Online Learning of Invariant Object Recognition in a Hierarchical Neural Network

Markus Lessmann and Rolf P. Würtz

Institut für Neuroinformatik, Ruhr-Universität, Bochum, Germany
{markus.lessmann,rolf.wuertz}@ini.rub.de

Abstract. We propose the *Temporal Correlation Net (TCN)* as an object recognition system implementing three basic principles: forming temporal groups of features, learning in a hierarchical structure and using feedback for predicting future input. It is an improvement of the Temporal Correlation Graph and shows improved performance on standard datasets like ETH80 and COIL100. In contrast to its predecessor it can be trained online on all levels at once instead of having to be trained in batch mode level per level. Training images are presented in a temporal order showing objects undergoing specific transformations in viewing conditions the system is supposed to become invariant to. Computation time and memory demands are low because of a sparse connectivity structure resulting from the learning algorithm and efficient handling of neural activities.

1 Introduction

Visual processing is one of the brain functions most mimicked in artificial systems. The aims of the different modeling approaches reach from theories about the brain's computational processes during analysis of the environment to practical applications like surveillance, driver assistance, or control of industrial production processes. Whereas 20 years ago both purposes were pursued using very different techniques in the last 10 years several systems have been developed that on the one hand stick to important aspects known about the neural architecture in the brain and how information processing could be executed within the given constraints of this architecture and properties of biological neurons and on the other hand, thanks to increasing calculating capacities of nowadays computers, become applicable to big standard test databases and hence interesting for practical applications listed above. Examples include HMAX [1], HTM [2], and VisNetL [3]. Three basic ideas each underly some of these systems:

1. Learning of temporal sequences for creating invariance to transformations contained in the training data.
2. Learning in a hierarchical structure such that invariance increases gradually from level to level. Additionally lower level knowledge can be reused in higher level contexts and thereby makes memory usage efficient.
3. Prediction of future signals for disambiguation of noisy input by feedback.

Here, we present a novel system for invariant object recognition implementing these basic ideas. The layout is the following: In the next chapter we present our system in detail. Chapter 3 describes how learning works in the network. Results of experiments are presented in chapter 4. Chapter 5 closes the article with an outlook on future work.

2 Our System

The system presented here is called *Temporal Correlation Net (TCN)* and is an improvement of the *Temporal Correlation Graph (TCG)* from [4, 5]. It shares with its predecessor the architecture and handling and computation of neural activities. What has changed is that the new system learns using biologically plausible learning rules, namely a normal associative Hebbian learning rule for training of neurons representing spatial patterns and the trace rule for learning of temporal groups of features. This permits to train the network online rather than in batch mode as the older one. This should give the system the capability to get taught new object categories after initial training has been finished.

The system is a multilayer neural network consisting of three different levels each made up of two sublayers. Figure 1 shows the general structure of the network. The lower layer contains neurons responding to spatial patterns in the input. On the lowest level spatial neurons are representatives of so called parquet graphs [6], visual features built up of wavelet responses. A codebook of those found in the training set is learned using vector quantization. Each entry then gets associated a spatial neuron. On higher levels spatial neurons represent certain configurations of temporal groups simultaneously active at adjacent positions in the preceding sublayer. These temporal neurons represent groupings in which spatial input patterns at the same position of the network appear often close in time. This is important for creating invariance to transformations in viewing conditions. If, e.g., invariance to rotation in depth shall be learned the system is presented different images of an object rotating around a certain axis. At one image position the same object part might be observed from different viewing angles (for example left profile, front view and right profile of a head). A neuron tuned to a group of the corresponding visual patterns will respond to any of these views and hence will represent the head at least partly invariant to viewing angle. This scheme is repeated on the next levels of the hierarchy. First spatial patterns of adjacent lower level temporal patterns are learned and then the resulting patterns are temporally grouped. Thus, the partly invariant presentation of the head may be combined with a (partly) invariant representation of other body parts. Connection weights to spatial and temporal neurons are learned unsupervised using neural learning rules. Top level temporal neurons are representatives of complete object categories. The associations between top level temporal activities and object categories are learned supervised by associating vectors of activities averaged over training images of the same category with a category name. The recognized category can then be read out using dot product decoding.

