Time Delay Neocognitron
A hierarchical network for visual pattern recognition.

Ángelo Cardoso
angelo.cardoso@ist.utl.pt
Instituto Superior Técnico Av Rovisco Pais 1049-001 Lisboa Portugal

Abstract. Hubel and Wiesel discovered that the mammalian visual cortex has a hierarchical structure. In this structure two cell types are arranged in successive layers. One recognizes shapes, the other allows shapes to be recognized even when they are shifted. These findings inspired Hierarchical Artificial Neural Networks. This kind of networks is suited for topological data, like images or sound. We try to capture the basic principles of hierarchical neural networks by analyzing a well known hierarchical network, Neocognitron. A simplified model, which we called Time Delay Neocognitron (TDN), is proposed based on the observation of the working principles of Neocognitron. The TDN can be understood as a sequence of filters which 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 yields a classification for the input pattern. Instead of a specifically crafted learning algorithm, the TDN separates two different learning needs: Information reduction were a clustering algorithm is suited (e.g. K-Means) and classification were a supervised classifier is suited (e.g. KNN). The performance of the proposed model is analyzed on handwriting recognition. The TDN achieved a similar performance relatively to Neocognitron.

Keywords: hierarchical network, neocognitron, neural network, pattern recognition, shift invariance, handwriting recognition, clustering

1 Introduction

Pattern Recognition deals with methods for classification of objects into categories [1, 2]. Classification is assigning an object to a class. The techniques for classification can be supervised or unsupervised.

In this text we’ll focus on classification with Artificial Neural Networks (ANNs) which have a hierarchical structure, neural networks for pattern recognition in general are reported in [3, 2]. Data on which hierarchical methods apply has some sort of topology, i.e. closer parts of pattern are more related to each other than farther parts. This relation between the closer parts can be a spatial relation, e.g. an image, or a temporal relation, e.g. a sound.

One approach to pattern classification with ANN is to use a hand-designed feature extractor to gather relevant information [4]. The gathered information is then feed into a trainable classifier, often a fully-connected multilayer neural network. This approach bottleneck is to construct the feature extractor.

Another approach is to feed the raw input and rely on the learning algorithms to make the first layers the feature extractor. This approach has some success, but when the input has some topology (e.g. temporal, spatial) the fully-connected ANNs ignore this information compromising partially their results, i.e. the input variables order could be switched to any other fixed arbitrary order and the results would remain the same. This approach has no built-in invariance to shifts and distortions of the inputs. To overcome this problem it’s necessary to preprocess the inputs, size normalizing and centering them. But preprocessing does not solve completely the problem, and a huge number of training patterns is necessary for the network to exhibit stronger invariance properties. A fully-connected network to classify even a small image has a huge number of connections, which make it susceptible to overfitting and computationally expensive [4].

2 Mammalian Visual System

Hubel and Wiesel made several important discoveries on the visual system [5–7]. They formulated an hypothesis of how the visual cortex works. This hypothesis was based on experiments with several mammals, including cats and monkeys. Many of the experiments consisted on presenting patterns to an eye and measuring the response with electrodes in some area of the visual cortex corresponding to a cell. According to this hypothesis the primary visual cortex (‘a brain area’) has mainly two types of cells, simple and complex. They found that the primary visual cortex has a hierarchical structure
where simple cells and complex cells are connected in successive layers. This hierarchy relation between simple and complex cells is repeated several times. They also found that the connections from the retina to the cortex are topographically organized, i.e., nearby cells in the cortex represented nearby regions in the visual field. The complexity of the cells responses increases as we move away from the input, i.e., the patterns to which cells respond become more difficult to express. The primary visual cortex has three or four stages [5].

2.1 Simple Cells

Simple cells react to patterns with a specific orientation and position. The cells have an excitatory and inhibitory region [5], normally cells have two or three regions. Typical cell’s regions can be seen on Fig. 1. Regions are rectangle shaped and in each cell all have the same orientation. Typical distributions of regions are one excitatory rectangular region surrounded by two inhibitory regions, or one inhibitory region side-by-side with an excitatory region. The responses of an S-cell to different stimulus can be seen on Fig. 2. The stimulus which generated the greatest response can be thought as the pattern that the cell has learned.

