Entropy optimized morphological shared-weight neural networks

Mohamed A. Khabou
Paul D. Gader, MEMBER SPIE
Hongchi Shi
University of Missouri—Columbia
Computer Engineering and Computer Science
201 EBW
Columbia, Missouri 65211
E-mail: gader@ece.missouri.edu

Abstract. Morphological shared-weight neural networks previously demonstrated performance superior to that of MACE filters and standard shared-weight neural networks for target detection. Empirical analysis showed that entropy measures of the morphological shared-weight networks were consistently higher than those of the standard shared-weight neural networks. Based on this observation, an entropy maximization term was added to the morphological shared-weight network objective function. In this paper, target detection results are presented for morphological shared-weight networks trained with and without entropy terms.
 [S0091-3286(99)00502-4]

Subject terms: entropy; morphological shared-weight nets; ATR; SAR; MSTAR.

Paper 980051 received Feb. 13, 1998; revised manuscript received Aug. 27, 1998; accepted for publication Sep. 7, 1998.

1 Introduction

Target detection is a difficult recognition problem, especially when targets are occluded. Matched filters and neural networks are two approaches that have been studied for this application. Previously, we have demonstrated the utility of shared-weight networks for target detection.[1–3]

Standard shared-weight neural networks (SSNNs) were popularized by Le Cun et al. at AT&T[4,5] and were used to perform handwritten-digit recognition. We performed experiments that suggested that a modification of the SSNN based on mathematical morphology, referred to as the morphological shared-weight network (MSNN), performed better than the SSNN both in faster training in the case of digit recognition and in faster training and better detection-versus-false-alarm rates in the case of target detection.[2,3]

Shared-weight neural networks (standard or morphological) learn feature extraction and classification simultaneously. Entropy has been proposed by several researchers as a means of evaluating features. Empirical analysis of the SSNNs and MSNNs developed for target detection revealed that entropy measures based on the features learned by these networks were much higher for the MSNN than the SSNN. Based on this observation, we modified the MSNN's objective function to encourage higher entropy measures. This paper discusses the modification.

First, we describe the structure of shared-weight neural networks and the basic operations of mathematical morphology. Second, we describe the data used in our experiments. We then discuss the entropy measures of previous networks and the modification of the MSNN training algorithm. Finally, we present the experimental results.

2 Architecture of Shared-Weight Networks

A standard shared-weight network, $\mathbf{W}$, is composed of two cascaded subnetworks, called stages: a feature extraction stage $\mathbf{F}$ followed by a feedforward stage $\mathbf{C}$, i.e., $\mathbf{W}=(\mathbf{F}, \mathbf{C})$. (See Fig. 1.) The feature extraction stage $\mathbf{F}$ usually has a two-dimensional array for the input and has local, translation-invariant connections. The layers in this stage perform feature extraction by linear convolution of their inputs with the kernels defined by the local connections. The nodes of the last feature extraction layer are the inputs to $\mathbf{C}$. A more precise description is given in the following paragraph.

