Volume 2, Issue 9, September 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Additive Studying in Hierarchical Neural Networks for Object Recognition

[1]Rajesh Babu, [2]Dr. Jitendra Kumar
[1]*Rakshpal Bahadur Management Institute, Bareilly, UP, India*
[2]*Drownacharya Engineering College, Gurgaon, Haryana, India*

Abstract: Robots that execute non-trivial projects in real-world surrounds are probably to find objects they have not seen before. Thus the power to learn new objects is a necessity skill for advanced mobile service robots. The model demonstrated in this paper has the power to learn new objects it is shown during run time. This improves the adaptability of the approach and thus enables the robot to adjust to new positions. The purpose is to assert whether and how well hierarchical neural networks are suited for this intention. The experiments conveyed to answer this question showed that the suggested incremental learning approach is applicable for hierarchical neural networks and furnishes satisfactory classification results.

Keywords: Object recognition, hierarchical neural networks, incremental learning.

1. INTRODUCTION

When doing tasks in complex environments it is likely that a robot is faced with objects it has not seen before and consequently cannot identify. This builds the ability to learn new objects during run time an important capability for advanced mobile service robots. The robot is then able to incrementally learn novel objects and thereby increase knowledge of its environment and adapt to new positions. Without the ability to additive learns new objects the robot could only conduct with predetermined objects learnt offline. All other objectives could only be classified as unknown objects. In real world environments which are fairly complex and subject to numerous changes solely being equal to of coping with previously learnt objects might not be sufficient. The presented approach is an extension of our previous work (Knoblauch et al., 2004 [7]) which enables the model to deal with unknown objects by incrementally learning to categories these objects.

2. HIERARCHICAL OBJECT RECOGNITION

The visual object recognition is implemented as a two stage process (Fay et al.,) where low resolution features such as color and shape information are used to localise objects of potential interest and en-suing high resolution features such as edges, corners or T-junctions are extracted to analyse the part of the image containing the detected object in greater detail. These characteristics are used in a trained neural network for object recognition. For the application at hand orientation histograms (Freeman and Roth, 1995[5]) (Coppola et al., 1998) summarizing all orientations (directions of edges represented by the gradient) within the region of interest are used as features for the object recognition.

2.1. Hierarchical Neural Networks

If neural networks are used for object recognition an object is constituted by a number of features, which form a d dimensional feature vector x within the feature space X ~ I R. A classifier therefore realizes a mapping from feature space X to a finite set of classes $C = \{1, 2, \ldots, l\}$. A neural network is aimed to perform a classification task using supervised learning algorithms. A set of training examples $S := \{(x, t), \mu = 1, \ldots, M\}$ is presented to the network. The training set consists of M feature vectors x µ 2 I R d each labelled with a class membership t µ 2 C. During the training phase the network parameters are adapted to approximate this mapping as accurately as possible. In the classification phase unlabelled data x µ 2 I R d are demonstrated to the trained network. The network output $K(x) = c$ 2 C is interpreted as an approximation of the class corresponding to the input vector x.

The object classification is performed by a hierarchical neural network (Fay et al.,). It comprises of several simple neural networks that are aggregated in a tree, i.e. the nodes within the hierarchy represent individual neural classifiers. For the application at hand RBF networks were used as neural words. The basic idea of hierarchical neural networks is the hierarchical decomposition of a complex classification problem into several less complex ones. This yields hierarchical class grouping whereby the decision process is split into multiple steps exploiting rough to detailed classification. The hierarchy emerges from recursive partitioning of the original set of classes C into several disjoint subsets C i until subsets consisting of single classes result. C i is the subset of classes to be classified by node i , where i is a re-cursively composed index reflecting the path from the root node to node i . The subset C i of node i is decomposed into s i disjoint subsets , where

C i , j ~ C i , C i = [i-^{1}
j = 0 C i , j and C i , j \ C i , k = ; .

