent to restricting Neocognitron so that there can only be one active cell for each position across all cell-planes and so that its activity for that cell is constant. This behavior reflects an irrevocable decision, made by each layer, on what feature is present in each position, unlike in Neocognitron, where a measure of similarity for each class in each position is passed along. The proposed model's C-Layer receives as input a single class vector. The input is scanned with a mask, and for each mask position, the positions of the classes inside the mask and their frequencies are removed. The proposed model's complex layer operation is similar to Neocognitron's, in fact the algorithmic level of

Layer	Cell Property	STOC	CTOS	3SPD	3S	2SPD	2S	1SPD	1S	LS
S1	size	2	2	6	7	7	9	9	11	11
	shift	1	2	2	2	2	2	2	2	2
	frame	0	0	8	16	16	9	9	4	11
	classes	2	2	16	4	4	7	7	3	85
	overlap	1	0	4	5	5	7	7	9	9
	cells per dimension	63	32	38	45	45	37	37	31	38
	predetermined	N	N	Y	Y	Y	N	Y	N	N
C1	size	2	2	6	3	3	5	5	6	25
	shift	2	1	1	1	1	1	1	1	1
	frame	0	0	3	5	5	2	2	3	7
	overlap	0	1	5	2	2	4	4	5	24
	cells per dimension	31	31	39	53	53	37	37	32	28
S2	size			6	10	10	9			
	shift			2	2	2	2			
	frame			4	3	3	4			
	classes			22	81	81	86			
	overlap			4	8	8	7			
	cells per dimension			21	25	25	19			
C2	size			6	8	8	12			
	shift			1	1	1	1			
	frame			3	1	1	5			
	overlap			5	7	7	11			
	cells per dimension			22	20	20	18			
S3	size			8	11					
	shift			2	2					
	frame			5	0					
	classes			50	89					
	overlap			6	9					
	cells per dimension			13	5					
C3	size			9	7					
	shift			1	1					
	frame			2	5					
	overlap			8	6					
	cells per dimension			9	9					
R	size	31	31	9	9	20	18	37	32	28

Table 3: Seed individuals for each of the populations: S-Layer to C-Layer thinning-out (STOC), C-Layer to S-Layer thinning-out (CTOS), 3-stage, predetermined S-Layer (3SPD), 3-stage (3S), 2-stage, predetermined-S-Layer (2SPD), 2-stage (2S), 1-stage, predetermined-S-Layer (1SPD), 1 stage (1S) and Large Set (LS).

Layer	Cell Property	STOC	CTOS	3SPD	3S	2SPD	2S	1SPD	1S	LS
S1	size	12	14	7	7	9	7	11	11	12
	shift	1	2	2	2	2	2	2	2	2
	frame	0	0	16	10	9	19	4	11	10
	classes	138	175	4	22	7	24	3	85	104
	overlap	11	12	5	5	7	5	9	9	10
	cells per dimension	53	26	45	39	37	48	31	38	37
	predetermined	N	N	Y	N	Y	N	Y	N	N
C1	size	23	20	3	7	5	4	6	25	20
	shift	2	1	1	1	1	1	1	1	1
	frame	0	0	5	2	2	5	3	7	5
	overlap	21	19	2	6	4	3	5	24	19
	cells per dimension	16	7	53	37	37	55	32	28	28
S2	size			10	7	9	6			
	shift			2	2	2	2			
	frame			3	2	4	3			
	classes			81	58	86	72			
	overlap			8	5	7	4			
	cells per dimension			25	18	19	28			
C2	size			8	9	12	16			
	shift			1	1	1	1			
	frame			1	7	5	3			
	overlap			7	8	11	15			
	cells per dimension			20	24	18	19			
S3	size			11	12					
	shift			2	2					
	frame			0	0					
	classes			89	80					
	overlap			9	10					
	cells per dimension			5	7					
C3	size			7	13					
	shift			1	1					
	frame			5	6					
	overlap			6	12					
	cells per dimension			9	7					
R	size	16	7	9	7	18	19	32	28	28

Table 4: Best individual after optimization for each of the populations: S-Layer to C-Layer thinning-out (STOC), C-Layer to S-Layer thinning-out (CTOS), 3-stage, predetermined-S-Layer (3SPD), 3-stage (3S), 2- stage, predetermined-S-Layer (2SPD), 2-stage (2S), 1-stage, predetermined-S-Layer (1SPD), 1-stage (1S) and Large test set (LS).

information representation is the same, but the word level of explanation differs.