Let $\mathbf{L}_1,\ldots,\mathbf{L}_k$ denote the layers of $\mathbf{F}$ (not counting the input layer). Each layer is partitioned into subsets called feature maps, $\mathbf{L}_j=(\mathbf{M}_{j1},\ldots,\mathbf{M}_{jk_j})$. All feature maps in layer $\mathbf{L}_1$ have a single kernel. Each feature map in a feature extraction layer $\mathbf{L}_{j+1}$ with $j\geqslant 1$ has one kernel for each feature map in the previous feature extraction layer $\mathbf{L}_j$. Every node $\mathbf{x}^{\mathbf{M}_{ji}}$ in a feature map $\mathbf{M}_{ji}$ has the same kernel. Thus, the number of free parameters in $\mathbf{M}_{ji}$ is equal to the size of its kernel over its input plus the number of the nodes in the feature map if there is one bias per node. Each node in $\mathbf{M}_{ji}$ is connected to a local region at a certain position in the feature maps $\mathbf{M}_{(j-1)p}$, $p=1,\ldots,k_{(j-1)}$, via the kernel. The regions have the same shape for each node, but are shifted. More precisely, suppose that $\mathbf{x}^{\mathbf{M}_{ji}}$ and $\mathbf{y}^{\mathbf{M}_{ji}}$ are nodes in $\mathbf{M}_{ji}$, and suppose that $\mathbf{x}^{\mathbf{M}_{ji}}$ is connected to n nodes $\mathbf{x}_1^{\mathbf{M}_{(j-1)p}},\ldots,\mathbf{x}_n^{\mathbf{M}_{(j-1)p}}$ in the feature map $\mathbf{M}_{(j-1)p}$. Then $\mathbf{y}^{\mathbf{M}_{ji}}$ is also connected to n nodes $\mathbf{y}_1^{\mathbf{M}_{(j-1)p}},\ldots,\mathbf{y}_n^{\mathbf{M}_{(j-1)p}}$ in the feature map $\mathbf{M}_{(j-1)p}$, and there is a translation $\mathbf{h}$ such that $\mathbf{x}_t^{\mathbf{M}_{(j-1)p}}=\mathbf{y}_t^{\mathbf{M}_{(j-1)p}}+\mathbf{h}$, $t=1,\ldots,n$. Furthermore, the weight between $\mathbf{x}^{\mathbf{M}_{ji}}$ and $\mathbf{x}_k^{\mathbf{M}_{(j-1)p}}$ is the same as that between $\mathbf{y}^{\mathbf{M}_{ji}}$ and $\mathbf{y}_k^{\mathbf{M}_{(j-1)p}}$ for $k=1,\ldots,n$. There may be downsampling. For example, if the input $\mathbf{M}_{(j-1)p}$ is 50×50 and the downsampling rate is $r=2$ for both directions, then the feature maps $\mathbf{M}_{j1},\ldots,\mathbf{M}_{jk_j}$ are 25×25. The translation $\mathbf{h}$ is given by $\mathbf{h}=(\mathbf{x}^{\mathbf{M}_{ji}}-\mathbf{y}^{\mathbf{M}_{ji}})/r$.

A shared-weight network (standard or morphological) takes a gray-scale image, which might be a subimage of a larger image, as input and produces n outputs. Each output O_i represents the network confidence that the input image

