Analysis of Neural Networks for Edge Detection

Berend Jan van der Zwaag[1], Kees Slump[1], and Lambert Spaanenburg[2,3]
[1] Systems and Signals, Dept. of Electrical Engineering, University of Twente
P.O.Box 217, 7500 AE Enschede, the Netherlands
E-mail: b.j.vanderzwaag@utwente.nl
[2] Dept. of Mathematics & Computing Science, Groningen University
P.O.Box 800, 9700 AV Groningen, the Netherlands
[3] Dept. of Information Technology, Lund University
Box 117, S-221 00 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, the elements of a feedforward-backpropagation neural network, that has been trained to detect edges in images, are described in terms of differential operators of various orders and with various angles of operation.**

Keywords—**Neural networks, rule extraction, edge detection, digital image processing, gradient filters.**

I. Introduction

Since the early development of artificial neural networks, but especially in the past 10 years, researchers have tried to analyze them to provide insight into their behavior. For certain applications and in certain problem domains this has been successful. In particular in decision making systems and 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], which is a nonparametric statistical analysis technique.

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 [3] that are easy to interpret by the user and that can even be used to optimize neural network systems. 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.

II. 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 investigation into the reorganized input space reveals how the network has restructured the input space.

Analyzing neural networks 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 they are 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]. 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 rule extraction. Table I lists a few more problem domains where neural networks have been successfully applied. For each of these domains, possible base functions are presented. In the following, we show for one such domain, i.e., digital image processing, how our analytical method can overcome the impracticality of more common knowledge extraction methods. In the case of edge detection – a digital image processing technique – the user will be familiar with image filters, particularly 2-dimensional differential operators. Hence, a description in terms of these digital image operators will enhance the understanding of the neural net's functionality. Therefore, we will illustrate the analysis of feed-forward–error-back-propagation neural networks trained for edge detection.

III. 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 bor-

TABLE I
Some application domains with potential base functions.

application domain	potential base functions
signal processing (1-D)	basic operational filters
digital image processing (2-D)	differential operators
general classification problems	feature map regions (compare Kohonen's self-organizing feature map)
decision theory	if-then rules (fuzzy or not) (i.e., rule extraction as a special case of the proposed method)
control theory	basic control operators

ders 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.

A. Finding edge pixels

In digital image processing, we can write an image as a set of pixels $f_{p,q}$ and an edge detection filter which detects edges with direction ϕ as a (template) matrix with elements $w_{n,m}$, see Figure 1. We can then determine whether a pixel $f_{p,q}$ is an edge pixel or not, by looking at the pixel's neighborhood, see Figure 2, where the neighborhood has the same size as the edge detector template, say $(2N+1)\times(2M+1)$. We then calculate the discrete convolution

$$g_{p,q} = \sum_{n=-N}^{N} \sum_{m=-M}^{M} w_{n,m} f_{p-n,q-m}, \qquad (1)$$

where $f_{p,q}$ can be classified 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 $g_{p,q}$.

$$\begin{bmatrix} w_{-N,M} & \cdots & w_{N,M} \\ \vdots & \ddots & \vdots \\ & w_{0,0} & \\ \vdots & \ddots & \vdots \\ w_{-N,-M} & \cdots & w_{N,-M} \end{bmatrix}$$

Fig. 1
A $(2N+1)\times(2M+1)$ template $w_{n,m}$.

$$\begin{bmatrix} f_{-P,Q} & \cdots & f_{P,Q} \\ \vdots & f_{p-N,q+M} \cdots f_{p+N,q+M} & \vdots \\ & \vdots \quad \ddots \quad \vdots & \\ & f_{p,q} \quad \ddots & \\ & f_{p-N,q-M} \cdots f_{p+N,q-M} & \\ f_{-P,-Q} & \cdots & f_{P,-Q} \end{bmatrix}$$

Fig. 2
A $(2P+1)\times(2Q+1)$ image with a $(2N+1)\times(2M+1)$ neighborhood around $f_{p,q}$.

