ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Visual object classification by sparse convolutional neural networks

Alexander Gepperth[1]

1- Ruhr-Universität Bochum - Institute for Neural Dynamics
Universitätsstraße 150, 44801 Bochum - Germany

Abstract. A convolutional network architecture termed *sparse convolutional neural network* (SCNN) is proposed and tested on a real-world classification task (car classification). In addition to the error function based on the mean squared error (MSE), approximate decorrelation between hidden layer neurons is enforced by a weight orthogonalization mechanism. The aim is to obtain a sparse coding of the objects' visual appearance, thus removing the need for a dedicated feature selection stage. Working on unprocessed image data only, it is demonstrated that classification accuracies can be improved by the proposed method compared to purely MSE-trained SCNNs and fully-connected multilayer perceptron architectures.

1 Introduction

In many real-world classification tasks there is a need for classifiers that can learn from examples, such as neural networks (NNs) or support vector machines. Typically, the performance of such classifiers depends strongly on a suitable preprocessing of the input, but it is far from clear what characterizes an optimal preprocessing or if there even exists an optimal solution. Sometimes it is required that preprocessing should reduce the dimensionality of the input as far as possible, whereas another objective is to make preprocessing invariant to certain transformations of the input (typically translation, rotation and scaling are investigated in this context). The process of choosing an appropriate preprocessing transform is referred to as *feature selection*. In addition to constraints on error rates, the available processing time is usually bounded from above, too, especially in computer vision. Therefore, not only the accuracy of classifiers is important but also their execution speed.

Convolutional neural networks (CNNs) [1] were proposed to address all of these issues. They are specialized instances of multilayer perceptrons (MLPs) and thus essentially feed-forward neural networks (NN). Due to their connectivity, CNNs can be implemented using standard techniques from digital image processing called *convolutions*, permitting very high execution speed (see [2, 3] for recent applications of CNNs). In addition, CNNs operating on unprocessed image data can be seen as learning a preprocessing transform, thus eliminating the need for feature selection.

In this contribution, the original CNN design is modified with the aim of obtaining non-redundant features for visual object classification in a systematic and well-defined way. This is achieved by an unsupervised (i.e. not using the class information) orthogonalization step in the learning mechanism. Thus the

proposed model, termed *sparse convolutional neural network* (SCNN), uses a systematic approach to ensure non-redundancy of the preprocessing transform. At the same time, it is easier to implement than the original CNN proposal, where the non-redundancy of features was achieved by the connection structure of the network. Nevertheless all the desirable properties of CNNs are retained.

The car classification task used to test the SCNN model is derived from an application in Advanced Driver Assistance Systems. The car detection problem and its background are described in [4]. The problem is treated offline, i.e., training and performance measurements are performed on datasets of examples.

2 Training data

Training datasets consisting of individual examples are taken from gray-valued video sequences. Since SCNNs are required to *learn* an appropriate preprocessing transform, no preprocessing is performed except the scaling of the gray-valued pixel data to a fixed size (matched by the input layer of the NNs) and the subtraction of the mean gray value.

A total of four datasets is created, denoted D_{train}, D_{val}, D_{test} and D_{ext}. Each dataset contains 1000 positive and 1000 negative examples. Although car classification is considered a simple problem, the classification task as defined by the datasets used here is quite difficult since the training examples exhibit strong variance in scale and position. For positive examples, scalings up to a factor of $\sqrt{2}$ and simultaneous translations up to 10% of an example's width and height are possible and must be tolerated by the classification.

3 Sparse convolutional neural networks

Like the original proposal [1] they are derived from, SCNNs are feed-forward neural networks, (i.e., MLPs) with "patchy connectivity" (see fig. 1). However, the connection of layers in SCNNs is simpler as compared to [1]: The SCNN model has an input layer of fixed dimension, one or more hidden layers, and an output layer containing a single element. Each layer is connected only to one other layer (the preceding one) as described in the next section (see also fig. 1).

3.1 NN structure

A layer l having dimensions $L_l^x \times L_l^y$ is subdivided into identical *cells* of neurons of dimension $C_l^x \times C_l^y$. Thus, a neuron can be assigned coordinates $\vec{n} = (l, \vec{c}, \vec{i})$, where $\vec{c}$ denotes the two-dimensional index of the cell within layer l, and $\vec{i}$ the neuron's coordinate within its cell. Within one cell, each neuron is connected to the same rectangular patch of neurons in layer $l-1$ which is termed a neuron's *receptive field* (RF). Receptive fields can overlap in x- and y-direction. Connection strengths are denoted by $w_{\vec{n}\vec{n}\prime}$ where $\vec{n}$ specifies the coordinates of the destination neuron and $\vec{n}\prime$ those of the source neuron. Please refer to fig. 1 for a visualization. The set of all weights connecting a neuron to its RF is denoted