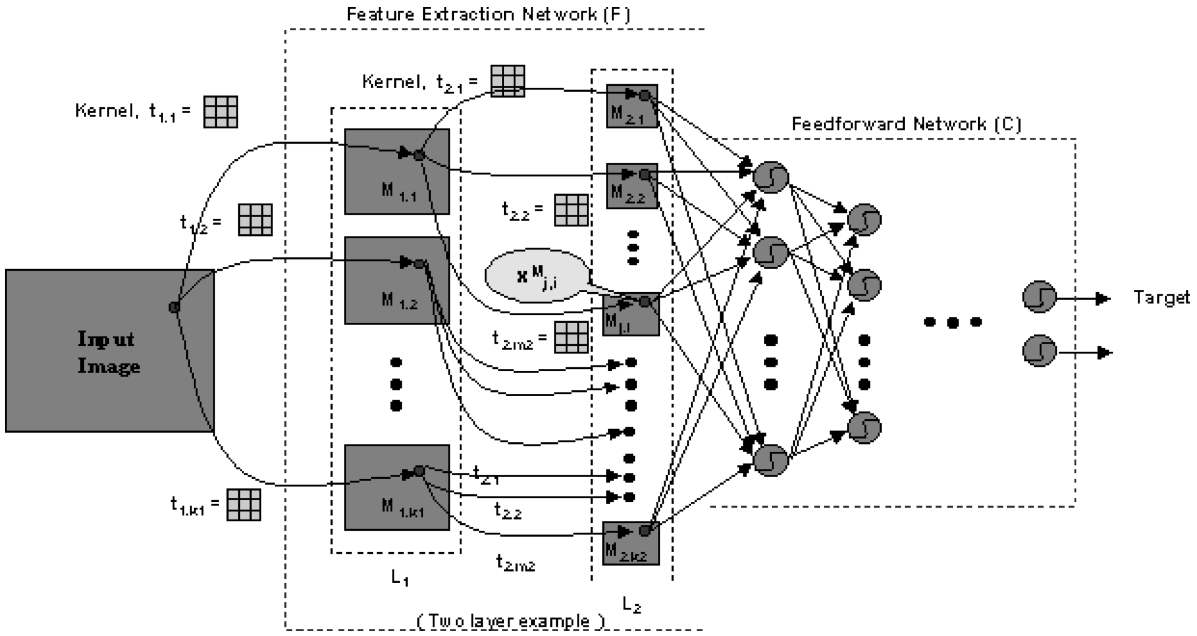


Fig. 1 Architecture of shared-weight networks.

belongs to class *i*. During training in target detection, a shared-weight network takes a subimage as input and produces two output values: target or no-target. For testing, the network scans an entire input scene and generates an output image, the detection plane. The values of the detection plane are proportional to the outputs from the target class node. Automatic target recognition systems require a point, called the target aim point (TAP), at which to aim. The performance of a target detection network can be measured by the location of the TAPs. In our experiments we used the TAP selection algorithm that involved thresholding and modified ultimate erosion (TMUE) described in Ref. 3.

The MSNN differs from the SSNN in the feature-map nodes. In the SSNN, these nodes perform linear convolution followed by evaluation of a sigmoid function. In the MSNN, they perform the gray-scale morphological hit-miss transform.[3]

We provide brief definitions of gray-scale morphological operations here. A full discussion can be found in Ref. 6. The basic morphological operations of erosion and dilation of an image f by a structuring element (SE) g are:

$$\text{erosion:} \quad (f \ominus g)(x) = \min\{f(z) - g_x(z) : z \in D[g_x]\}, \quad (1)$$

$$\text{dilation:} \quad (f \oplus g)(x) = \max\{f(z) - g_x^*(z) : z \in D[g_x^*]\}, \quad (2)$$

where $g_x(z) = g(z-x)$, $g^*(z) = -g(-z)$, and $D[g]$ is the domain of g. The gray-scale hit-miss transform, defined as

$$f \otimes (h,m) = (f \ominus h) - f(\oplus m^*), \quad (3)$$

measures how a shape h fits under f using erosion and how a shape m fits above f using dilation.[3] High values indicate good fits (Fig. 2). The hit-miss transform is independent of shifting in gray scale, i.e.,

$$(f + \lambda) \otimes (h,m) = f \otimes (h,m). \quad (4)$$

3 Experimental Data

We conducted two sets of experiments using two different data sets. In the first set of experiments we used the data set called the Blazer data set collected at Eglin Air Force Base. It consists of 256×256 visible images of a parking lot filled with vehicles. The objective is to detect a particular vehicle, a Blazer, which appears in all the images. We used 36 images for training and 52 images for testing. The Blazer appears nonoccluded in all the training images. In the testing images the level of occlusion ranges from no occlusion to complete occlusion. This data set is completely described in previous publications.[1,3] Samples are shown in Fig. 3.

In the second set of experiments we used the data set called the MSTAR data set collected by the Sandia National Laboratory and publicly released on CD-ROM by the U.S. Air Force Wright Laboratory.[7] The data are divided into two separate sets: a target data set and a clutter data set. The target data set consists of 128×128 synthetic aperture radar (SAR) images of military objects—T72 tanks, and BMP2 and BTR70 armored personnel carriers (APCs)—divided into two subsets: training and testing. The MSTAR data set contains three different T72s. In our experiments we used the SAR images of the T72 tank with serial number sn_s7. Samples are shown in Fig. 4. The clutter data set consists of 100 SAR images of clutter scenes collected at various rural and urban locations. The images are of different sizes, but most of them are about 1784×1474.