The total set of classes C 0 = C is assigned to the root node. Consequently nodes at higher levels of the hierarchy typically classify between larger sub-sets of classes whereas nodes at the lowest level discriminate between single classes. This divide-andconquer strategy yields several simple classifiers, that are more easily manageable than one extensive classifier. These simple classifiers can be amended much more easily to the decomposed simple classification tasks than one classifier could be adapted to the original complex classification task. Furthermore different feature types X i are used within the hierarchy. For each classification task the feature type that allows for the best discrimination is chosen. An example of such a hierarchy is shown in figure 1. The feature types X i and X j used for node i and j can be identical, i.e. both can use the same feature type.

2.2. Hierarchy Generation

The hierarchy is generated by unsupervised k-means clustering. In order to decompose the set of classes C i assigned to one node i into s i disjoint subsets a k-means clustering is performed with all data points { x µ 2 X i |t 2 C i } belonging to these classes. De-pending on the distribution of the classes across the k-means clusters s i disjoint subsets are formed. One successor node j corresponds to each subset. For each successor node j again a k-means clustering is performed to further decompose the corresponding subset .The k-means clustering is performed for each feature type. The different clusterings are evaluated and the clusterings which group data according to their class labels are preferred. Since the k-means algorithm depends on the initialisation of the clusters, k-means clustering is performed several times per feature type. In this study the number of k-means clustering per feature type was 10.

The number of clusters k must be at least the number of successor nodes or the number of subsets s respectively but can also exceed this number. If the number of clusters is higher than the number of successor nodes, several clusters are grouped together so that the number of groups equals the number of successor nodes. For reasons of simplicity in the following only the case where the number of clusters k equals the number of successor nodes s is considered. The valuation function prefers clusterings that group data according to their class labels. Clusterings where data are uniformly distributed across clusters notwithstanding their class labels receive low ratings. Furthermore clusterings are preferred which evenly di-vide the classes. Thus the valuation function rewards unambiguity regarding the class affiliation of the data assigned to a prototype as well as uniform distribution regarding the number of data points assigned to each prototype.

To generate the hierarchy at first the set of all classes is assigned to the root node. Starting with a clustering on the complete data set the set of classes is divided into subsets. Each subset is assigned to a successor node of the root node. Now the decomposition of the subsets is continued until no further decomposition is possible or until the decomposition does not lead to a new division. An example of a classification hierarchy is shown in figure 1.

2.3. Training and Classification

The hierarchy is trained by separately training the individual classifiers with the data { x µ 2 X i |t 2 C i } that belong to the subsets of classes assigned to each classifier. For the training the respective feature type X i identified during the hierarchy generation phase is used. The data will be relabelled so that all data points of the classes belonging to one subset have the same label j , i.e. ˜ t µ = j , x 2 X i , t µ 2 C i , j . The number of input nodes of the single classifiers is defined by the dimension d i of the respective feature type X i assigned to the corresponding node i . The number of output nodes equals the number of successor nodes s i . The classifiers are trained using supervised learning algorithms. The classifiers within the hierarchy can be trained independently, i.e. all classifiers can be trained in parallel.

Within the hierarchy different types of classifiers can be used. Examples of classifiers would be radial basis function (RBF) networks, linear vector quantisation classifiers (Simon et al., 2002) or support vec-

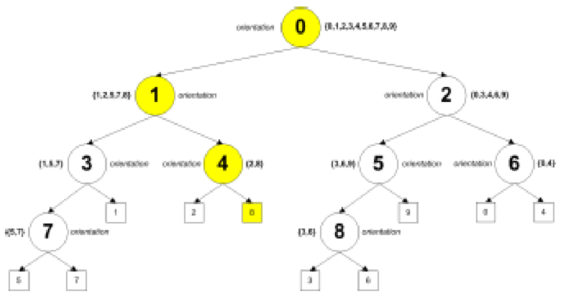


Figure 1: Classifier hierarchy generated for the classification of 10 classes of the COIL20 data set using orientation histograms as feature type.

Each node within the hierarchy represents a neural network which is used as a classifier. The end nodes represent classes. To each node a feature type and a set of classes is as-signed. The corresponding neural network uses the assigned feature type to discriminate between the as-signed classes. The highlighted path shows the nodes activated during the classification of a sample that is classified as class 8. tor machines (Schwenker, 2001[11]). We chose RBF net-works as classifiers. They were trained with a three phase learning algorithm (Schwenker et al., 2001[11]).