Fig. 1: Left: sketch of the SCNN connectivity. Receptive fields are drawn in by solid ellipses, cells by dashed ellipses. Input filters connecting a neuron to its receptive field are shown as dashed or dotted lines. Note that receptive fields can overlap. Dashed and dotted lines represent sets of *equivalent* input filters, see text for details. Right: Speed advantage in whole-image classification. SCNNs, represented as boxes, are repeatedly applied and then shifted to different locations, covering the whole image. Only non-overlapping regions need to be calculated again at such a step, since the input filters do not depend on the spatial position within the SCNN. It is therefore sufficient to perform a convolution centered around each image pixel only once.

input filter. A weight-sharing constraint enters via the requirement that neurons within a layer l, having the same within-cell coordinates $\vec{i}$ but being connected to different RFs, must have identical input filters. It is this constraint which allows to implement a network run by a series of convolutions. In contrast, each neuron in one cell is allowed to be connected to the common RF by different filters than the other neurons in that cell. Effectively, the size of one cell, $C_l^x \times C_l^y$, specifies the number of convolution filters necessary for the simulation of each layer, whereas the size of receptive fields determines the dimensions of the convolution filters. For each layer l, sets of weights that are required to have the same value by the weight-sharing property are called *equivalent*. Obviously it is desirable to obtain a trained SCNN which requires as few convolution filters as possible while maintaining high classification accuracy.

The activity $A_{\vec{n}}$ of a neuron is calculated from the activities of its RF and the weight values in its input filter as $A_{\vec{n}} = \sigma(\sum_{\vec{n}\prime \in \mathrm{RF}} A_{\vec{n}\prime} w_{\vec{n}\vec{n}\prime})$ using the sigmoidal activation function $\sigma(x) = \frac{x}{1+|x|}$.

Each neuron (except for the input layer) is connected to a bias neuron whose activation is constant (here: 1.0).

3.2 Learning in SCNNs

The SCNN model differs from conventional fully-connected MLPs in three respects: firstly, neurons receive input only from a rectangular patch of the preceding layer, and secondly, certain input filters are required to be equal. Thirdly, the input filters of adjacent neurons (more precisely: neurons in the same cell, see previous section) are required to be orthogonal, leading to approximate decorrelation of neuron outputs. A dedicated learning algorithm must therefore be used to ensure the second and third property. The procedure is straightforward:

During each learning step or *epoch*, all weights of the SCNN are treated as if they were independent. An improved variant of the well-known Rprop learning algorithm (IRprop+, see [5]) is applied to the NN for n epochs. After each epoch, the weight-sharing condition is enforced by computing, for each layer l, the average of each set of equivalent weights, and then setting the weights in each set to that value.

Finally, input filters in each cell of the network are orthogonalized by performing an iterative procedure (see [6] for details and proof) 10 times and then normalized to 1.0. Treating the input filters of each neuron within one cell as column vectors of a matrix W, the orthogonalization is defined by

$$W_0 = W/||W||$$
$$W_{t+1} = 1.5W_t - 0.5W_tW_t^TW_t$$

3.3 Error measures

During training, the error is calculated as $E(D) = E_{\mathrm{MSE}}(D) = \frac{1}{|D|}\sum_{p=0}^{|D|}(A_p^{out} - c_p)^2$ using a dataset D. E_{MSE} denotes the mean squared error (MSE), which uses the class label c_p of pattern p and the activation A_p^{out} of the CNN's output neuron in response to pattern p. The minimization of E is performed by gradient descent, see also [5]. When evaluating the performance of a trained network, the classification error $\mathrm{CE}(D)$ on a dataset D is used. It is defined as $\mathrm{CE}(D) = 1 - \frac{1}{|D|}\sum_{p=0}^{|D|}\theta(A_p^{out} - \tau)$, where θ denotes the step function and τ a threshold assigned to each NN (here always taken to be 0).

4 Experimental setup

Weights are initialized randomly to a range of $[-0.01; 0.01]$ before learning, which is always performed for 60 epochs using the dataset D_{train}. The NN with the best error $E_{\mathrm{MSE}}(D_{\mathrm{val}})$ is taken to be the outcome of learning. Learning is repeated 10 times for each NN, each time with a different random seed. To test the performance of a trained network, the mean value of $\mathrm{CE}(D_{\mathrm{test}} \cup D_{\mathrm{ext}})$ is determined. A Wilcoxon rank-sum test is used to ascertain that experimental results are indeed meaningful.