The objective of this set of experiments is to detect T72 tanks in clutter scenes. We used 36 training T72 tank im-

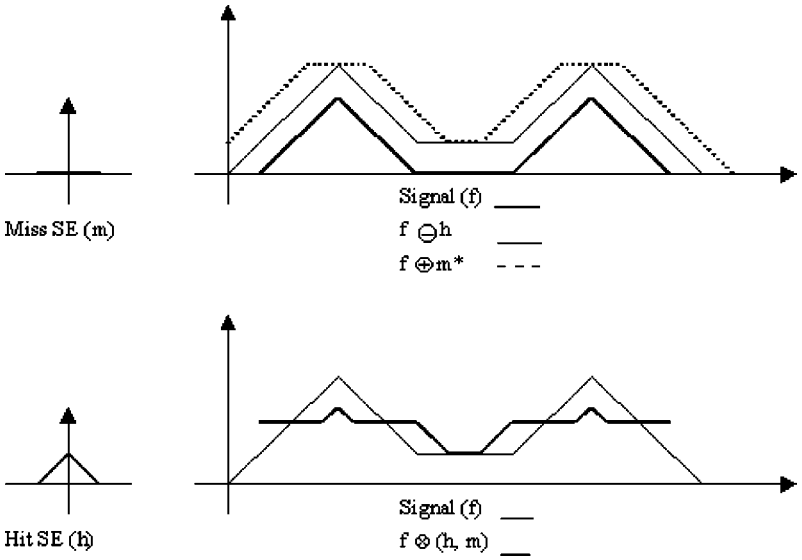


Fig. 2 Example of a hit-miss transform.

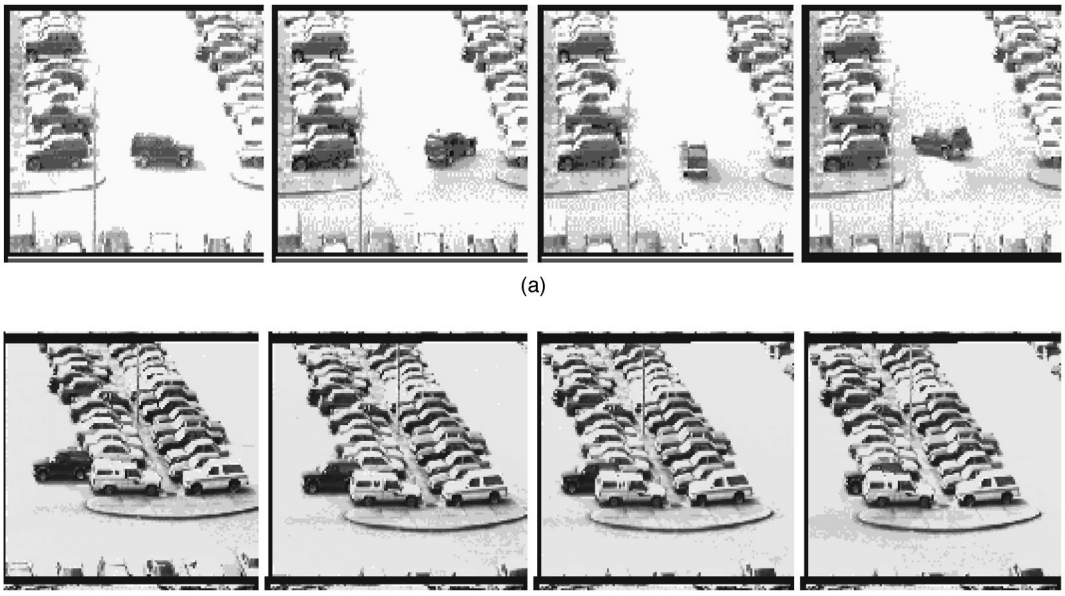


Fig. 3 Sample (a) training and (b) testing images from the Blazer data set.