The unsupervised learning method in Neocognitron was specifically developed for it, and can be classified under the competitive learning paradigm (Fukushima, 1989). Cells compete with each other, making each cell specialize in a certain stimulus. In the proposed model, the learning algorithms are not specifically tailored, and well-known algorithms can be used.

There are two different learning needs in this problem: One in the stages, where the purpose is information reduction and a clustering algorithm is the most suitable (e.g., K-Means, SOM), the other in the recognition layer, where the purpose is classification and a supervised classifier is the most suitable (e.g., KNN, RBF). In the stage's clustering algorithm, each cluster is equivalent to a specialization in a certain stimulus in Neocognitron, i.e., each cluster is a template. The recognition layer can use any kind of classifier to map the output of the last stage into a class.

Hierarchical networks are highly sensitive to parametrization. A random search was used to tune the model for the ETL1 data set of handwritten digits.

The results of the experiments show good performance, achieving a 92,91% test classification rate with only a 200-pattern training set. This result is similar to Neocognitron's. The network parametrization that performed best,

Architecture Name	Average Test CR	Test CR Standard Deviation
STOC	89,20%	0,95%
CTOS	88,05%	1,38%
3SPD	86,25%	1,34%
2SPD	89,10%	2,35%
1SPD	74,00%	0,00%
3S	86,85%	3,50%
2S	89,15%	1,49%
1S	90,85%	1,16%
LS	91,75%	0,75%

Table 5: Results for a train set and a test set with 200 patterns each. The average test classification rate and standard deviation of 10 runs is shown for each of the population's best individual: S-Layer to C-Layer thinning-out (STOC), C-Layer to S-Layer thinning-out (CTOS), 3-stage, predetermined-S-Layer (3SPD), 3-stage (3S), 2-stage, predetermined-S-Layer (2SPD), 2-stage (2S), 1-stage, predetermined-S-Layer (1SPD), 1-stage (1S) and Large test set (LS).

however, is quite different from Neognitron's (Fukushima, 2003), which has 4 stages. The best parametrization found in the experiments has only one stage and a recognition layer (corresponding to a Neocognitron with 2 stages), a much simpler network parametrization.

Models like Neocognitron use the class space combined with the background information. Without the background, all topological information about the original image is lost in the following stages. The principle of this classification technique is the usage of the background information and the loss of information provoked by the class representation. Consequently it is suggested that no significant advantage is achieved by the usage of more than two stages.

The Map Transformation Cascade can be understood as a sequence of filters that transforms the input pattern into a space where patterns of the same class are close. The output of the filters is then passed to a simple classifier, which produces a classification for the input pattern. The experiments conducted show that Map Transformation Cascade's performance is similar to Neocognitron's.

Acknowledgments. The author would like to gratefully acknowledge two anonymous reviewers and the editor for their valuable suggestions.

Appendix A. Appendix

There are several versions of the Neocognitron. To give a short explanation of the model, we will focus on its most common characteristics (Fukushima, 1988, 2003).

Appendix A.1. S-cell Layers

S-cells resemble simple cells in the visual cortex. They are feature-extracting cells. The connections converging to these cells can be modified through learning, making each of them react to a distinct feature. All of the S-cells in a cell-plane react to the same feature. This is accomplished by making all of them share the same weights, which results in the capacity to identify features independently of the position in which they are presented. Each of the cell-planes in an S-cell layer is a feature map. They recognize a certain stimulus at different positions. Each cell-plane only recognizes one stimulus, i.e., they only have one template to which they tend to react. The number of cell-planes in each S-cell layer tends to be greater in higher stages.