The classification result is obtained similar to the retrieval process in a decision tree (Duda et al., 2001[3]). Starting with the root node the respective feature vector of the object to be classified is presented to the trained classifier. By means of the classification out-put the next classifier to categorise the data point is determined, i.e. the classifier j ~ corresponding to the highest output value o (j ~) is chosen such that

j ~ = a r g m a x j = 1 . . s i(o (j)) . Thus a path through the

hierarchy from the root node to an end node is obtained which not only represents the class of the object but also the subsets of classes to which the object belongs. Hence the data point is not presented to all classifiers within the hierarchy and the hierarchical decomposition of the classification problem yields additional intermediate information.

If only intermediate results are of interest it might not be necessary to evaluate the complete path. In order to solve a task it might be sufficient to know whether the object to be recognised belongs to a set of classes and the knowledge of the specific category of the object might not add any value. If the task for example is to grasp a cup, it is not necessary to distinguish between red and green cups. Moreover, when looking for a specific object it might in some cases not be necessary to retrieve the final classification result if a decision at a higher level of the hierarchy already excludes this object.

2.4. Incremental Learning of New Classes

While performing tasks a robot is likely to encounter unfamiliar objects. Hence the ability to learn new objects plays an important role. Given the situation that an object is in the robot's visual field and it is told the name of the new object by an instructor, the robot might look at the object from different points of view and thereby generate a few data sam-

p l e s S˜ := { (x , c) , ν = 1 , . . . , N } o f t h e n e w c l a s s

c ˜ 2/ C . So compared to the already known classes C the number of samples N of the new class c ˜ is appreciably lower. This scenario forms the basis for the incremental learning approach developed here. In the overall system the online learning is triggered by instruction sentences starting with ”this is” followed by an object name.

In order to quickly achieve results the learning is performed in two stages. In the first stage fast but less sophisticated methods are used to obtain initial results, i.e. the novel objects are learnt but the recognition rate might be weak. In this first stage the recognition rate can be improved by using a similar method to retrain the new object with additional data gained by taking different views of the object. In a second stage more complex algorithms are used to adapt the system and to further improve the classification results. This retraining is more advanced but also more time consuming.

At first it is necessary to identify whether the presented object is already known or not. This is accomplished by presenting the new data to the trained classifier and taking the strength of the classifier response into account. Thereby a strong response is considered as an unambiguous decision and weak responses indicate a dubious decision which could be evoked by unknown classes or if the object to classify bears resemblance to more than one class. The thresholds for this are derived from the classifier responses when testing the known data. If the new data is unambiguously classified as one class c without exception it is assumed that the object is already known and belongs to class c . Otherwise the object is regarded as a hitherto unidentified object. If an object is identified as unfamiliar it is learnt by fitting it into the hierarchy and if necessary retraining the affected nodes. The new class c ˜ 2/ C associated with the unknown object is inserted as a new leaf. The position of the new leaf is determined by classifying all new data and evaluating the corresponding paths through the hierarchy. The leaf will be inserted where the paths start to diverge. As complete identicalness for all data cannot be presumed even at the root node since the network has not been trained with this data a certain variance needs to be considered. Otherwise the new leaf would in most instances be added at the root node. Therefore the common path is successively determined similar to retrieving the classification result. Starting with the root node all new samples are presented to the corresponding classifier. Depending on the classification results either the next node in the path is determined or the search is stopped and the new class is added as a new leaf at this node. If all new samples xν are assigned to one successor node j by classifier i node i is added to the common path without retraining and classifier j is the next classifier. If not all but a significant majority of the data points is assigned to the same successor node j this classifier is retrained using the samples of the known objects as well as the new samples and node i is then added to the common path. If there is no clear decision or if all successor nodes of node i are end nodes the new class is added as an additional end node to this node and the node is afterwards retrained. Because of the way the hierarchy is built only part of the hierarchical classifier needs to be amended. The rest of the hierarchy remains unchanged. Figure 2 depicts how a new class is inserted into the hierarchy by means of the proposed incremental learning approach.