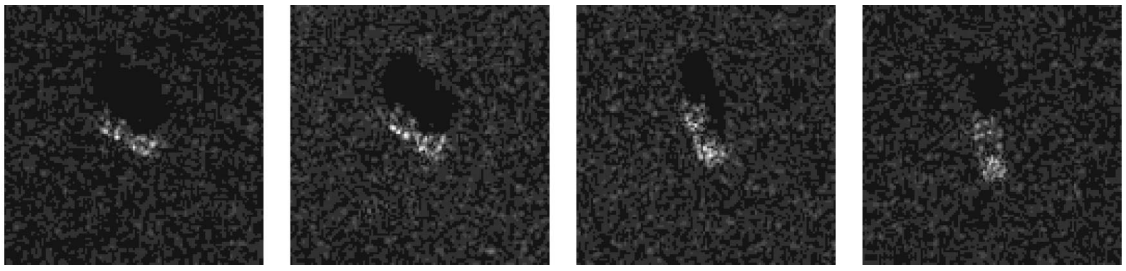


Fig. 4 Sample T72 tank images from the MSTAR data set.

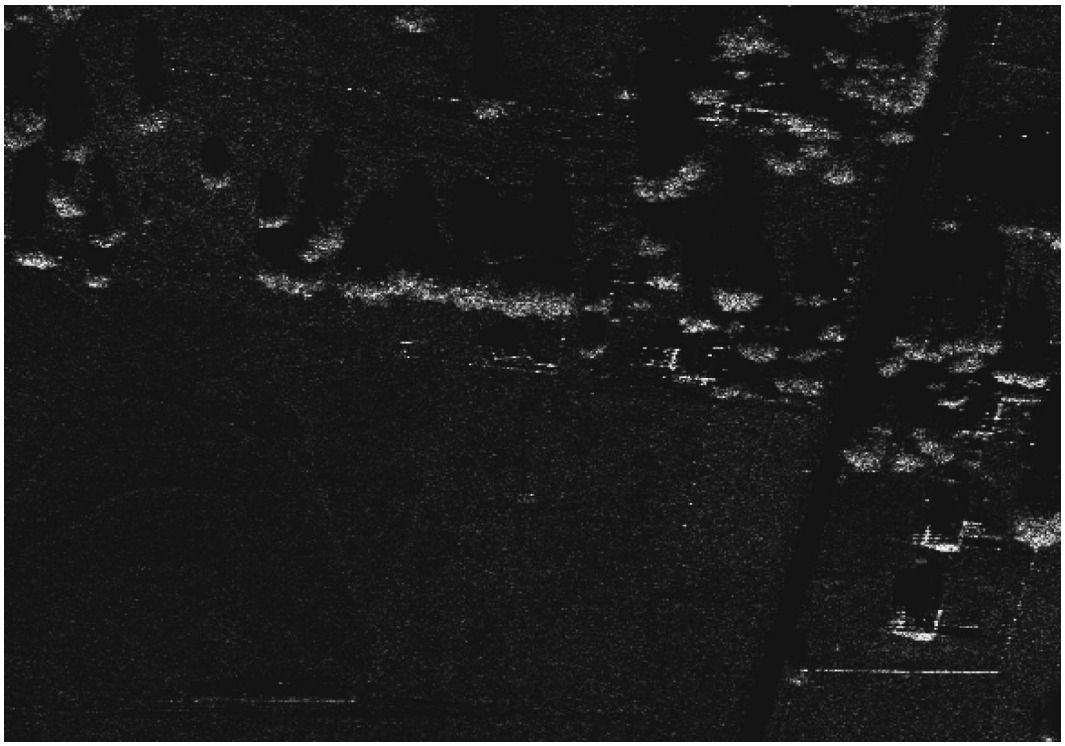


Fig. 5 Sample MSTAR background scene used in training.

ages with 10-deg orientation increments for target training, and we selected 10 512×512 subimages from the clutter-scenes for background training. These background scenes were selected to include different features (fields, trees, shadows, houses, etc.) to represent the different rural and urban backgrounds available in the MSTAR scenes. A sample is shown in Fig. 5.

