On the Analysis of Neural Networks for Image Processing

Berend Jan van der Zwaag[1], Kees Slump[1], and Lambert Spaanenburg[2]

[1] Dept. of Electrical Engineering, University of Twente
P.O.Box 217, 7500 AE Enschede, The Netherlands
b.j.vanderzwaag@utwente.nl
[2] Dept. of Information Technology, Lund University
Box 117, S-22100 Lund, Sweden

Abstract. This paper illustrates a novel method to analyze artificial neural networks so as to gain insight into their internal functionality. To this purpose, we will show analysis results of some feed-forward–error-back-propagation neural networks for image processing. We will describe them in terms of domain-dependent basic functions, which are, in the case of the digital image processing domain, differential operators of various orders and with various angles of operation. Some other pixel classification techniques are analyzed in the same way, enabling easy comparison.

1 Introduction

Since the early development of artificial neural networks, researchers have tried to analyze them to gain insight into their behavior. For certain applications and in certain problem domains this has been successful. In particular in decision making and in other systems that can easily be expressed in sets of rules, great advances have been made by the development of so-called rule extraction methods [1]. Neural network systems with relatively few inputs can sometimes be analyzed by means of a sensitivity analysis [2].

However, most neural network systems are so high-dimensional that an extracted rule base would become too large to be easily interpreted, or so nonlinear that a sensitivity analysis would only be valid for a small part of the input space. For this reason, we propose domain-specific neural network analysis methods that utilize domain-specific base functions [5] that are easy to interpret by the user. An analysis in terms of base functions may also make clear how to (re)construct a superior system using those base functions, thus using the neural network as a construction advisor.

2 Analysis of Neural Networks

In general, artificial neural networks with unsupervised training merely reorganize the input space, so analyzing them after training becomes fairly simple: an

V. Palade, R.J. Howlett, and L.C. Jain (Eds.): KES 2003, LNAI 2774, pp. 950–957, 2003.

Table 1. Some application domains with potential domain-specific base functions

application domain	potential base functions	remarks
signal processing (1-D)	basic operational filters	
digital image processing (2-D)	differential operators	
general classification problems	feature map regions	cp. Kohonen's self-organizing map
decision theory	if-then rules (fuzzy or not)	i.e., classical rule extraction
control theory	basic control operators	

investigation into the reorganized input space reveals how the net has restructured the input space.

Analyzing neural nets trained under supervision is far more complicated, for input and output spaces are usually in different domains (e.g., a character recognition system has an image as input, and a character as output), whereas in the unsupervised case, input and output spaces are basically the same, although organized in different ways.

The idea of describing a trained neural network in terms of basic domain-specific functions was introduced and presented in earlier publications [3, 4, 5]. For many problems in certain domains, such as linguistics and decision theory, the common, domain-dependent base functions could be chosen to be *if–then* rules or decision trees, in which case the analysis reduces to common rule extraction. Table 1 lists a few more problem domains where neural networks have been successfully applied. For each of these domains, potential base functions are presented.

In digital image processing, the user will be familiar with image filters, particularly 2-dimensional differential operators. Hence, a description of an image processing neural network in terms of these digital image operators will enhance the understanding of the network's functionality. In the following sections, we will illustrate the analysis of feed-forward–error-back-propagation neural networks trained for digital image processing, along with some well-known non-neural image processing techniques for comparison.

3 Digital Image Processing

We will treat several basic techniques commonly used in digital image processing, or more specifically, in pixel classification.

Edge Detection Edge detection is frequently used in image segmentation. In that case an image is seen as a combination of segments in which image data are more or less homogeneous. Two main alternatives exist to determine these

segments: (1) classification of all pixels that satisfy the criterion of homogeneousness; (2) detection of all pixels on the borders between different homogeneous areas. To the first category belong pixel classification (depending on the pixel value the pixel is part of a certain segment) and region growing methods. The second category is edge detection.

In fact, edge detection is also some sort of pixel classification: every pixel is either part of an edge or not. All edges together form the contours of the segments. After edge detection sometimes edge linking is used, in order to try to get the contours closed, as in practice not all pixels will be classified correctly, due to noise, etc.

Many edge detection filters only detect edges in certain directions, therefore combinations of filters that detect edges in different directions are often used to obtain edge detectors that detect all edges. Some examples of filter templates for edge detection are:

$$\text{Sobel, } 0^\circ\text{: } \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}\text{; Kirsch, } 45^\circ\text{: } \begin{bmatrix} -5 & -5 & 3 \\ -5 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}\text{; compass, } 90^\circ\text{: } \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}.$$

The dependency on the edge direction ϕ is not very strong; edges with a direction $\phi \pm 45^\circ$ will also activate the edge detector. This will also become clear when we investigate the analysis results in Sect. 4.