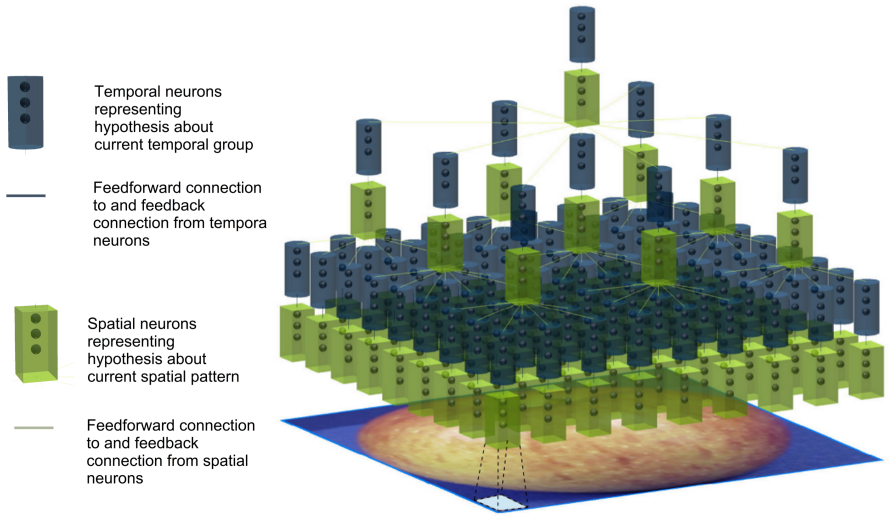

Fig. 1. Visualization of the network architecture. Connections of nodes represent possible synaptic connections between all neurons in one node and all in the other.

The system has to handle a very large number of neurons (potentially 10^5 or even 10^6 on some levels). For keeping computations tractable only non-zero weights are stored and weights of newly created neurons are initialized very sparsely. Additionally, neurons are simulated dynamically by just storing an index and a real activity value in specially designed containers (called nodes). Neurons can be addressed via their index using hashing techniques for fast access. In addition neurons can be sorted according to activity and those with weak activation can be deleted. On the lowest level 9×9 nodes are placed in a regular grid on the input image (in both sublayers). On higher levels convergence takes place with a common factor C of 3, i.e., a higher level spatial node receives input from 3×3 adjacent temporal nodes of the preceding layer. Between two sublayers the mapping is one to one, each spatial node yields the input to one temporal node of the same level. This scheme leads to 3×3 nodes in the (sublayers of the) middle level and 1 at the top level.

3 Inference and Learning

Learning and inference are not independent since training requires inference on the already trained network levels.

3.1 Inference

Inference is done by computing activities of all neurons for a given input image and then reading out the activities of temporal neurons at the top level.

These identify the recognized object category. Calculation of activities is done from bottom to top level for one node position after another. This can also be done in parallel for node positions on the same level. Activities are calculated and stored in a temporary memory. If their calculation is completed they are transferred to a bigger container storing activities of the last T time steps, where they are used for learning and for feedback calculation.

Computing spatial feedforward input: On level 0 graph feasts are extracted, their nearest neighbor in the codebook is looked up and the corresponding neuron is activated with the similarity. During learning a graph feast is added to the codebook if the similarity is below a threshold ϑ_Q and a new neuron is created. On higher levels neurons compute their feedforward activation as neurons in an ordinary MLP by multiplying the activity of input neurons with the corresponding connection weight and adding these terms up.

Inhibiting spatial neurons: Next the amount of active neurons at each node position is reduced by deleting all but the K most active neurons from the hash map. During learning K is 1, during recall it is usually higher. This step can be seen as application of inhibition between neurons at the same node position.

Computing spatial feedback input: The next step is to add feedback input to the remaining active neurons coming from temporal neurons which have been active on the T previous images.