2.2 Complex Cells

Complex cells react to patterns with a specific orientation but allow their position to be shifted. Their receptive field is larger than the simple cells [5]. These cells respond in the same way to a pattern wherever it’s placed in their receptive field. This behavior makes them adaptable to position shifts in patterns and can be seen on Fig. 3.

![Fig. 1. Typical receptive fields for simple cells](image1.png)

![Fig. 2. Stimulus responses of a simple cell](image2.png)

![Fig. 3. Stimulus responses of a complex cell](image3.png)

3 Hierarchical Artificial Neural Networks

In most ANNs, in each layer every unit receives input from all the units of 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 has to be concerned only with a smaller and 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 [8]: the units in each layer have to concern themselves with simpler problems (only a part of the input), and these networks can work with much less units. Hierarchical Artificial Neural Networks like ANN in general can be supervised or unsupervised. These networks were inspired by biological findings. In both Neocognitron and Convolutional Neural Networks three key principles exist: local receptive fields, shared weights and subsampling. Local receptive fields and shared weights reduce computational costs by reducing the number of connections, neural units and parameters. Shared weights also improve generalization abilities by reducing the machine capacity [4]. Subsampling improves shift and distortion invariance capabilities [4].
3.1 Neocognitron

Neocognitron [9, 10] is a neural model mainly for vision, which can perform unsupervised learning. It’s an evolution of a previously proposed model, the cognitron [11]. Neocognitron was introduced as a model of the visual system and it’s based on the classical hypothesis [5] of Hubel and Wiesel. Neocognitron has two main purposes to recognize visual patterns and to be a biological model. There are several versions of the neocognitron, for the explanation of the model we’ll focus on the most recent versions and the more common characteristics. Neocognitron has good generalization capabilities, being able to learn a pattern by presenting to it only a few typical examples of that pattern. For the learning to be successful it’s not necessary to present all the deformed versions of patterns that might appear in the future [12].

Network architecture The neocognitron has three key architectural principles: local receptive fields, shared weights and subsampling. The neocognitron has two main types of cells: S-cells which resemble simple cells, and C-cells which resemble complex cells. It’s predecessor the cognitron, has only simple cells. A layer is a set of cells of the same type. A stage is a sequence of two different type layers, where the first is a S-cell layer and the later a C-cell layer. Networks can have one or more stages, e.g. the network on Fig. 4 has three stages. Cells in higher stages (closer to the output) tend to have larger receptive fields and to be more insensitive to the position of the stimulus (the input for example is the first layer’s stimulus). In some of the more recent versions of the neocognitron, there’s also a contrast extraction layer between the input layer and the first S-cell Layer.

The network outline can be seen in Fig. 4. Each square represents a matrix of cells which is called cell-plane. The connections entering each of the cells in a cell-plane are homogenous and topographically ordered [12]. The number of stages depends on the data to be classified. If the data has a high complexity the number of stages has to be larger [13]. The number of cell-planes in each stage also has to increase has the number of classes to classify increases. Other parameters that depend greatly on the data are receptive field’s size and how much they overlap. The choice of all this parameters can influence greatly the results achieved. and only some hints on how to choose them exist.

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 by 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 S-cell layer cell-planes is a feature map. They recognize a certain stimulus at different positions. Each cell-plane recognizes only one stimulus, i.e. they have a template to only which they tend to react. The number of cell-planes in each S-cell layer increases in higher stages. These cell-planes of S-cells have variable connections, allowing them to adapt to a certain stimulus by learning, i.e. each of these cell-planes specializes itself in one kind of stimulus.

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 S-cells are fixed and invariable which means they do not learn, a variation in which they learn is explained ahead. 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. C-cell layers blur the output of the S-cell layers. The density of cells in both S-cell and C-cell layers tends to decrease as the order of stages gets higher. In the output layer there’s a 1 × 1 matrix for each class the network classifies. The matrix with the higher response is the classification yielded by the network.

Learning Several training methods exist in different versions of the neocognitron. The neocognitron network can be trained by unsupervised or supervised learning. Another important distinction in training methods is between simultaneous and sequential construction [14].
In simultaneous construction, the learning of all the layers of the network progresses simultaneously. In sequential construction, each stage is trained separately from the ones closer to the input layer to the ones closer to the output layer. I.e. 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 contrary, sequential learning can finish learning fast but does not accept incremental learning. Sequential learning is more common in the more recent versions.