Line Detection A similar filter is the line detector, which detects lines rather than edges. In fact, a line can be seen as two edges lying parallel and close to each other. Three simple line detector templates are:

$$\text{horizontal: } \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}\text{; vertical: } \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}\text{; diagonal: } \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}.$$

Combining these three, together with a second diagonal filter perpendicular to the one shown above, would result in a practically omnidirectional line detector.

Spot Detection A third filter type is the spot detector. Example templates for spot detection are:

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & 24 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix},$$

where the template on the right is less sensitive to noise than the one on the left, because of the difference in the neighborhood sizes.

Differential Operators A special type of image filters are the differential operators. Usage of these operators is based on the detection of changes in greylevel. The gradient vector of a 2-dimensional continuous image $f(x, y)$ is defined as

$$\vec{\nabla} f(x,y) = \left[\frac{\partial f(x,y)}{\partial x} \quad \frac{\partial f(x,y)}{\partial x} \right]^T = \begin{bmatrix} f_x(x,y) \\ f_y(x,y) \end{bmatrix}. \tag{1}$$

For discrete images this can be seen as a template $[-1\ 1]$ for the gradient in the horizontal direction and as a template $\left[\begin{smallmatrix}1\\-1\end{smallmatrix}\right]$ for the gradient in the vertical direction. The gradients in the diagonal directions can be determined with Roberts' templates $\left[\begin{smallmatrix}-1 & 0\\ 0 & 1\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}0 & 1\\ -1 & 0\end{smallmatrix}\right]$ for gradients in 45° and 135° directions, respectively. The direction of the edges detected by a differential operator is perpendicular to the direction of the gradient.

3.1 Describing a Template in Terms of Differential Operators

We will now describe how an arbitrary image filter in matrix form can be seen as a composition of several differential operators, where the operators are of varying orders and operate in varying directions.

We can write an image as a set of pixels $f_{p,q}$ and an image filter as a (template) matrix with elements $w_{n,m}$. We can then classify a pixel $f_{p,q}$ by looking at the pixel's neighborhood, which has the same size as the filter template, say $(2N+1)\times(2M+1)$. We then calculate the discrete convolution

$$g_{p,q} = \sum_{n=-N}^{N} \sum_{m=-M}^{M} w_{n,m} f_{p-n,q-m}, \tag{2}$$

where $f_{p,q}$ can be classified by thresholding $g_{p,q}$. For example, we can classify $f_{p,q}$ as an edge pixel, if $g_{p,q}$ exceeds a certain threshold and is a local maximum in the direction perpendicular to ϕ in the image $g_{p,q}$.

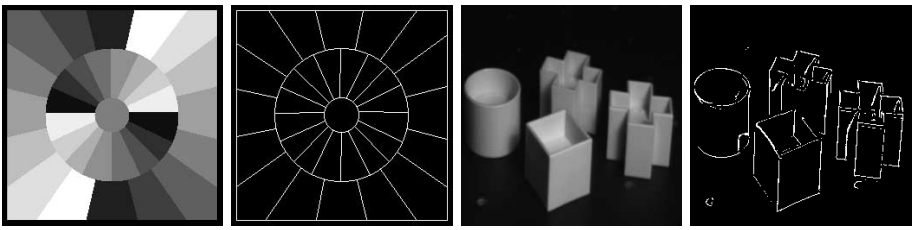
In order to transform a template into a set of gradient filters, we first calculate the Taylor series expansion of the Fourier transformed template, and then we apply the inverse Fourier transform to get a description of the template which gives us knowledge about the differential components [3]. The reason for using a Fourier transformation lies in the fact that a Fourier transformed filter description consists of a series of sinusoidals, which are easily differentiated to determine the Taylor components. For a better insight into the types of differential operators, it can be determined if a filter is directional, and if so, what its main direction of operation is. To this purpose, we rotate the coordinate axes over an angle θ to new coordinate axes.

The whole sequence of transformations is described in more detail in [3]. For the sake of brevity, we simply state here the result of the consecutive transformations described above as a component-wise description of the filter operation:

$$g'_\theta(\xi,\eta) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \beta_{\theta,i,j} \frac{\partial^{i+j}}{\partial \xi^i \partial \eta^j} f'_\theta(\xi,\eta)\,, \tag{3}$$

with $i+j=r$ and

$$\beta_{\theta,i,j} = (-1)^{i+j} \sum_n \sum_m w_{n,m} \sum_{k=0}^{i} \sum_{l=0}^{j} \frac{(-1)^l n^{k+l} m^{i+j-k-l}}{k!l!(i-k)!(j-l)!} (\sin\theta)^{i-k+l} (\cos\theta)^{j+k-l}\,. \tag{4}$$



a b c d

Fig. 1. (a) input training image; (b) reference edge map; (c) test image; (d) result after edge detection by a neural network