Because the MSTAR data set did not include scenes with targets, we created 10 testing scenes by embedding 200 T72 testing tanks into clutter scenes. Each testing scene contained 20 T72 tanks. The tanks were embedded in *hard* positions, i.e., near tree shadows and behind trees; however, none of the tanks were occluded. Sample testing scenes are shown in Fig. 6.

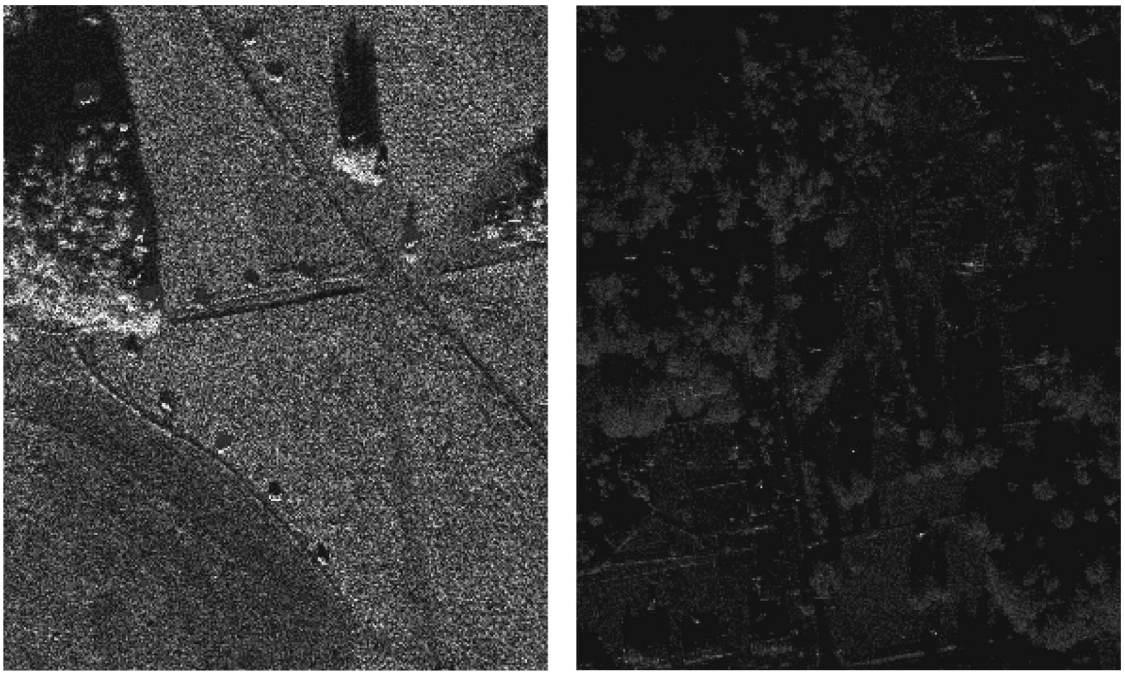


Fig. 6 Sample rural and urban testing scenes.