Some examples of templates for edge detection are:

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

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.

B. 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

$$\vec{\nabla} f(x,y) = \begin{bmatrix} \dfrac{\partial f(x,y)}{\partial x} & \dfrac{\partial f(x,y)}{\partial x} \end{bmatrix}^T = \begin{bmatrix} f_x(x,y) \\ f_y(x,y) \end{bmatrix}. \quad (2)$$

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.

IV. Describing a Template in Terms of Differential Operators

We will now describe how an arbitrary edge detection 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. 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. 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.

A. Fourier transformation

For Fourier transformation, Dirac functions (δ) are assigned to the pixels $f_{p,q}$ and template elements $w_{n,m}$. Thus, the Fourier transformed (sub)image is

$$\begin{aligned} F(u,v) &= \mathcal{F}\left\{ \sum_{n=-N}^{N} \sum_{m=-M}^{M} f_{p-n,q-m}\delta(x-n\Delta, y-m\Delta) \right\} \\ &= \sum_{n=-N}^{N} \sum_{m=-M}^{M} f_{p-n,q-m} e^{-2\pi j\Delta(nu+mv)}, \end{aligned} \tag{3}$$

where δ is the Dirac function, $x{=}p\Delta_x$, and $y{=}q\Delta_y$, with $\Delta_x{=}\Delta_y{=}\Delta$ as the sampling period. Unlike the original images, the transformed images are continuous functions. The Fourier transformation $G(u,v)$ of the edge map $g_{p,q}$ is calculated similar to the Fourier transformed input image $F(u,v)$. If we write the filter function $W(u,v)$ as the transformation of the template $w_{n,m}$,

$$W(u,v) = \sum_{n=-N}^{N} \sum_{m=-M}^{M} w_{n,m} e^{-2\pi j\Delta(nu+mv)}, \tag{4}$$

then

$$G(u,v) = W(u,v)F(u,v). \tag{5}$$

This equation describes the filter operation in the frequency domain. A convolution in the space domain becomes a simple multiplication in the frequency domain.

B. Taylor series expansion

The Taylor series expansion of a function $h(u,v)$ around $(0,0)$ is

$$h(u,v) = \sum_{r=0}^{\infty} \frac{1}{r!} \sum_{i=0}^{r} \binom{r}{i} u^i v^{r-i} \left. \frac{\partial^r h}{\partial u^i \partial v^{r-i}} \right|_{(0,0)}. \tag{6}$$

The Taylor series of $W(u,v)$ is then

$$W(u,v) = \sum_n \sum_m w_{n,m} \sum_{r=0}^{\infty} \sum_{i=0}^{r} \frac{u^i v^{r-i}}{i!(r-i)!} (-2\pi j\Delta)^r n^i m^{r-i}, \tag{7}$$

with which the transformed edge map $G(u,v)$ can be written as

$$\begin{aligned} G(u,v) &= \sum_n \sum_m w_{n,m} \\ &\quad \cdot \sum_{r=0}^{\infty} \sum_{i=0}^{r} \frac{u^i v^{r-i}}{i!(r-i)!} (-2\pi j\Delta)^r n^i m^{r-i} F(u,v) \\ &= \sum_{r=0}^{\infty} \sum_{i=0}^{r} (-2\pi j u\Delta)^i (-2\pi j v\Delta)^{r-i} \frac{F(u,v)}{i!(r-i)!} \\ &\quad \cdot \sum_n \sum_m w_{n,m} n^i m^{r-i}. \end{aligned} \tag{8}$$

Equation (8) describes the output edge map, transformed to the frequency domain. If we now want a similar description of the edge map in the space domain, i.e., expansion into gradients of different orders, inverse Fourier transformation is required. This yields a continuous analogon of $g_{p,q}$:

$$g(x,y) = \sum_{r=0}^{\infty} \sum_{i=0}^{r} \frac{\partial^r f(x,y)}{\partial x^i \partial y^{r-i}} \frac{(-1)^r}{i!(r-i)!} \sum_n \sum_m w_{n,m} n^i m^{r-i}. \tag{9}$$