Each S-cell has variable excitatory connections from the preceding layer, and a variable inhibitory connection from a V-cell. The excitatory connections in the same region are fed into the S-cell and into the V-cell. The output of an S-cell of the kth cell-plane of the lth stage with receptive field center location $\boldsymbol{n}$ is determined by the following equation (Fukushima, 2003):

$$u_{Sl}(\boldsymbol{n}, k) = \frac{\theta_l}{1-\theta_l}\varphi\left[\frac{1 + \sum_{\kappa=1}^{K_{Cl}} \sum_{|\boldsymbol{v}|<A_{Sl}} a_{Sl}(\boldsymbol{v}, \kappa, k) u_{Cl-1}(\boldsymbol{n}+\boldsymbol{v}, \kappa)}{1 + \theta_l b_{Sl}(k) v_l(\ \boldsymbol{n})} - 1\right], \tag{A.1}$$

where $a_{Sl}(\boldsymbol{v}, \kappa, k)$ is the strength of the variable excitatory connection coming from the C-cell $u_{Cl-1}(\boldsymbol{n}+\boldsymbol{v}, \kappa)$ of the κth cell-plane of the preceding stage, $b_{Sl}(k)(\geq 0)$ is the strength of the variable inhibitory connection from the V-cell, A_{Sl} is the radius of the connectable area of the S-cell, $\theta_l(> 0)$ is a constant threshold that controls the selectivity of the S-cells and $\varphi[]$ is a nonlinear function defined by

$$\varphi[x] = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}. \tag{A.2}$$

A high threshold (θ_l) makes the cells more selective, whereas a low threshold makes them less selective.

The output of a V-cell in layer u_{Sl} is given by the following equation (Fukushima, 2003):

$$v_l(\boldsymbol{n}) = \sqrt{\sum_{\kappa=1}^{K_{Cl-1}} \sum_{|\boldsymbol{v}|<A_{Sl}} c_{Sl}(\boldsymbol{v}) \left\{u_{Cl-1}(\boldsymbol{n}+\boldsymbol{v}, \kappa)\right\}^2}, \tag{A.3}$$

where $c_{Sl}(\boldsymbol{v})$ is a monotonically decreasing function of $|\boldsymbol{v}|$ representing the strength of the fixed excitatory connections to the V-cell.

Appendix A.2. C-cell Layers

C-cells resemble complex cells in the visual cortex. Their purpose is to allow positional changes and distortions of the features. They do this by blurring the stimulus they receive. In most Neocognitron versions, their input connections from the S-cells are fixed and invariable, which means they do not learn. Each C-cell receives connections from a group of S-cells that extract the same feature from slightly different positions. This makes the same C-cell respond if the stimulus feature is slightly shifted and a nearby

S-cell is activated instead. The density of cells in both the S-cell and the C-cell layers tends to decrease towards the higher-order stages becomes higher. In the output layer, there is a 1×1 matrix for each class that the network classifies. The matrix with the highest response is the classification yielded by the network.

The output of a C-cell of the kth cell-plane of the lth stage with receptive field center location $\boldsymbol{n}$ is determined by the following equation (Fukushima, 2003):

$$u_{Cl}(\boldsymbol{n}, k) = \psi \left[\sum_{|\boldsymbol{v}|<A_{Cl}} a_{Cl}(\boldsymbol{v}) u_{Sl}(\boldsymbol{n}+\boldsymbol{v}, k)) \right], \tag{A.4}$$

where A_{Cl} is the radius of the connectable area of the C-cell and $\psi\,[x] = \varphi[x]/(1+\varphi[x])$.

In Neocognitron, the layer sizes are gradually reduced. The first layer of the first stage has the same size as the patterns to be recognized, and the last layer of the last stage is 1×1. This implies that the dimension of the layers is progressively reduced from the input layer to the output layer. The thinning takes place through the integration of local features into more global features. The dimension of the layers is reduced from the S-cell layers to the C-cell layers of the same stage (Fukushima, 2003), i.e., their output size is smaller than their input size. For example, in a version for handwritten digit recognition (Fukushima, 2003) the thinning-out ratio is 2:1.

Appendix A.3. Learning

There are several training methods in different versions of Neocognitron. The Neocognitron network can be trained by unsupervised or supervised learning. Another important distinction in training methods is between simultaneous and sequential construction (Fukushima, 1999). In simultaneous construction, the learning of all layers of the network progresses simultaneously. In sequential construction, each stage is trained separately, starting from the ones closest to the input layer and progressing to the ones closer to the output layer, where the training of a layer only starts when the training of all the preceding layers is completely finished. Simultaneous learning has a slow learning speed, but can accept incremental learning. On the other hand, sequential learning can finish learning fast but does not accept incremental learning. Sequential learning is more common in recent versions.