Application of the activation function: The neuron activities are processed by a tanh nonlinearity. This activation function provides besides inhibition the second non-linearity in the system enabling it to robust classification. Additionally it prevents activity values from growing to infinity, which could happen because of feedback connections. For using more of its non-linear regime neuron activities (which are between 0 and 1 because of normalization of weights) are multiplied with π before.

Transfer into activity stack: Then activities are written into an activity stack of the last T images. The current activities are written to position 0 and the older ones are shifted one position, the last entry gets deleted. During learning activity in the stack is deleted when a new object category is presented for preventing the system from learning transitions between different categories.

Computations for temporal neurons: Now the same kind of calculations are done for neurons representing temporal groups. At first temporal neurons collect their feedforward input, then inhibition is applied and feedback is given to remaining temporal neurons (this time only from higher level spatial neurons active on the last image). At last the activation function is used again and activities are moved to the activity stack.

3.2 Learning

After inference learning is done for each training image on all levels and sublayers at once. It is executed globally on each level meaning that neurons, which can be added during learning, can become active independently at each node position of a level and their connection weights can be trained on any node position.

On level 0 the codebook and spatial neurons are learned using vector quantization. On each other sublayer (on level l) a new neuron gets created if the current maximum activity at a node position is below a threshold ϑ_S^l for spatial or ϑ_T^l for temporal neurons. Incoming weights of new neurons are tuned to currently active input neurons thus that they would have been maximally active for the current input. They are randomly connected to higher level neurons active on recent time steps. All incoming weights of a temporal neuron are normalized to a Euclidean norm of 1.0. For spatial neurons all incoming weights from the same input node are normalized to an Euclidean norm of $1/C$. Weights whose absolute value falls below a threshold ϑ_L get deleted.

Spatial neurons need to learn which temporal neurons at adjacent input positions are active at the same time. This is just an associative learning task which can be executed well by a Hebbian learning rule:

$$\Delta W_{i,j}^{l,\mathrm{t};l+1,\mathrm{s}}(v) = \alpha \cdot \mathbf{n}_i^{\,l,t}(v) \cdot \mathbf{n}_j^{\,l+1,s} \tag{1}$$

α is a learning factor, $\mathbf{n}_i^{\,l,t}$ is the activity of temporal neuron i on level l at the currently considered input position v, $\mathbf{n}_j^{\,l+1,s}$ that of spatial neuron j on level $l+1$ and $W_{i,j}^{l,\mathrm{t};l+1,\mathrm{s}}$ the connection between both of them. Temporal neurons have to learn temporal groupings of spatial input patterns. This can be achieved using the trace rule, an associative Hebbian learning rule based on the trace of a temporal neuron's activity at time τ:

$$\overline{\mathbf{n}}_j^{\,l,t^\tau} = (1-\eta) \cdot \mathbf{n}_j^{\,l,t^\tau} + \eta \cdot \overline{\mathbf{n}}_j^{\,l,t^{\tau-1}} \tag{2}$$

It provides a memory of the neurons past activities and is high not only if the neuron was activated strongly on the current but also on past images. Parameter η determines the weighting between current and past input. The trace value enables the following learning rule to adjust connection weights thus that spatial neurons which are activated frequently on consecutive time steps all excite the same temporal neuron. The trace rule itself reads as:

$$\Delta W_{i,j}^{l,\mathrm{s};l,\mathrm{t}} = \alpha \cdot \overline{\mathbf{n}}_j^{\,l,t^\tau} \cdot \mathbf{n}_i^{\,l,s^\tau} \tag{3}$$

Here $\mathbf{n}_i^{\,l,s^\tau}$ is the activity of spatial neuron i on level l,$\overline{\mathbf{n}}_j^{\,l,t^\tau}$ the already introduced trace value and $W_{i,j}^{l,\mathrm{s};l,\mathrm{t}}$ the weight between the considered neurons.

4 Experiments

The system was tested on the ETH80 [7] (in the "cropped close perimg" version) containing images of 8 different categories and the COIL100 [8], consisting of images of 100 different objects, each being its own category. In both databases a black background was used instead of additional segmentation information. All tests used one-fold cross validation: the number of views per object was split into two sets, the first was used for training, the rest for testing.