From Eq. (9) it can be deduced that the image filter described by $w_{n,m}$ can be regarded as composed of a series of differential operators, with the following continuous analogon:

$$g(x,y) = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \alpha_{i,j} \frac{\partial^{i+j}}{\partial x^i \partial y^j} f(x,y), \tag{10}$$

with $i{+}j{=}r$ and

$$\alpha_{i,j} = \frac{1}{i!j!} \sum_{n=-N}^{N} \sum_{m=-M}^{M} w_{n,m} n^i m^j \tag{11}$$

being the coefficients for the gradient vectors for the various values of i (order in x-direction) and j (order in y-direction).

C. Coordinate transform

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, the x- and y-axes can be rotated over an angle θ to new

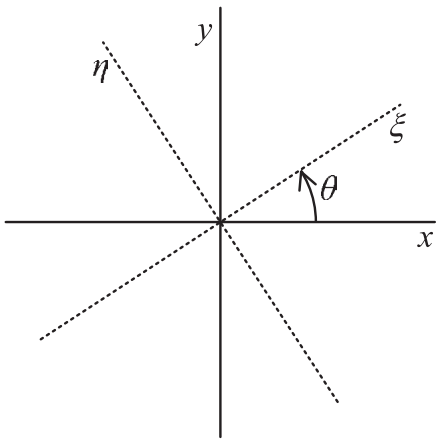

Fig. 3
Coordinate transformation from (x, y) to (ξ, η).

coordinate axes, see Figure 3. Now x and y can be written as functions of ξ and η, depending on θ:

$$\begin{aligned} x &= x_\theta(\xi,\eta) = \xi\cos\theta - \eta\sin\theta \\ y &= y_\theta(\xi,\eta) = \xi\sin\theta + \eta\cos\theta \end{aligned} \tag{12}$$

With this transformation, $f(x, y)$ can also be written as

$$f(x,y) = f_\theta\left(x_\theta(\xi,\eta), y_\theta(\xi,\eta)\right) = f'_\theta(\xi,\eta). \tag{13}$$

N.B. the notation f'_θ has nothing to do with the gradient of f_θ.

We can now adapt Eqs. (3)-(11) to the coordinate transformation of Eq. (12). The Fourier transformed input (sub)image is then

$$\begin{aligned} F'_\theta(\mu,\nu) &= \mathcal{F}\Big\{\sum_n\sum_m f_{p-n,q-m} \\ &\qquad \cdot\delta[(x-n\Delta)\cos\theta + (y-m\Delta)\sin\theta, \\ &\qquad\quad -(x-n\Delta)\sin\theta + (y-m\Delta)\cos\theta)]\Big\} \\ &= \mathcal{F}\Big\{\sum_n\sum_m f_{p-n,q-m}\delta(\xi - n\Delta\cos\theta - m\Delta\sin\theta, \\ &\qquad\qquad \eta + n\Delta\sin\theta - m\Delta\cos\theta)\Big\} \\ &= \sum_n\sum_m f_{p-n,q-m} \\ &\qquad \cdot e^{-2\pi j\Delta((n\cos\theta + m\sin\theta)\mu + (-n\sin\theta + m\cos\theta)\nu)}. \end{aligned} \tag{14}$$

In a similar manner as without the coordinate transform, the output edge map can be calculated from

$$G'_\theta(\mu,\nu) = W'_\theta(\mu,\nu)F'_\theta(\mu,\nu), \tag{15}$$

with

$$\begin{aligned} W'_\theta(\mu,\nu) &= \sum_n\sum_m w_{n,m} \\ &\qquad \cdot e^{-2\pi j\Delta((n\cos\theta + m\sin\theta)\mu + (-n\sin\theta + m\cos\theta)\nu)}. \end{aligned} \tag{16}$$