4.1 Network architectures

In order to reduce the effects which NN topology can have on learning capability, five SCNN topologies (the *reference architectures*) are defined which will be used for all further investigations. SCNN C_5 is included to demonstrate that the model can easily include several layers if necessary. Input filter sizes are chosen to be compatible with computer vision real-time requirements. Overlaps between input filters are varied from 1 to 7 neurons in order to demonstrate the possible effect of this difference on SCNN learning. Table 1 gives an overview over the reference architectures.

SCNN	hidden layers	RF	overlaps
C_1	9	9	1
C_2	10	9	5
C_3	15	9	7
C_4	18	9	7
C_5	15-9	9-7	5-3

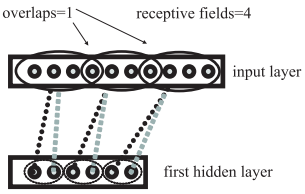

Table 1: Left: SCNN reference set used in the experiments. The column "RF" describes the sizes of rectangular receptive fields, or equivalently the dimensions of the input filters projecting to the next layer. Column "overlaps" gives the number of neurons by which receptive fields overlap (in both directions). Right: visualization of the quantities given in the table, using an example SCNN structure. For a detailed explanation, see fig. 1.

4.2 Experiments

In the first experiment (referred to in the following as X_{MLP}), a fully connected MLP with one hidden layer (containing 20 neurons) is trained using standard MLP learning with E_{MSE} as error measure (see section 3.2). In experiment X_{WS}, all SCNN architectures defined in the previous section are trained as described in section 3.2 but without using the decorrelation-enforcing mechanism. This is turned on in experiment $X_{\mathrm{WS,DEC}}$ and learning is again performed on all SCNN architectures.

5 Results

The results of the experiments are summarized in table 2. The mean error taken over all 50 performance measurements (10 for each SCNN architecture) is significantly[1] lower in experiment $X_{\mathrm{WS,DEC}}$ than in experiment X_{WS}. Care must be taken in interpreting the best results of the two experiments, since the best X_{WS} result is comparable to that of $X_{\mathrm{WS,DEC}}$. It must be borne in mind, however, that the two distributions have been shown to have different true means with very high probability. Therefore, it is very likely that increasing the sample size in both experiments will yield better results for experiment $X_{\mathrm{WS,DEC}}$.

Of special interest is the fact that the results of experiment X_{MLP} are strongly inferior to those of experiments X_{WS}, $X_{\mathrm{WS,DEC}}$. This is unexpected since the number of free parameters is much lower in the latter experiments (especially the last), and shows that the preprocessing step performed by an SCNN facilitates classification significantly.

6 Discussion

As the results plainly show, the use of SCNNs using a decorrelation-enforcing mechanism is a suitable way of reducing the classification error in a difficult real-

[1] Wilcoxon rank-sum test, $p \geq 0.99$

	X_{MLP}	X_{WS}	$X_{\mathrm{WS,DEC}}$
mean	45.1	18.06	15.64
best	36.54	10.66	11.67

Table 2: Classification errors of MLP and SCNN models (with and without orthogonalization). The line two lines refer to the best/mean classification error $\mathrm{CE}(D_{\mathrm{test}} \cup D_{\mathrm{ext}})$ of all 50 runs of one experiment.

world classification problem. Notable is the fact that even without this additional mechanism, the SCNN model performs much better than a fully connected MLP.

In order to extend the SCNN model, other feature-generating mechanisms can be used: especially, the possibility of performing (nonlinear) principal or independent component analysis using the input filters as the mixing matrix (see [6] and references therein) seems very promising.

On a broader perspective, the possibility of eliminating the feature selection process from object recognition is a strong motivation to continue research on SCNN models. Thus, one could work with "raw" image data only and learn relevant object features from training examples alone. A possible application could be *feature base learning*, i.e., learning a common preprocessing transform for several object classes. This could be a way to increase the speed advantage in using SCNNs still further.

7 Acknowledgements

I wish to thank J. Edelbrunner for helpful discussions and inspiration on the subject of convolutional neural networks.

References

[1] Y LeCun, L Bottou, Y Bengio, and P Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324, 1998.

[2] C Garcia and M Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1408–1423, November 2004.

[3] M Szarvas, A Yoshizawa, M Yamamoto, and J Ogata. Pedestrian detection using convolutional neural networks. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 224–229, 2005.

[4] A Gepperth, J Edelbrunner, and T Bücher. Real-time detection and classification of cars in video sequences. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 625–631, 2005.

[5] C Igel and M Hüsken. Empirical evaluation of the improved Rprop learning algorithm. *Neurocomputing*, 50(C):105–123, 2003.

[6] A Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10:626–634, 1999.