Most versions are trained by unsupervised learning. In supervised learning the ”teacher” points the position of the features to be extracted in the patterns [12]. The cells whose receptive fields coincide with position of the features become winners. The rest of the learning is similar to unsupervised learning.

The first layer of S-cells (U$_S$) 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. The intermediate layers have a variable number of cell-planes which depends on a threshold $\theta_S$ which is explained ahead.

**S-cells learning** S-cells compete with each other inside their competition area. When an input is presented to the layer there can be only one winner cell inside each competition area. Only winner cells get their input connections increased. The increment in the connection of a winner cell is proportional to the presynaptic activity. This behavior creates a specialization in each cell. They 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 [15]. The inhibitory connection from the v$_S$-cell is also increased making the cell acquire a selective response on the stimulus [14].

**Shift Invariance Analysis** It has been demonstrated in [16] that the neocognitron is not intrinsically shift invariant. The problem resides in the subsampling which is done in the C-cell planes. The neocognitron is subject to the following trade-off: it can be configured to be reasonably invariant to shifts by sacrificing its discriminatory power, or it can have sharp discrimination power by sacrificing shift invariance. This however, does not compromise its usefulness for pattern recognition applications. Most applications require that slightly distorted versions be recognized as the same object.

### 3.2 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. CNNs have three architectural ideas: local receptive fields, shared weights and often subsampling. These three architectural ideas are also present in the neocognitron. Each unit receives inputs from units of the previous layer in a small neighborhood area, this idea is similar to Hubel and Wiesel’s discoveries on the mammalian visual system [5].

The weights of several units are synchronized, making them all represent the same feature (e.g. a vertical line) in different positions of the layer, so that this feature can be detected anywhere in the layer. By making several units represent the same feature at different positions this set of units output can be seen as a feature map. This operation sequential implementation would be to scan the input image with a template, identifying in which positions the feature is present. The weight sharing improves the generalization ability by reducing the number of free parameters [4]. The idea is that the features can be detected independently of their position. Shifting the input won’t change the detected features. Only their position would be shifted like the input.

Each convolutional layer is followed by another layer that performs local averaging and subsampling. The local averaging and subsampling layer reduces the resolution of the feature map of its previous layer. These layers are responsible for reducing the importance of the position of the features and 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 one of complex cells. The number of feature maps increases from the input layer toward the output layer, as each feature meaning gets more complex. CNNs with fixed-size that share weights along a single temporal dimension are called time-delay neural networks (TDNNs), CNNs for composite object recognition like for example patterns representing words (the number of letters varies) are called space displacement neural networks (SDNNs), and consist in replicated CNNs.

The key difference between the neocognitron and CNNs is the learning. 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 minima problems however in CNNs this problem is slightly reduced, when compared with fully-connected
networks, since the number of free parameters is reduced by weight sharing.

3.3 Applications

There are several applications of neocognitron and convolutional networks. Neocognitron and CNNs have a similar scope of application. These applications have in common that the data has a topology.

**Handwritten Character Recognition** The neocognitron has been applied to handwritten text recognition [22] achieving a recognition rate of 98.6% on the ETL1 database with 3000 training patterns and 92.8% with 200 training patterns.

The training of the neocognitron for this task is done in both a supervised and unsupervised way. The first S-cell layer is trained in a supervised way. Training patterns, namely, straight lines of several orientations are presented [22]. Learning in the intermediate layers is unsupervised. Training patterns are fed into the network, and the output of the first layer previously trained is used as input for the training of the second layer. This also applies to the training of the third layer, which uses as input the output of the second layer. The network response to an example pattern can be seen on Fig. 5.

![Fig. 5. Neocognitron response to an example pattern (only C-cell layers are represented).](image)

The training of the fourth and last layer is supervised. When the network learns varieties of deformed training patterns, often more than one cell-plane is generated for each pattern in the forth S-cell layer. Each of these cell-planes has a label indicating which pattern it represents. Every time a training pattern is presented to the fourth S-cell layer competition occurs among the S-cells. If the label of the winner is the same as the label of the pattern then this cell learns in the same way as the previous layers. However, if the labels do not match, a new cell-plane is created, and the label of the pattern is assigned to it. During recognition the strongest output of the fourth S-cell layer determines the classification that the network yields. In this application of the neocognitron to handwritten character recognition, there is an additional layer between the input layer and the first layer, a contrast extraction layer.