All images had a size of 128×128 pixels. A 3-level network was used as shown in figure 1. Nodes on the lowest level were placed on the input images with a spacing of 14 pixels and an offset of 7. Since the network learns temporal groups the training images had to be in a meaningful order. Therefore, views were sorted according to their great-circle distance on the viewing hemisphere. For fifty-fifty-partitioning every other view of the order was taken for training and the rest for testing, for other split-ups every third or fourth, etc.

First the most relevant parameters are listed for a better overview:

K: determines how many neurons are kept active in each node for further computations.
η: determines how strong past activities are considered for temporal traces.
α: the learning rate of the neural learning rules.
$T_{tr/te}$: determines for how many past images activities are kept in memory for neuron creation during learning and feedback calculation during recognition.
T_R: this parameter determines for how many past time steps trace values are kept in memory for neuron creation during learning.
ϑ_Q: threshold used during learning of the codebook using vector quantization.
ϑ_S^l: threshold for creation of spatial neurons.
ϑ_T^l: threshold for creation of temporal neurons.
N_E: number of training epochs before adaptation of neuron numbers.
ϑ_L: weights with absolute value below this threshold get deleted.

Preliminary parameter tests on ETH80 resulted in the following optimal values: $\vartheta_Q = 0.92$, $T_{tr} = 65$, $T_R = 50$, $\eta = 0.3$, $\alpha = 0.00175$, $\vartheta_S^l = 0.9$, $\vartheta_T^l = 0.0$, $N_E = 3$ and $\vartheta_L = 10^{-5}$. These gave a recognition rate of 99.69%. For COIL100 most parameters were kept except T_{tr} and T_R which were set to 25, ϑ_Q to 1.00 and ϑ_S^l to 0.95, yielding a recognition rate of 99.94%. T_{te} and K were optimized for each single test.

The following test shows the generalization capabilities of the system for ETH80 and COIL100. The system was trained on different percentages of all images in the database and recognition on the remaining images was done using the two optimal parameter sets.

The results in table 1 demonstrate that the new system outperforms the old on the ETH80 dataset considerably. Even for only 2.50% of training data rates of almost 64.00% are reached which is quiet good. They also don't decrease much for more than 50.00% training data as they do for TCG since parameters were optimized for this training amount. For COIL100 also some improvements could be reached in most of the tests although they are not as remarkable as those on ETH80.

Table 2 shows results of tests on COIL100 with further percentages. We compare the TCN with TCG, the system of Westphal [6] (which uses the same features as we) and Linde and Lindberg [9, 10], which have the best results that we could find. They create a high-dimensional histogram for each image and classify these using SVMs. It has to be noted that results of Westphal have been obtained using 5-fold cross validation in contrast to the others.

Table 1. Tests for generalization over viewing angle. Given are the percentages of images of each dataset the system was asked to use for training, the number of images that were actually used and the recognition rates of both systems reached on the remaining images. For TCN and TCG performance on training set was always 100%. Results of TCG differ from [4] due to further parameter tests. On the left results for ETH80 on basic category level (apple, car, cow and so forth) and on the right results for COIL100 on name level.

% requ.	# obt.	TCN	TCG	# obt.	TCN	TCG
	ETH80			COIL100		
2.50	160	**63.65**	27.44	200	**61.06**	58.00
5.00	240	**68.32**	25.95	400	**76.57**	74.91
10.00	400	**85.69**	31.70	800	**93.95**	88.89
20.00	720	**98.16**	87.89	1500	**97.56**	97.46
30.00	1040	**99.06**	90.98	2200	**99.58**	99.18
40.00	1360	**99.11**	98.12	2900	**99.81**	99.63
50.00	1680	**99.69**	99.06	3600	99.94	**100.00**
60.00	2000	**99.53**	99.30	4300	**100.00**	99.93
70.00	2320	**99.79**	99.17	5100	**99.95**	99.86
80.00	2640	**99.53**	95.47	5800	**100.00**	99.93
90.00	2960	**100.00**	96.25	6500	**100.00**	99.71

The first column shows the distance in viewing angle of two consecutive images in the training set. The other columns show the obtained recognition rates on test set, on training set was always 100.00% for TCN and TCG.