Equation (4) gives the Taylor series coefficients of the edge detection filter template, from which we can deduce of which orders of differential operators the filter consists, i.e., those i and j that give the larger $\beta_{\theta,i,j}$, and in which direction(s) these operators work optimally, i.e., the angle(s) θ for which $\beta_{\theta,i,j}$ is maximal given certain i and j. This can be represented graphically by drawing the value of $\beta_{\theta,i,j}$ as a function of θ for various i and j. See Figs. 2 and 3 for examples. In these graphs, the absolute value of $\beta_{\theta,i,j}$ as a function of θ is represented by the distance from the center of the graph in the direction of θ; positive values of $\beta_{\theta,i,j}$ are shown in blue (thick lines), negative values in red (thin lines).

3.2 Neural Network Edge Detector

In order to test our analysis method, we trained several artificial neural networks for edge detection. The neural networks were of the feed-forward error-backpropagation type, with 3×3 to 11×11 inputs, 4 to 8 units in the single hidden layer, and a single output. All units used sigmoid activation functions. Some networks were trained with a training image containing sharp edges only, see Fig. 1a,b. A different image was used as a test set, see Fig. 1d for a test result by a neural network edge detector. Other networks were trained with a similar image as the one in Fig. 1a, but containing sharp edges as well as blurred ones, and that had Gaussian noise added to one half of the image. This made the neural network edge detectors less sensitive to noise, as will also become visible in the analysis results in Sect. 4.

4 Results

As (4) gives the Taylor series coefficients of any image filter template, we can apply it to existing edge detector templates as well as to a neural network edge detector's hidden units, whose weights can be regarded as templates as well. First, we will show brief analysis results for three horizontal edge detection templates. As can clearly be seen from the left half of Fig. 2, these templates show

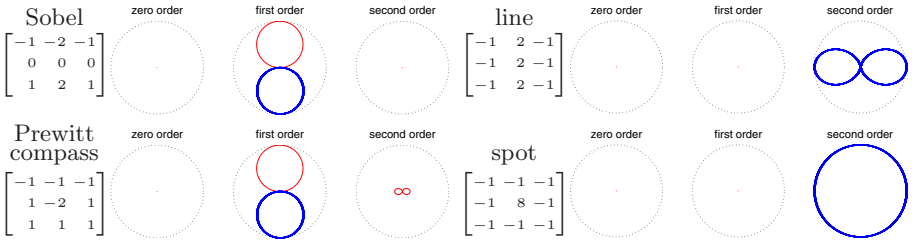


Fig. 2. Templates and low-pass, gradient, and second-order gradient analysis results for two horizontal edge detector templates, one vertical line detector, and one spot detector

no low-pass (averaging) behavior, have a strong first-order gradient operation in vertical direction, as could be expected, and no or weak second-order gradient behavior. The differences between these three filters are visible in the second- (and higher-) order behavior.

In the case of line-detecting templates, it is clear from the first template shown in Sect. 3, that it detects horizontal lines, and from the second one that it detects vertical lines, therefore it is not surprising that the second-order coefficients are strongest in the vertical and horizontal directions respectively, as seen in Fig. 2. The zero-order and first-order coefficients are all zero. This is also the case for the spot detector shown in the same figure, except that in this case the second-order behavior is omnidirectional, or rotation-invariant. This, of course, follows from the fact that spots have no direction.

Figure 3(left) shows results for the 4 hidden units of a small neural network edge detector, which was trained with the sharp edges shown in Fig. 1a,b. For the purpose of showing the hidden units' weight values graphically, they have been scaled to values between –1 (black) and +1 (white). The weight templates shown in the first column already give some insight into their behavior, but the Taylor series coefficient analysis clearly shows that three of the units detect edges in various directions, and one unit acts as a second-order gradient filter with minor first-order gradient behavior. Notice that neither of these four units has a significant low-pass (zero-order gradient) component.

Another neural network with the same architecture was trained with sharp, blurred, and noisy variants of the images shown in Fig. 1a. The weight templates of the hidden units are shown in Fig. 3(right) along with the graphical representations of their Taylor series coefficients. This network's units have similar gradient components as the previous one, although the second-order gradient components are somewhat stronger. The low-pass components are more significantly present, as compared to the previous network. This is a result of the different training. Low-pass or averaging behavior makes the network less sensitive to noise and improves the edge detection ability of the second network.