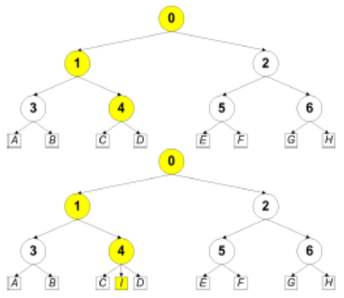


Figure 2: Example of the incremental learning of a new class I. At first the position where to insert the new class is determined.

Then the new class is added as a new leaf and the affected nodes are retrained. Here the new class is added to node 4. The classifiers 0 and 1 are retrained with the samples of the additional class I. One additional node is added to the output layer of classifier 4 and the classifier is then retrained. The retraining or incremental training of the classifiers is conducted by adding a new neuron to the hid-den layer and then retraining the output weights with the joint sample set of old and new samples. The centre of the new prototype is determined by the mean of all new samples. The width of the corresponding gaussian function is set to the mean distance of the new samples to the centre and the new output weights are learnt by calculating the pseudo-inverse (Penrose, 1955). A similar mechanism is applied when retraining al-ready learnt classes. The only differences are that no additional leaf is added and that the path through the hierarchy is already known. The single classifiers on this path are retrained if there are any incorrect or ambiguous decisions.

Scenarios for retraining an already learnt class could be when a class is only represented by few samples or the classification performance for this class is not satisfactory. The classification rate of novel classes is likely to be lower than the classification rate for the previously learnt classes. Thus a retraining of all affected nodes can yield improved classification results once sufficient additional samples for the novel class are available. Another reason could be that a new instance of a class with noticeably different characteristics has to be learnt.

The online learning phase is followed by an offline learning phase where more sophisticated learning algorithms such as three phase learning are used which will further improve the classification performance. All nodes to which the novel classes have been as-signed are newly trained. These algorithms would be too time consuming for usage during run time.

This incremental learning approach can also be used for non-hierarchical neural networks. However, here it is not necessary to determine the position were to insert the new leaf as non-hierarchical networks can be regarded as hierarchical networks consisting only of one node, but a retraining of the complete network takes place whereas only parts of the hierarchical net-work is amended.

3. RESULTS

By means of classification experiments the suit-ability of hierarchical neural networks for extension was examined. It should be verified whether new classes that are only represented by a few samples can be learnt sufficiently well in moderate time and whether it is possible to learn new classes without negatively affecting the classification performance of already learnt classes. The incremental learning approach has been evaluated using the Columbia Object Image Library (COIL20) (Nene et al., 1996) consisting of 20 objects. Therefore 10-times 10-fold cross-validation experiments have been conducted on the camera images of the COIL20 image library. Another approach for learning new objects would be a complete redesign of the classifier. This means that everything learnt so far is dropped and the classifier is rebuild and trained with the joined samples of the known and unknown objects. For the classifier training only simple training algorithms such as two-phase learning (Schwenker et al., 2001) can be used as sophisticated learning algorithms would be too time-consuming and hence not applicable for online learning. Compared to the incremental learning approach this method features longer training time as a complete rebuild of the hierarchy and training of all nodes within the hierarchy are required. The approach pro-posed has been compared to this method. In order to facilitate comparability of the results of the different approaches for all experiments the same division of the data for the cross-validation has been used.

For the experiments 10 classes of the 20 classes of the COIL20 data set were chosen to represent the familiar objects. The remaining 10 classes formed a pool of potentially new objects. To test the incremental learning approach proposed the first 10 classes were used to generate and train a hierarchy. To train the hierarchy sophisticated learning algorithms were used. Here the three-phase learning for RBF networks was used. The hierarchy structure was held constant for reasons of comparability while the hierarchy training was subject to cross-validation. This means the same hierarchy was trained 10×10 times resulting in 100 trained hierarchies. These trained hierarchies for the classification of the first 10 classes formed the basis for the incremental learning experiments. Each of the 10 new classes was then learnt incremental so that $10 \times 10 \times 10$ hierarchies for the classification of 11 classes emerged. For the incremental learning 10 samples of the unknown class were used compared to 64 to 65 samples (depending on the specific cross-validation run) of the other classes. In the test data set all classes are represented with the same number of samples.