The Taylor series of the filter function $W'_\theta(\mu, \nu)$ is now written as

$$\begin{aligned} W'_\theta(\mu,\nu) &= \sum_n\sum_m w_{n,m}\sum_{r=0}^{\infty}\sum_{i=0}^{r}\frac{\mu^i\nu^{r-i}}{i!(r-i)!}(-2\pi j\Delta)^r \\ &\qquad \cdot(n\cos\theta + m\sin\theta)^i(-n\sin\theta + m\cos\theta)^{r-i} \end{aligned} \tag{17}$$

or, using the polynomial equality

$$(a+b)^i = \sum_{k=0}^{i}\binom{i}{k}a^k b^{i-k}, \tag{18}$$

we get

$$\begin{aligned} W'_\theta(\mu,\nu) &= \sum_{r=0}^{\infty}\sum_{i=0}^{r}(2\pi j\mu\Delta)^i(2\pi j\nu\Delta)^{r-i}\frac{(-1)^{r-i}}{i!(r-i)!} \\ &\qquad \cdot\sum_n\sum_m w_{n,m}\sum_{k=0}^{i}\binom{i}{k}(n\cos\theta)^k(m\sin\theta)^{i-k} \\ &\qquad \cdot\sum_{l=0}^{r-i}\binom{r-i}{l}(-n\sin\theta)^l(m\cos\theta)^{r-i-l}. \end{aligned} \tag{19}$$

Equation (19) is a Taylor series description of the filter function in the frequency domain. Now the complete output edge map can be calculated in the frequency domain and inverse Fourier transformation results in the following equation in ξ and η:

$$\begin{aligned} g'_\theta(\xi,\eta) &= \sum_{r=0}^{\infty}\sum_{i=0}^{r}(-1)^r\frac{\partial^r f'_\theta(\xi,\eta)}{\partial\xi^i\partial\eta^{r-i}}\sum_n\sum_m w_{n,m}\sum_{k=0}^{i}\sum_{l=0}^{r-i} \\ &\qquad \frac{(-1)^l n^{k+l}m^{r-k-l}}{k!l!(i-k)!(r-i-l)!}(\sin\theta)^{i-k+l}(\cos\theta)^{k+r-i-l} \end{aligned} \tag{20}$$

or

$$g'_\theta(\xi,\eta) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\beta_{\theta,i,j}\frac{\partial^{i+j}}{\partial\xi^i\partial\eta^j}f'_\theta(\xi,\eta), \tag{21}$$

with $i+j=r$ and

$$\begin{aligned} \beta_{\theta,i,j} &= (-1)^{i+j}\sum_{n=-N}^{N}\sum_{m=-M}^{M}w_{n,m}\sum_{k=0}^{i}\sum_{l=0}^{j} \\ &\qquad \frac{(-1)^l n^{k+l}m^{i+j-k-l}}{k!l!(i-k)!(j-l)!}(\sin\theta)^{i-k+l}(\cos\theta)^{j+k-l}. \end{aligned} \tag{22}$$

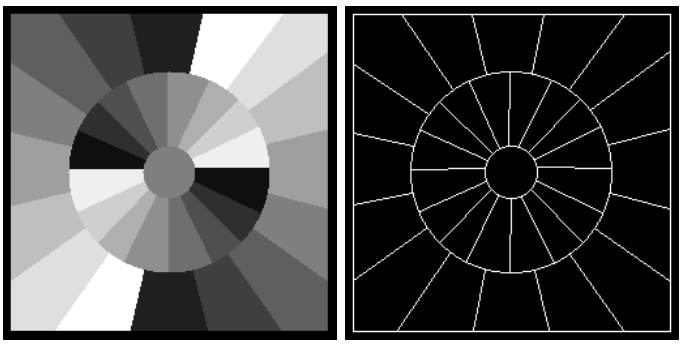

Fig. 4
Input training image (128×128 pixels) and reference edge map.