Most versions are trained by unsupervised learning. In supervised learning, the "teacher" points to the position of the features to be extracted in the patterns (Fukushima, 1998; Fukushima and Wake, 1991). The cells whose receptive fields coincide with the position of these features become winners. The rest of the learning process is similar to that of the unsupervised learning.

The first layer of S-cells (U_{S1}) has a predetermined number of cell-planes: Each of the cell-planes corresponds to a template. Each of these predetermined cell-planes represents a straight line with a specific orientation. A common number of cell-planes for U_{S1} is 16, so each cell-plane differs by $\frac{2\pi}{16}$ (Fukushima, 2003). The intermediate layers have a variable number of cell-planes, which depends on a selectivity threshold.

Appendix A.4. S-cells learning

S-cells compete with each other inside their competition area. When an input is presented to the layer, there can only be one winner cell inside each competition area. Only the winner cells have their input connections increased. The increase in the connection of a winner cell is proportional to the presynaptic activity. This behavior creates a specialization in the cells: They each specialize in a certain stimulus, i.e. they form a template of the stimulus that made them winners. This principle can be classified under the competitive-learning paradigm (Fukushima, 1989). The threshold θ_S of the S-cell layer determines the number of cell planes generated for the S-cell Layers (except the edge-extracting layer, which has a predetermined number of cell-planes). A lower threshold implies a decreased number of cell-planes and a higher threshold an increased number of cell-planes. Different thresholds can be used in the learning and recognition phase. A high threshold in the learning phase allows more feature-extracting cells to be generated, and a low threshold in the recognition phase allows a greater generalization ability(Fukushima and Tanigawa, 1996; Shouno et al., 1999).

Barnard, E. and Casasent, D. (1990). Shift invariance and the neocognitron. *Neural Networks*, 3(4):403–410.

Beasley, D., Bull, D., and Martin, R. (1993). An Overview of Genetic Algorithms: Part 2, Research Topics. *University Computing*, 15(4):170–181.

Davis, L. et al. (1991). *Handbook of genetic algorithms.* Van Nostrand Reinhold New York.

Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20:121–136.

Fukushima, K. (1980). Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1:119–130.

Fukushima, K. (1989). Analisys of the process of visual pattern recognition by the neocognitron. *Neural Networks*, 2:413–420.

Fukushima, K. (1998). *Neocognitron: a model for visual pattern recognition*, pages 613–617. MIT Press, Cambridge, MA, USA.

Fukushima, K. (1999). Self-organization of shift-invariant receptive fields. *Neural Networks*, 12(6):791–801.

Fukushima, K. (2003). Neocognitron for handwritten digit recognition. *Neurocomputing*, 51:161–180.

Fukushima, K. and Tanigawa, M. (1996). Use of different thresholds in learning and recognition. *Neurocomputing*, pages 1–17.

Fukushima, K. and Wake, N. (1991). Handwritten alphanumeric character recognition by the neocognitron. *IEEE Transactions on Neural Networks*, 2(3):355–365.

Hecht-Nielsen, R. (1989). *Neurocomputing.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Hubel, D. H. (1988). *Eye, Brain, and Vision.* Scientific Ammerican Library, Oxford, England.

Kemke, C. and Wichert, A. (1993). Hierarchical self-organizing feature maps for speech recognition. In *Proceedings World Congres on Neural Networks*, pages 45–47. Lawrence Erlbaum.

LeCun, Y. and Bengio, Y. (1998). *Convolutional networks for images, speech, and time series*, pages 255–258. MIT Press, Cambridge, MA, USA.

Lovell, D., Simon, D., and Tsoi, A. (1993). Improving the performance of the neocognitron. In *Fourth Australian Conference on Neural Networks*, pages 22–25.

Otsu, N. (1979). A threshold selection method from gray level histograms. *IEEE Trans. Systems, Man and Cybernetics*, 9:62–66. minimize inter class variance.

Shouno, H., Fukushima, F., and Okada, M. (1999). Recognition of handwritten digits in the real world by neocognitron. In *Knowledge-based Intelligent Techniques in Character Recognition*. CRC Press.