Although the above only gives some analysis results for the units in the hidden layer, it should be clear that a description of the neural network as a whole can be derived from these results. The weight between a hidden units and the output

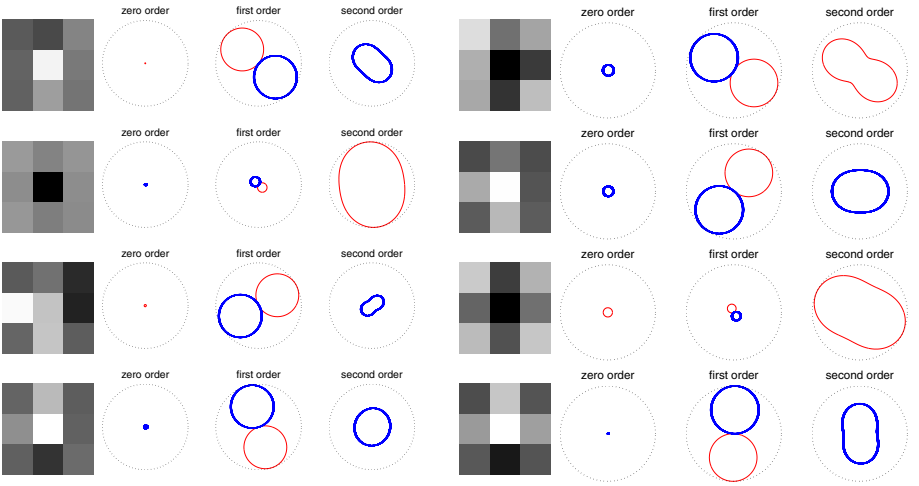


Fig. 3. Weight templates and gradient analysis results for all hidden units of two (3×3)_4_1 neural network edge detectors, one trained with sharp edges (left) and one trained with sharp, blurry, and noisy edges (right). Dark colored squares in the templates represent negative weight values, whereas light colored ones stand for positive values

unit represents the "importance" of the hidden unit's edge detection outcome, which is then combined with the other hidden units' outcomes into a single answer indicating whether the pixel under investigation belongs to an edge or not.

Some larger neural networks have also been trained and analyzed, with similar results, although in general, the larger the network, the more variety in behavior among the neural units. In a few cases, certain units showed very strong higher-order behavior, indicating that those units functioned as noise detectors only. Although such units usually have weak connections to the output unit (low importance), removing them from the network (pruning) often results in worse edge detection capabilities for the network as a whole. This is because the noise detecting unit decreases the confidence of an edge detection outcome if the local neighborhood around the pixel under investigation is very noisy. Units detecting sharp edges could easily misclassify such pixels as edge pixels.

5 Conclusions

We have trained neural networks to detect edges in digital images and analyzed them into gradient filter components. From the results displayed and described in the previous sections it is clear that it is indeed feasible to describe the trained neural networks in terms of basic functions from the image processing domain.

The description with gradient filter components gives easy insight into the behavior of the neural network as an edge detector, and allows simple comparison

with other edge detectors which can in the same way be described in terms of gradient filter components.

In general, the analysis consists in describing the internal functionality of the neural network in terms of basic domain functions, functions that can be considered basic in the application domain of the neural network. This means that users who may not be familiar with artificial neural networks, but who are familiar with basic functions that are often used in their problem domain, can gain insight in the way the neural network solves their problem. For such users, this is often an important factor in deciding to apply artificial neural networks to a problem that may be difficult to solve otherwise.

References

[1] Craven, M. W., Shavlik, J. W.: Using sampling and queries to extract rules from trained neural networks. Machine Learning: Proceedings of the Eleventh International Conference, San Francisco, CA (1994) 950

[2] Hashem, S.: Sensitivity analysis for feedforward artificial neural networks with differentiable activation functions. Proceedings of the 1992 International Joint Conference on Neural Networks, Vol. 1. IEEE Press, Piscataway, NJ, USA (1992) 419-424 950

[3] Van der Zwaag, B. J., Slump, C.: Analysis of neural networks for edge detection. Proceedings of the ProRISC Workshop, Veldhoven, the Netherlands (2002) 580-586 951, 953

[4] Van der Zwaag, B. J., Slump, C., Spaanenburg, L.: Process identification through modular neural networks and rule extraction. In Ruan, D., D'hondt, P., Kerre, E. E. (eds.): Computational Intelligent Systems for Applied Research: Proceedings of the 5th International FLINS Conference, Ghent, Belgium. World Scientific, Singapore (2002) 268-277 951

[5] Van der Zwaag, B. J., Spaanenburg, L., Slump, C.: Analysis of neural networks in terms of domain functions. Proceedings IEEE Benelux Signal Processing Symposium SPS-2002, Leuven, Belgium (2002) 237-240 950, 951