With the same partition of the samples the alternative method which completely rebuilds the hierarchy was evaluated. Therefore 10 hierarchies for the classification of 11 classes each were generated. For each of the 10 hierarchies 10×10 cross validation experiments were conducted.

Both approaches showed essentially the same classification quality. The incremental learning approach had an average classification rate of $95.94 \pm 2.96\%$ and the average classification rate of the alternative method was $95.54 \pm 2.56\%$. Despite being less extensive the incremental learning approach achieved the same results. Figure 3 visualises these results as box plots.

The confusion matrix for the incremental learning experiments displayed in figure 4 shows that although being represented by a significantly lower number of samples the classification rates of the new classes is equal to the classification rates of the primarily learnt classes.

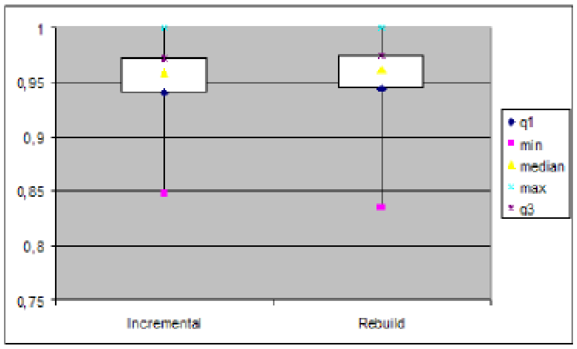
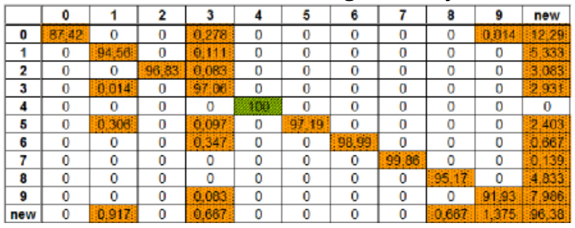


Figure 3: Box plot of the classification results of the two alternative approaches for learning new classes. The classification results do not differ significantly.



	0	1	2	3	4	5	6	7	8	9	new
0	87,42	0	0	0,278	0	0	0	0	0	0,014	12,29
1	0	94,56	0	0,111	0	0	0	0	0	0	5,333
2	0	0	96,83	0,083	0	0	0	0	0	0	3,083
3	0	0,014	0	97,06	0	0	0	0	0	0	2,931
4	0	0	0	0	100	0	0	0	0	0	0
5	0	0,306	0	0,097	0	97,19	0	0	0	0	2,403
6	0	0	0	0,347	0	0	98,99	0	0	0	0,667
7	0	0	0	0	0	0	0	99,86	0	0	0,139
8	0	0	0	0	0	0	0	0	95,17	0	4,833
9	0	0	0	0,083	0	0	0	0	0	91,93	7,986
new	0	0,917	0	0,667	0	0	0	0	0,667	1,375	96,38

Figure 4: Confusion matrix for the experiments utilising incremental learning.

Having a look at the positions within the hierarchy were the new classes were added, it could be found that in the most instances a class was mainly assigned to one node. If this is not the case then at least the class was added to nodes lying on the same path of the hierarchy as can be observed for class 13. The only exception to this is class 11, which is distributed over different pathes of the hierarchy. It can also be observed that in the majority of cases the classes were added to nodes in deeper levels of the hierarchy. No classes were added to the root node. Figure 5 illustrates for each new class how often it was added to which node of the hierarchy shown in figure 1.

These results show that the new classes are not added arbitrarily to the hierarchy but for each class preferred positions emerge. The noticeable frequency of node 8 is likely to be a characteristic of the COIL20 data set.