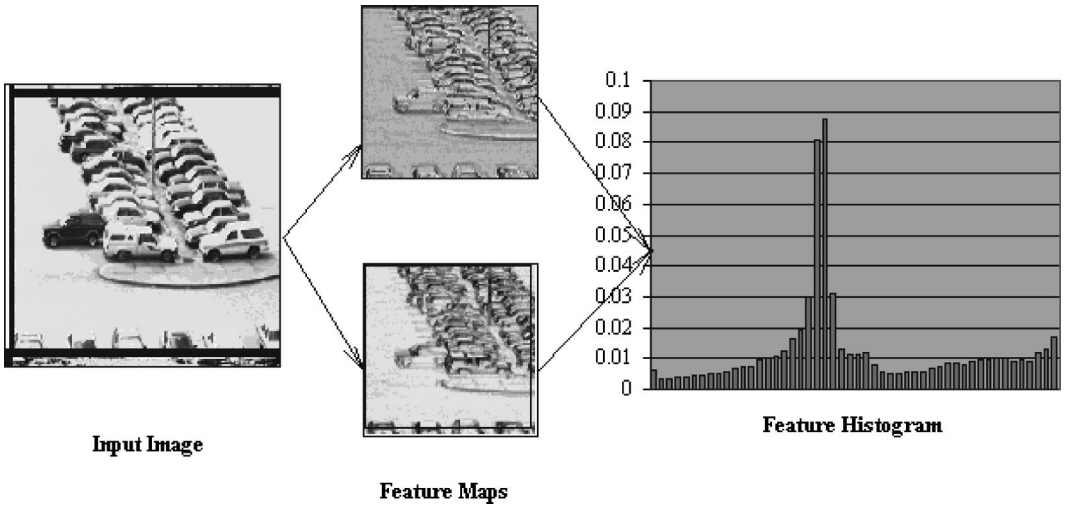


Fig. 7 Generation of a feature histogram.

4 Entropy of Existing Networks

Given a random process with outcomes X_1, $X_2,\ldots,X_n$ and their corresponding probabilities p_1, $p_2,\ldots,p_n$, a measure of information in these outcomes is given by Shannon's entropy, defined as

$$E=-\sum_{i=1}^{n} p_i \ln p_i . \tag{5}$$

The function E has a maximum value of $\ln n$ when $p_1=p_2=\cdots=p_n$, and a minimum value of zero when $p_i=1$ and $p_j=0$ for $j\neq i$.

Entropy has been used to measure the quality of generated features.[8–10] Features with higher entropy values are favored over others. The rationale is that features with higher entropy values are ''smoother,'' are ''more probable,'' and ''assume less'' according to Shannon's interpretation of entropy as an information measure.[9] The maximum entropy principle, stated briefly, is that when we make inferences based on incomplete information, we should draw them from the probability distribution that has the maximum entropy permitted by the information we have.

We define the entropy of a shared-weight network with one feature extraction layer as a function of an input image. We define it as the entropy of the normalized histogram of all the values of the feature maps. The following paragraph provides a precise definition.

Let $\mathbf{W}$ be a shared-weight network with one feature extraction layer $\mathbf{L}$ containing k feature maps, $\mathbf{L}=(\mathbf{M}_1,\mathbf{M}_2,\ldots,\mathbf{M}_k)$. For a given input image $\mathbf{I}$, let $\mathbf{M}_j(\mathbf{I},\mathbf{y})$ denote the output of the j'th feature map at position $\mathbf{y}$. Let $L_{\min}=\min_{j,\mathbf{y}}\mathbf{M}_j(\mathbf{I},\mathbf{y})$ and $L_{\max}=\max_{j,\mathbf{y}}\mathbf{M}_j(\mathbf{I},\mathbf{y})$. Let $n_{\text{bins}}>0$ be an integer that represents the number of bins in the normalized feature histogram. Let $d=(L_{\max}-L_{\min})/n_{\text{bins}}$ and $f_i=|\{\mathbf{M}_j(\mathbf{I},\mathbf{y}):j=1,...,k \text{ and } L_{\min}+(i-1)d\leq\mathbf{M}_j(\mathbf{I},\mathbf{y})\leq L_{\min}+id\}|$, where $|\cdot|$ denotes set cardinality. Let $p_i=f_i/\Sigma_{j=1}^{n_{\text{bins}}}f_j$. Then $\mathbf{H}=\{p_1,p_2,\ldots,p_{n_{\text{bins}}}\}$ is the normalized frequency histogram of the values $\{\mathbf{M}_j(\mathbf{I},\mathbf{y}):j=1,...,k\}$. The histogram values approximate the probability density function of the feature values (Fig. 7).