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition*, page 958, Washington, DC, USA. IEEE Computer Society.

Sinha, A. (1999). An improved recognition module for the identification of handwritten digits. Master's thesis, MIT.

Srinivas, M. and Patnaik, L. (1994). Genetic algorithms: a survey. *Computer*, 27(6):17–26.

Tomassini, M. (1995). A survey of genetic algorithms. *Annual Reviews of Computational Physics*, 3:87–118.

Wichert, A. (1993). MTCn-nets. In *Proceedings World Congres on Neural Networks*, pages 59–62. Lawrence Erlbaum.

Wichert, A., Abler, B., Grothe, J., Walter, H., and Sommer, F. T. (2002). Exploratory analysis of event-related fmri demonstrated in a working memory study. In Sommer, F. and Wichert, A., editors, *Exploratory analysis and data modeling in functional neuroimaging*, chapter 5, pages 77–108. MIT Press, Boston, MA.

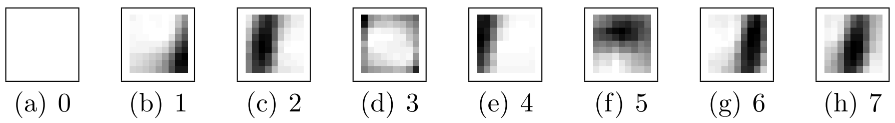

Figure 5: (a) Represents the background information by the class 0. (b) to (h) represent the seven learned classes by 1, 2, 3, 4, 5, 6, 7.

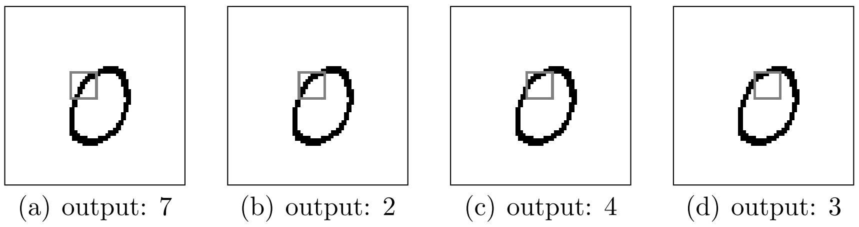

Figure 6: Scanning of a mask over the input pattern of the S-Layer of the input stage. In each position, the sub-pattern in the mask is compared with the previously learned classes. The classes (see Figure 5) are represented by the numbers from 0 to 7.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3	3	1	1	3	3	3	0	0	0	0
0	0	0	0	0	3	1	1	1	1	1	1	3	3	3	0	0	0
0	0	0	0	3	1	1	1	1	5	5	1	7	2	3	3	0	0
0	0	0	3	1	1	7	3	5	5	5	5	5	2	4	3	0	0
0	0	3	1	1	7	5	5	5	5	5	5	5	7	2	4	0	0
0	0	3	1	6	7	2	4	3	3	3	1	6	7	2	4	3	0
0	0	1	6	7	2	4	3	0	0	0	3	6	7	2	4	3	0
0	3	1	6	7	2	4	3	0	0	0	3	6	7	2	4	3	0
0	3	1	7	2	4	3	0	0	0	0	3	6	7	2	4	3	0
0	1	6	7	2	4	3	0	0	0	0	1	6	7	2	4	3	0
0	1	6	7	2	4	0	0	0	0	0	1	6	7	2	4	3	0
0	1	6	7	2	4	0	0	0	0	3	1	6	7	4	4	0	0
0	1	6	7	2	3	0	0	0	3	1	6	7	2	4	3	0	0
0	1	6	7	2	4	3	3	3	1	1	7	7	2	4	3	0	0
0	1	6	7	2	4	1	1	1	1	7	7	2	4	3	0	0	0
0	3	6	5	5	1	1	1	7	7	5	5	3	3	3	0	0	0
0	3	3	5	5	5	5	5	5	5	5	3	3	3	0	0	0	0
0	0	3	3	5	5	5	5	5	4	3	3	0	0	0	0	0	0
0	0	0	3	3	3	3	3	3	3	3	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7: The representation of the pattern of Figure 4 in the class space. Only the information in the center, where the pattern is represented, is shown here. The background information is represented by 0.