There are applications of the neocognitron to related tasks, namely, recognition of handwritten musical notes is reported in [17] and recognition of handwritten Hangul is reported in [23]. The use of a Convolutional Network trained with the backpropagation algorithm, also to handwritten character recognition is reported in [18]. The comparison of several methods for this task is reported in [24–26].

3.4 Neocognitron vs. Convolutional Networks

Mainly two ANN were inspired by the classical hypothesis: Neocognitron and Convolutional Neural Networks. The networks have several similarities, namely, local receptive fields, shared weights and subsampling. Both these networks, unlike fully-connected ANNs, allow interpretation of the network operation, namely, how cells form templates of features, like edges or shapes. There’s still some domain knowledge that has to be incorporated regarding the network parameters, namely, receptive field’s size, subsampling ratio, number of features in some layers, and how much the receptive fields overlap with each other.

The main difference between neocognitron and CNNs, is the learning. CNNs are trained with the backpropagation algorithm, which is form of gradient descent. The neocognitron learning algorithm has been classified under the competitive learning paradigm.

4 Time Delay Neocognitron

We try to capture the basic principles of hierarchical neural networks. The simplified model is based on the observation of the working principles of Neocognitron. It is composed as well of two types of cells. A layer is a set of cells of the same type. The S-Layer resembles a layer of simple cells, and the C-Layer a layer of complex cells. A stage is a sequence of two different type layers, where the first is a S-Layer and the latter a C-Layer. The purpose of each S-Layer class is to detect the presence of particular pattern of activities.

The classification of our model begins with the binary input pattern of the S-Layer of the first stage.
The binary input pattern is tiled with a squared mask $M$ of size $i \times i$ in which a corresponding class is determined. The class is determined using the elements in each squared mask. Each sub-pattern in a mask is replaced by a number which indicates a corresponding class. Each mask corresponds to a $i \times i$ dimensional vector. The corresponding classes can be learned by a simple clustering algorithm like k-means, however if k-means is to be used the number of classes $k$ has to be determined by experiments. During the classification of the S-Layer the most similar class to each sub-pattern is determined. This is done by finding the cluster center (representing the class) which is more similar to the sub-pattern. By convention a “one” in the input binary pattern represents information and a “zero” the background of the binary pattern ‘no information’. If all elements of a mask are zero, the output class indicates the presence of the background, no classification needs to be performed.

It should be as well noted, that the mask may overlap. The output of a S-Layer is a pattern of integer numbers, each number for a corresponding class. The dimension of this output pattern depends on the size and the overlap of the mask.

Finally, the last layer corresponds to a classifier, which in our case is realized by a simple k-nearest neighbor.

The model operation incorporates then two different steps. First the information reduction which is performed in the stages. Second the mapping between the reduced information and the classes performed by the classifier. The architecture outline is shown in Fig. 6.

The choice of masks to ensure shift invariance and discrimination has to be done with care and is a difficult process. The choice leads either to an architecture which reasonably invariant to shifts at the expense of losing a certain amount of discrimination power, or to sharp discriminatory power by sacrificing shift invariance [16]. We solve this problem by a genetic optimization over the size of different masks.

**Example** Consider a network with a single stage which is followed by a recognition layer. The S-Layer mask is $9 \times 9$ and the C-Layer mask is $5 \times 5$. For a given position of the input more than one mask can be present, the size of the zone in which more than one mask can be present is referred to as overlap. In this example the overlap of the S-Layer mask is 7 and of the C-Layer mask is 4.

![Fig. 6. Architecture Outline.](image)

![Fig. 7. Example of an input pattern from E111 data set.](image)

![Fig. 8. S-Layer classes.](image)

**Classification** The classification works as follows: a pattern like the one shown in Fig. 7 is presented to the input layer which passes it to the S-Layer of the first stage. The S-Layer transforms the pattern by scanning
it with a single unit. In each position the single unit applies a mask to the input and the masked input is compared with the previously learned classes. In this example the previously learned classes are shown in Fig. 8. For each position the input is transformed into the most similar class according to the Euclidean distance. The most similar class is represented by an identifying number. In this example a number from 0 to 7, which corresponds to the classes of Fig. 8. The 0 represents the background by convention and the remaining represent seven other classes.