The results show that the TCN performs for most training sets better than TCG. In total both systems outperform the approach of Westphal but cannot compete with the system of Linde/Lindeberg for very sparse training sets. Whereas they reach recognition rates of over 97% for 90° distance between training images the TCN drops to 78.15%. However, it needs to be considered that they include color information, which is not used in our systems.

Further tests were conducted on the ALOI1000 [11] to show the capability to handle large databases. All images were rescaled to 128×128 pixel, and a fifty-fifty-partitioning test was performed on both the viewing angle subset and the subset with varying illumination directions. Recognition rates (test/train) were 99.59%/99.88% and 99.91%/99.97%, respectively, which is very good and shows that the TCN can handle such a large number of object categories.

5 Conclusion

We have presented a powerful object recognition system that improves in several ways on an older predecessor. It generalizes very well over different views of the same object on standard datasets and is capable of online learning. In the future we want to test if it can be trained with different categories incrementally.

Table 2. Tests for generalization over viewpoints on COIL100 on name level.

viewpoint diff.	TCN	TCG	Westphal	Linde/Lindeberg
10°	99.94	**100.00**	99.68	**100.00**
20°	99.02	98.91	97.97	**99.96**
30°	97.08	95.87	92.93	**100.00**
40°	**96.63**	93.05	88.45	—
45°	91.39	88.61	—	**99.37**
50°	93.95	**88.89**	83.20	—
60°	80.98	79.02	76.61	**99.00**
70°	**83.33**	81.44	75.79	—
80°	**81.10**	77.28	72.39	—
90°	72.16	70.38	65.63	**97.13**

Acknowledgments: The authors gratefully acknowledge funding from the DFG in the priority program "Organic Computing" (WU 314/5-3) and from the land of Northrhine-Westphalia in the project *MoGES*, which is co-financed by the EFRE program from the European Commission.

References

1. Riesenhuber, M., Poggio, T.: Hierarchical models of object recognition in cortex. Nature Neuroscience **2**(11) (1999) 1019–1025
2. George, D.: How the brain might work: a hierarchical and temporal model for learning and recognition. PhD thesis, Stanford University (2008)
3. Rolls, E.T.: Invariant visual object and face recognition: Neural and computational bases, and a model, VisNet. Frontiers in Computational Neuroscience (2012)
4. Lessmann, M., Würtz, R.P.: Learning of invariant object recognition in a hierarchical network. In Hammer, B., Villmann, T., eds.: Proceedings of New Challenges in Neural Computation, Graz. Number 03/2012 in Machine Learning Reports (2012) 104–112
5. Lessmann, M., Würtz, R.P.: Learning of invariant object recognition in a hierarchical network. Neural Networks (2014) In press.
6. Westphal, G., Würtz, R.P.: Combining feature- and correspondence-based methods for visual object recognition. Neural Computation **21**(7) (2009) 1952–1989
7. Leibe, B., Schiele, B.: Analyzing appearance and contour based methods for object categorization. In: Proc. CVPR. Volume 2. (2003) 409–415
8. Nene, S., Nayar, S., Murase, H.: Columbia Object Image Library (COIL-100). Technical Report CUCS-006-96, Columbia University (1996)
9. Linde, O., Lindeberg, T.: Object recognition using composed receptive field histograms of higher dimensionality. In: Proc. ICPR. (2004) 1–6
10. Linde, O., Lindeberg, T.: Composed complex-cue histograms: An investigation of the information content in receptive field based image descriptors for object recognition. CVIU **116**(4) (2012) 538–560
11. Geusebroek, J., Burghouts, G., Smeulders, A.: The Amsterdam Library of Object Images. International Journal of Computer Vision **61** (2005) 103–112