0	0	0	1	6
0	0	0	3	6
0	0	0	3	3
0	0	0	0	3
0	0	0	0	0

Figure 8: C-Layer mask input in a given position when scanning Figure 7, its output is $\{1, 6, 3\}$. It indicates the presence of these classes.

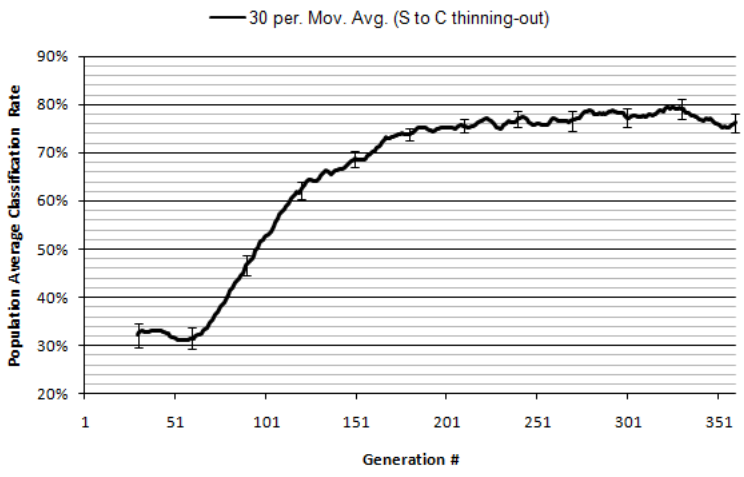

Figure 9: Average validation classification rate of population with 1 stage and thinning-out from the S-Layer to the C-Layer.

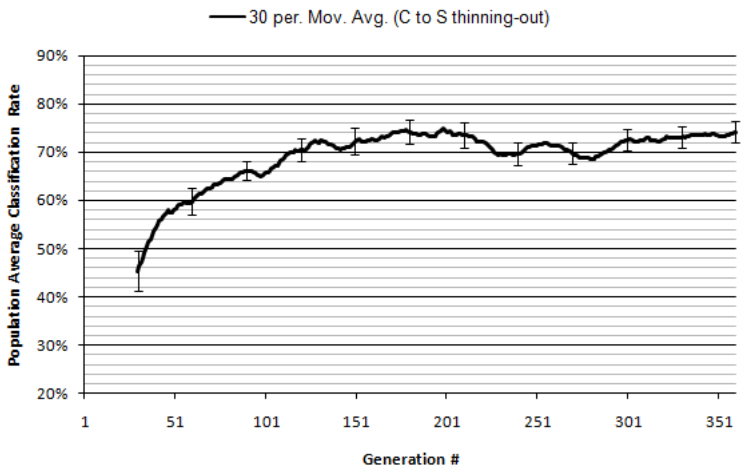

Figure 10: Average validation classification rate of population with 1 stage and thinning-out from the C-Layer to the S-Layer.

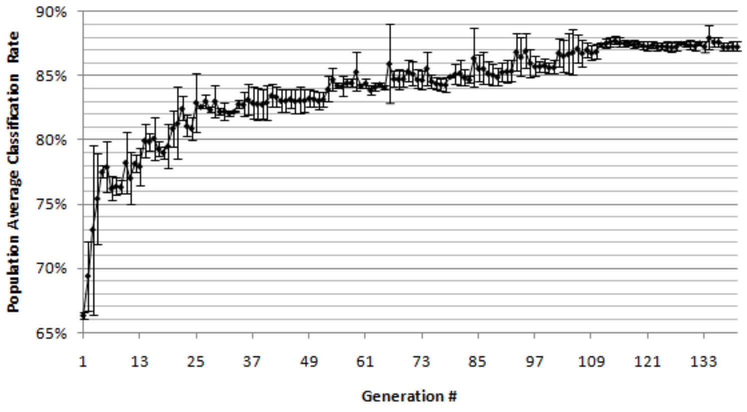

Figure 11: 3-stage, predetermined-S-Layer – validation classification rate. The classification rate shows a significant increase in the first few generations and then tends to stabilize.