The single unit moves along the input, its movement and output on four horizontal contiguous positions is shown in Fig. 9. The output of the single unit for each position is the class which minimizes the Euclidean distance to the masked pattern. The output of the single unit for all positions of the input pattern shown in Fig. 7 is shown in Fig. 10. This output of the single unit for all positions, is the output of the S-Layer. This transformation keeps the relative positions of the patterns respecting the data topology. This operation results in the pattern being transformed to a different representation where only the previously learned classes are used to describe the pattern.

![Fig. 9. The class recognized by the S-Layer single-unit for different positions.](image)

The output of the S-Layer is then passed to the C-Layer which blurs it. The C-Layer purpose is allowing positional shifts in the input. It does this by removing the positional information of all classes inside the mask. Using the S-Layer output of Fig. 10, consider that the C-Layer mask is positioned like described in Fig. 12. Then the output of the C-Layer mask for this position will be the set of all present classes like is shown in Fig. 13. The background is discarded by the C-Layer.

![Fig. 10. S-Layer output cropped for pattern in Fig. 7](image)

The operation of this network during classification is graphically represented in Fig. 11.

**Learning** The learning of the proposed model is only performed in the S-Layers. The result of the learning is a set of classes for each S-Layer. For this example the learned classes of the only S-Layer are shown in Fig. 8. The classes are the result of a clustering algorithm which takes as input all positions of the mask of the S-Layer for all patterns of the train set. The masked patterns are the obtained in the same way as it was shown for the classification in Fig. 9. During learning the mask is used to pass the masked input into the clustering algorithm. The number of classes for the k-means clustering is previously determined by experiments. Another alternative for the initialization is to

---

1 The real S-Layer output is 37 x 37 however this was cropped to 18 x 18 by removing the background around the information on the center.

2 The images were generated with real data. In the mask overlap areas the images were added. Finally the images were normalized in 256 levels gray scale.
define threshold distance and then iterate over all data points. When the distance of any data point is above that threshold a new cluster is created. This kind of initialization is similar to the threshold concept that is used in Neocognitron during learning in S-cell layers.

5 Experiments

The experiments were conducted on the ETL1 data set of handwritten characters [27]. In the experiments only the digits which total 14456 patterns were used. The images were preprocessed using thrn [28], a binarization program based on the discriminant criterion [29].

Like the previous models, the proposed model is highly susceptible to the network parameters. Several networks with incorrect parameters for this data set performed as low as classifying randomly the input, i.e., picking a random number between 1 and 9.

To solve this problem, a genetic optimization algorithm was used to tune the model parameters for the ETL1 database.

5.1 Outline

The first experiment, Thinning-out, will analyze two populations with different thinning-out methods. One that performs the thinning-out from the S-Layer to the C-Layer (STOC) and the other from the C-Layer to the S-Layer (CTOS). This experiment will be used to choose a thinning-out for the remaining experiments.

A second experiment, Stages, will analyze three populations with different number of stages and a predetermined S-Layer: one with 3 stages (3SPD), other with 2 stages (2SPD), and finally one with 1 stage (1SPD). The results of the optimizations of this experiment are used as seeds for another experiment and allow a relative comparison between the two.

The third experiment, No predetermined S-Layer, analyzes three populations again with different number of stages but know without a predetermined S-Layer (3S, 2S, 1S). This experiment allows a comparison between networks with a different number of stages know without a predetermined S-Layer. It also allows a comparison between networks with or without a predetermined S-Layer.

The fourth and final experiment, Large Test Set (LS), analyzes a single population whose seed individual is the best individual so far from previous experiments. The optimization is performed with a larger test set which is randomly generated for each generation. The best individual of this population is finally tested on different size test sets.
5.2 Results

It’s interesting that the networks with only 1 stage (1S,LS) were the ones that achieved a better performance, 92.91% which compares slightly favorably with the results of Neocogniton [22] which achieved 92.8% for this data set also with 200 training patterns and a 3000 patterns test set. This result shows a slightly improved performance of the proposed model when compared to Neocogniton. The network parametrization is however quite different from the reported parametrization of Neocogniton [22], which has 4 stages (in Neocogniton the last stage has a purpose similar to TDN recognition layer) and the best parametrization found in the experiments has only one stage and a recognition layer (corresponding to a Neocogniton with 2 stages). The results are shown in Table 5.2.