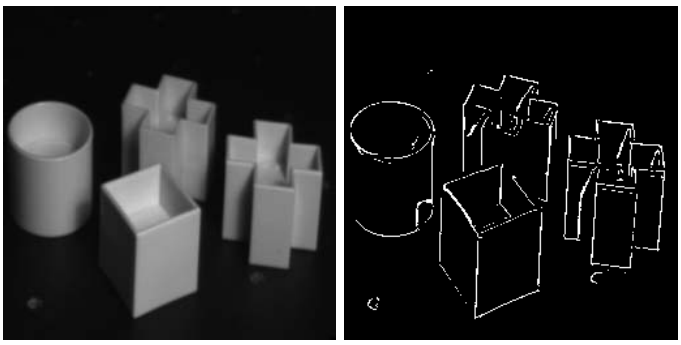

Fig. 5
Test image and result after edge detection by a neural network.

Equation (22) 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 for 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 Figures 6-9 for some 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).

V. 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 Figure 4), others were trained with an image that contained sharp edges as well as blurred ones and additional Gaussian noise. A different image was used as a test set, see Figure 5 for a test result by a neural network edge detector.

VI. Results

As Eq. 22 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 a template as well. First, we will show a brief analysis result for the Kirsch template shown in Section III-A. As can clearly be seen from Figure 6, this template shows no low-pass (averaging) behavior, has a strong first-order gradient operation in diagonal direction, as could be expected, and weak second-order gradient behavior.

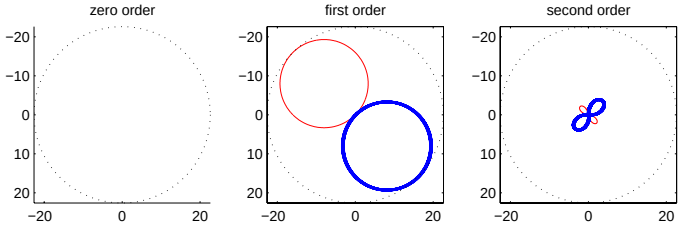

Fig. 6
Low-pass, gradient, and second-order gradient analysis results for the Kirsch edge detector template of Section III-A.

Non-edge detecting templates can also be analyzed. In the case of the line-detecting template of Figure 7, it is clear that it detects horizontal lines, therefor it is not surprising that the second order coefficients are strongest in vertical direction. The zero-order and first-order coefficients are all zero.

Figure 8 shows results for the 4 hidden units of a small neural network edge detector, which was trained with the sharp edges shown in Figure 4. For the purpose of showing the hidden units' weight values graphically, they have been scaled to values between -1 (black) and $+1$ (white).

$$\begin{bmatrix} 1 & 1 & 1 \\ -2 & -2 & -2 \\ 1 & 1 & 1 \end{bmatrix}$$

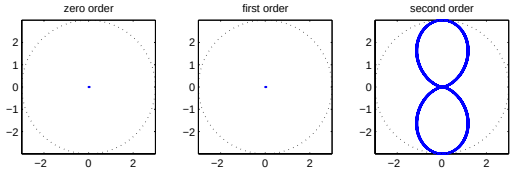

Fig. 7
Low-pass, gradient, and second-order gradient analysis results for the line detector template shown on the left.