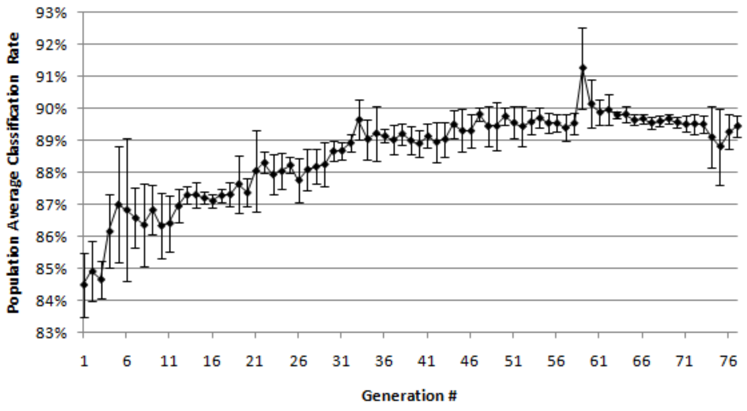

Figure 12: 2-stage, predetermined-S-Layer – validation classification rate. The classification rate starts with a high value. After a high starting value, the classification increases very steadily until it stabilizes. The seed individual for the 2-Stage, predetermined-S-Layer population is the best individual from the 3-Stage, predetermined-S-Layer population. The initial high classification rate empirically suggests that removing a stage from a network that already has good performance does not seriously compromise its results.

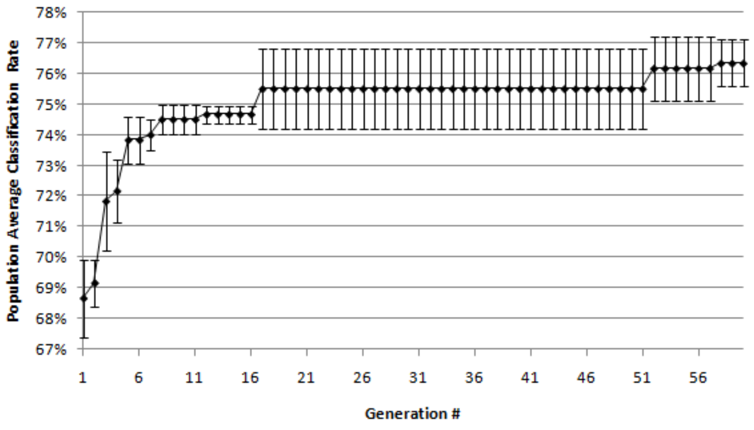

Figure 13: 1-stage, predetermined-S-Layer – validation classification rate. The performance of this population starts with a much lower level than the individual that served as its seed (the best individual of the 2-stage, predetermined-S-Layer population, with the last stage removed). It shows an initial accentuated increase, but then it increases very slowly through several plateaus. This is empirically explained by the fact that this population's only way of adapting is the parametrization, since the only S-Layer is predetermined and no learning takes place.

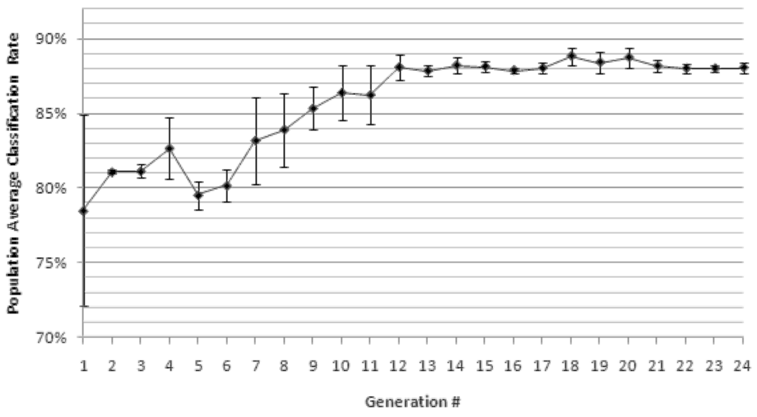

Figure 14: 3-stage population – validation classification rate. The seed individual of this population is the best from the 3-stage, predetermined-S-Layer population. In a few generations, the 3-stage population achieved a slightly better performance than the 3-stage, predetermined-S-Layer one. The classification rate rapidly increases and then stabilizes after a few generations.