4. RELATED WORK

An example for other incremental learning approaches are ART networks. ART networks (Gross-berg, 2000) (Carpenter and Grossberg, 2002) allow for online learning of evolving data sets. If a presented sample is similar enough to an learnt prototype this prototype is adjusted to the sample,

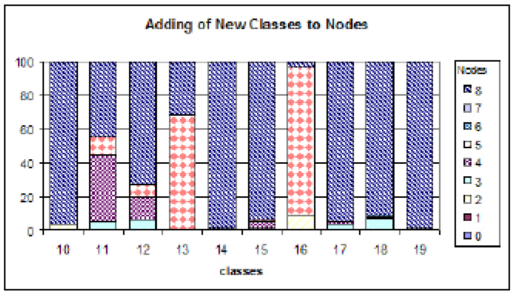


Figure 5: Adding of new classes to nodes. For each of the 10 ×10 cross-validation experiments conducted for each of the new classes 10 to 19 a leaf was added to different nodes of the classification hierarchy. For each new class is shown in percent to which node the corresponding leaf was added.

Otherwise a new prototype is defined by the sample. In the ARAM model (Tan, 1995) new classes can be learnt while preserving previously learnt classes and so the stability-plasticity dilemma is regarded. However, ART networks are non-hierarchical networks and they consider new samples one at a time.

5. CONCLUSIONS

The proposed approached has proven functional. Although the networks were trained with only a few samples of the new classes they were able to classify the new class and no considerable deterioration of the classification results of the former classes could be observed. The experiments conducted showed encouraging results. It could be proved that hierarchical neural net-works are suitable for incremental adaption. New objects can be learnt with sufficient classification rates and in adequate time. The proposed approach enables the robot to deal with varying object categories in addition to the pre-defined categories. New objects can be learnt rather quickly with satisfactory quality. The quality can even be increased by retraining with additional samples and in an offline phase sophisticated learning algorithms are used to further improve the classification quality.

REFERENCES

[1] Carpenter, G. A. and Grossberg, S. (2002). Adaptive Resonance Theory. In Arbib, M., editor, *The Handbook of Brain Theory and Neural Networks,* pages 87–90. MIT Press, Cambridge, 2nd edition.

[2] Coppola, D. M., Purves, H. R., McCoy, A. N., and Purves, D. (1998). The distribution of oriented contours in the real world. *Proceedings of the National Academy of Sciences USA,* 95(7):4002–4006.

[3] Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification.* John Wiley & Sons, New York, 2nd edition.

[4] Fay, R., Kaufmann, U., Schwenker, F., and Palm, G. Learning object recognition in an neurobotic system. *3rd IWK Workshop SOAVE2004 - SelfOrganization of AdaptiVE behavior, Illmenau, Germany,* pages 198– 209.

[5] Freeman, W. T. and Roth, M. (1995). Orientation histograms for hand gesture recognition. In *IEEE Inter-national Workshop on Automatic Face- and Gesture-Recognition,* pages 296–301, Z¨urich, Switzerland.

[6] Grossberg, S. (2000). Adaptive resonance theory. Technical Report TR-2000-024, Center for Adaptive Systems and Department of Cognitive and Neural Science, Boston University.

[7] Knoblauch, A., Fay, R., Kaufmann, U., Markert, H., and Palm, G. (2004). Associating Words to Visually Recognized Objects. In Coradeschi, S. and Saffiotti, A., editors, *Anchoring symbols to sensor data. Papers from the AAAI Workshop. TechnicalReport WS-04-03,* pages 10–16. AAAI Press, Menlo Park, California.

[8] Nene, S. A., Nayar, S. K., and Murase, H. (1996). Columbia object image library (coil-20). Technical Report Technical Report CUCS-005-96, Columbia University.

[9] Penrose, R. (1955). A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society,* 51:406–413.

[10] Schwenker, F. (2001). Solving multi-class pattern recognition problems with tree structured support vector ma-chines. In Radig, B. and Florczyk, S., editors, *Mustererkennung 2001,* pages 283–290. Springer.

[11] Schwenker, F., Kestler, H. A., and Palm, G. (2001). Three learning phases for radial-basis-function networks. *Neural Networks,* 14:439–458.

[12] Simon, S., Schwenker, F., Kestler, H. A., Kraetzschmar, G. K., and Palm, G. (2002). Hierarchical object classification for autonomous mobile robots. In *International Conference on Artificial Neural Networks (ICANN),* pages 831–836.

[13] Tan, A.-H. (1995). Adaptive resonance associative map. *Neural Networks,* 8(3):437–446.