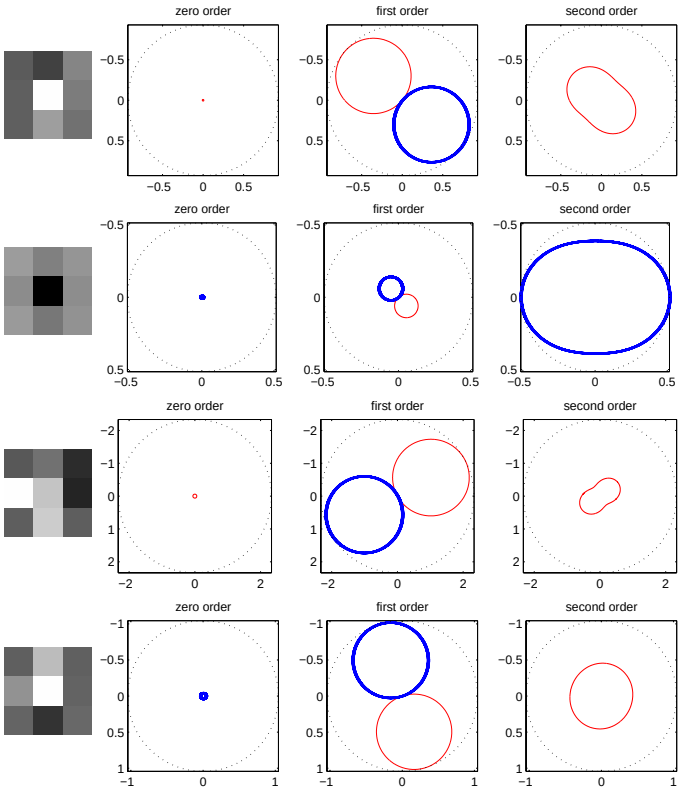

Fig. 8

Weight templates and low-pass, gradient, and second-order gradient analysis results for all 4 hidden units of a $(3\times3)\_4\_1$ neural network edge detector trained with sharp edges.

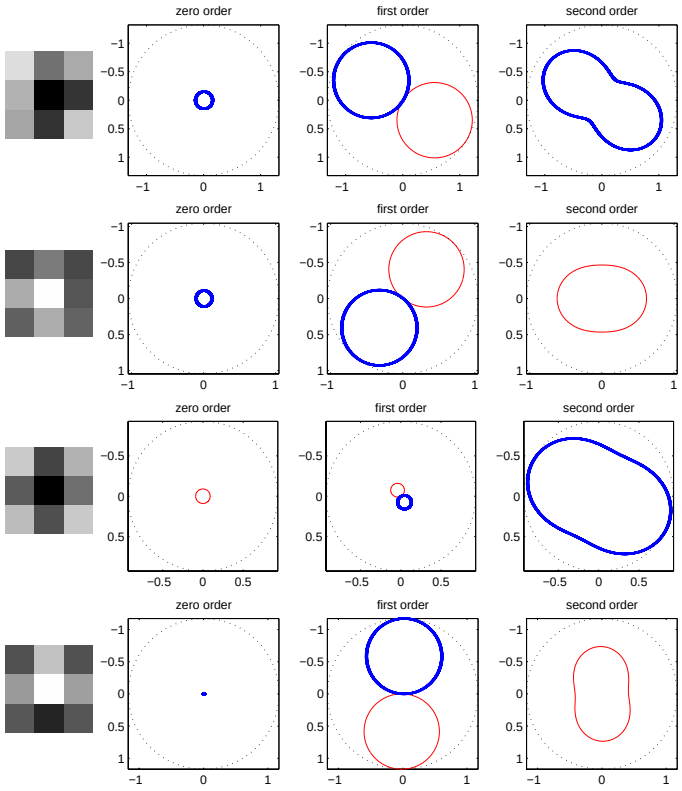

Fig. 9

Weight templates and low-pass, gradient, and second-order gradient analysis results for all 4 hidden units of a $(3\times3)\_4\_1$ neural net edge detector trained with sharp/blurry/noisy edges.

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 all four units do not have 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 Figure 4. The weight templates of the hidden units are shown in Figure 9 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 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 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.

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

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

[3] B.J. van der Zwaag, L. Spaanenburg, and C. Slump (2002), "Analysis of neural networks in terms of domain functions," *Proceedings IEEE Benelux Signal Processing Symposium SPS-2002* (Leuven, Belgium, 21-22 March), pp. 237-240.

[4] B.J. van der Zwaag, C. Slump, and L. Spaanenburg (2002), "Process identification through modular neural networks and rule extraction," in D. Ruan, P. D'hondt, and E.E. Kerre (eds.), *Computational Intelligent Systems for Applied Research: Proceedings of the 5th International FLINS Conference* (Ghent, Belgium, 16-18 Sep.), World Scientific, Singapore, pp. 268-277.