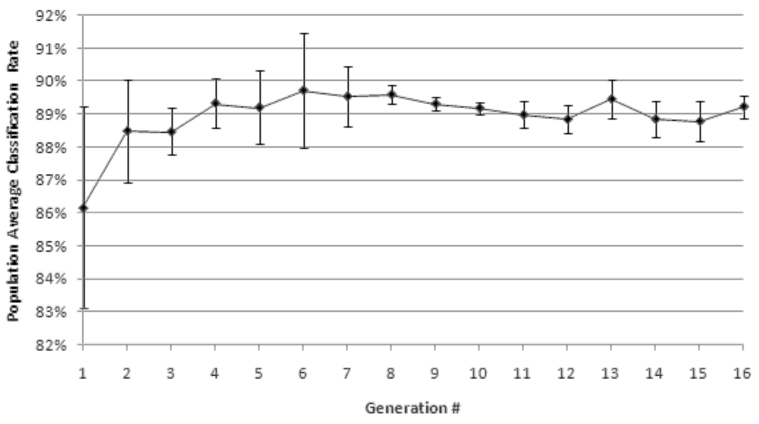

Figure 15: 2-stage population – validation classification rate. The seed individual of this population is the best from the 2-stage, predetermined-S-Layer population. In a few generations, the 2-stage population achieved performance similar to the 2-stage, predetermined-S-Layer one. The classification rate increases slightly in the first 6 generations and then stabilizes.

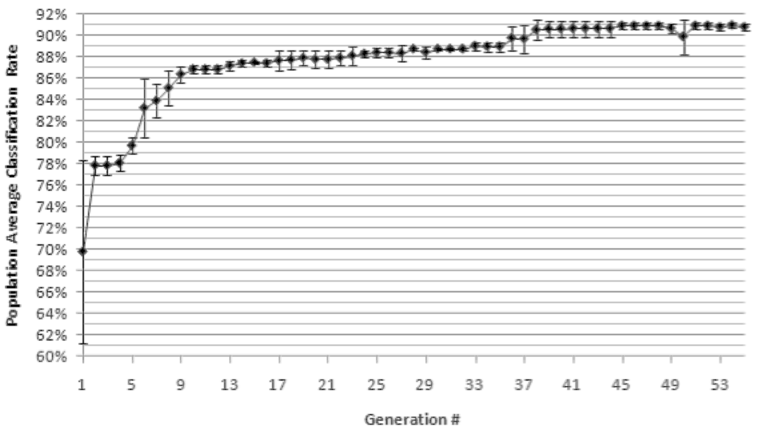

Figure 16: 1-stage population – validation classification rate. The seed individual of this population is the best from the 1-stage, predetermined-S-Layer one. In a few generations, the 1-stage population achieved a much better performance than the 1-stage, predetermined S-Layer one. The classification rate increases very sharply in the beginning and then stabilizes. The performance of this population is clearly better than that of the 1-stage, predetermined-S-Layer population.

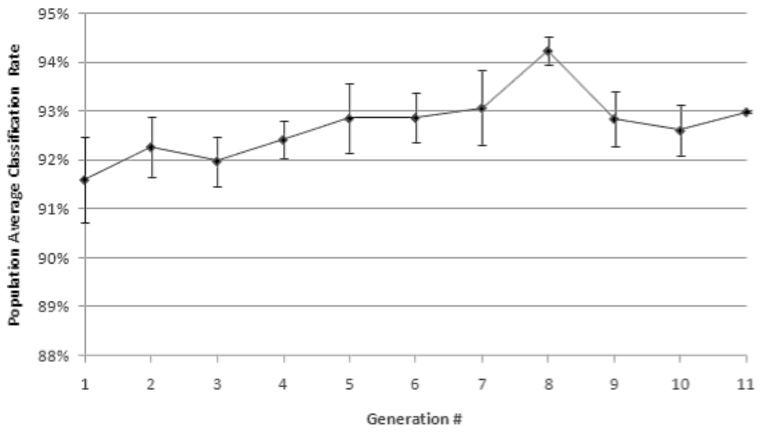

Figure 17: Large Set population – validation classification rate. The seed individual of this population is the best from the 1-stage population. The Large Set population achieved a significant increase in the classification rate over the 1-stage population that served as seed for this optimization.