<table>
<thead>
<tr>
<th>Architecture Name</th>
<th>Train Set Size</th>
<th>Test Set Size</th>
<th>Best Individual Average Test %</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOC</td>
<td>20</td>
<td>20</td>
<td>80.00%</td>
</tr>
<tr>
<td>CTOS</td>
<td>20</td>
<td>20</td>
<td>80.00%</td>
</tr>
<tr>
<td>3SP2</td>
<td>20C</td>
<td>20C</td>
<td>87.70%</td>
</tr>
<tr>
<td>2SP2</td>
<td>20C</td>
<td>20C</td>
<td>89.82%</td>
</tr>
<tr>
<td>1SP2</td>
<td>20C</td>
<td>20C</td>
<td>77.00%</td>
</tr>
<tr>
<td>3S</td>
<td>20C</td>
<td>20C</td>
<td>88.43%</td>
</tr>
<tr>
<td>2S</td>
<td>20C</td>
<td>20C</td>
<td>89.50%</td>
</tr>
<tr>
<td>1S</td>
<td>20C</td>
<td>20C</td>
<td>91.06%</td>
</tr>
<tr>
<td>1S</td>
<td>20C</td>
<td>10000</td>
<td>92.91%</td>
</tr>
</tbody>
</table>

Table 1. Average test classification rate of the best individual of each population for a given train and test size.

6 Conclusion

A new hierarchical network model was proposed by analyzing the key principles of Neocogniton and the Mammalian Visual System. The purpose of this model is to achieve the key principles in a more simpler way. A main contribution for the simplicity of the Time Delay Neocogniton is the learning. The proposed model learning can be performed by well know learning algorithms avoiding the additional complexity of a specifically designed learning algorithm.

The proposed model unlike Neocogniton has no cell-planes. In each S-Layer ‘simple layer’, a single unit scans the input in all different positions for all different classes. Only one class can be present at each position. This behaviour is equivalent to restricting Neocogniton so that it can only be one active cell for each position across all cell-planes and its activity for that cell to be constant. This behavior reflects an irreversible decision by each layer on what feature is present in each position, unlike in Neocogniton where a measure of similarity for each class in each position is passed along.

The proposed model’s C-Layer (complex layer) also does not have several cell-planes, like it’s S-Layer. The C-Layer receives as input a single matrix of features. A single unit scans the input and for a given receptive field, it keeps the information of which features were present but forgets their position inside the receptive field. The proposed model complex layer operation is very similar to Neocogniton’s and it’s key difference derives from the fact that the S-Layer does not have several cell-planes, so there are not several cell-plane outputs to blur, but just one.

The unsupervised learning method in Neocogniton was specifically developed for it and can be classified under the competitive learning paradigm [15]. 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 know algorithms can be used.

Two different learning needs exist. One in the stages, where the purpose is information reduction, and a clustering algorithm is fitted, e.g. K-Means. The other in the recognition layer is classification where a supervised classifier is more fitted, e.g. KNN, RBF, Back-Propagation NN.

In the stage’s clustering algorithm, each cluster is the equivalent to a specialization in a certain stimulus in Neocogniton, i.e. each cluster is a template like all cells in a cell-plane have a common template. The recognition layer can use any kind of classifier. The classifier maps the output of the last stage into classes the network classifies.

Hierarchical networks are highly sensible to parametrization, the proposed model tackles this problem by removing some of the parameters that need to be chosen in Neocogniton, like the learning and recognition thresholds of S-cell Layers. A genetic optimization was also used successfully to tune the model for the E1L1 data set of handwritten digits. This suggests that the model does not need to be manually tuned and also that this results could be replicated for Neocogniton even tough additional parameters need to be tuned.

The proposed model was applied to handwritten digit recognition to analyze it’s performance. Experiments were conducted using the E1L1 database of handwritten characters. The results of the experiments show a good performance achieving 92.91% test classification rate with just a 200 patterns training set.
result is similar to Neocognitron which achieved 92.8% test classification rate with a 200 pattern training set.

The network parametrization which performed better is however quite different from the one of Neocognitron [22], which has 4 stages (in Neocognitron the last stage has a purpose similar to TDN recognition layer). 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.

The TDN can be understood as sequence of filters which transform the input pattern into a space where the same classes of patterns are close. The output of the filters is then passed to a simple classifier which yields a classification for the input pattern. The conducted experiences show a similar performance of TDN relatively to Neocognitron. This shows that TDN performs the same task as the Neocognitron, even tough it has a simplified architecture. The experiments results show that the TDN captures the essential behavior of Neocognitron in a simpler way.

References