The entropy is then defined as

$$E=-\sum_{i=1}^{n_{\text{bins}}} p_i \ln p_i . \tag{6}$$

Figure 8 shows the MSNN and SSNN entropy values for sample training and testing images from the Blazer data set.

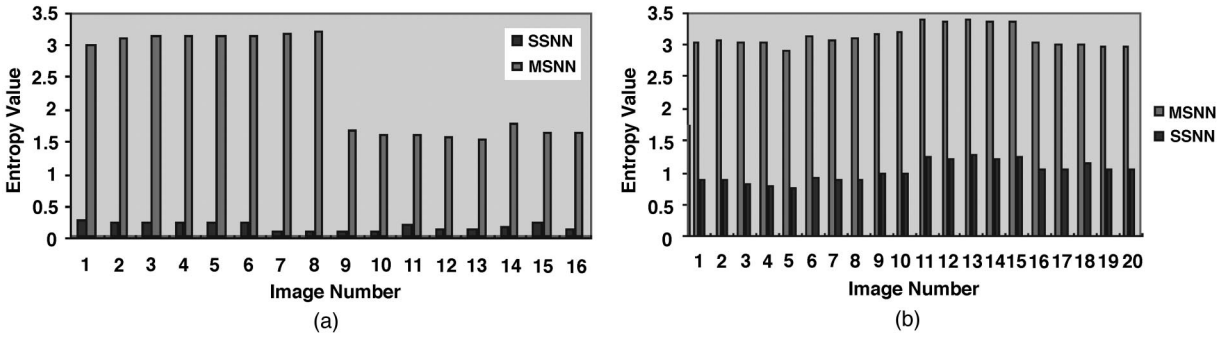


Fig. 8 MSNN and SSNN entropy values for sample (a) training and (b) testing images from the Blazer data set.

As can be seen, features generated by the MSNN consistently have higher entropy values than those generated by the SSNN for both training and testing images. This implies that the MSNN features carry more information than those generated by the SSNN and may explain the better performance of the MSNN in target detection.

5 Entropy Optimization

Since the MSNN outperformed the SSNN, and the MSNN entropy values were consistently higher than those of the SSNN, we hypothesize that higher entropy values imply better performance. Based on this hypothesis, we added an entropy maximization term to the original MSNN objective function. The original MSNN objective function is the standard sum of squared errors, which measures how close the network outputs are to the desired ones:

$$E_1=\sum_{i=1}^{N}[d_i-f(x_i)]^2, \tag{7}$$

where N is the number of training samples, x_i is the i'th training sample, $f(x_i)$ is the network output for the sample x_i, and d_i is the desired output for x_i.

The added entropy maximization term is

$$E_2=-\sum_{i=1}^{N}\sum_{j=1}^{n_{\text{bins}}} p_{ij}\,\ln p_{ij}=\sum_{i=1}^{N} E_{i2}, \tag{8}$$

where p_{ij} is the j'th value of the feature histogram for the i'th training sample. Thus, the new objective function to minimize is

$$E=\sum_{i=1}^{N}\left\{[d_i-f(x_i)]^2+\lambda\sum_{j=1}^{n_{\text{bins}}} p_{ij}\,\ln p_{ij}\right\}=E_1-\lambda E_2, \tag{9}$$

where λ is a coefficient that determines the tradeoff between E_1 and E_2.

6 Derivation of Learning Algorithm

In this section we first define the notation and node activation functions for the feature extraction stage of the MSNN; then we provide the learning rules for the feature extraction based on the entropy-modified objective function.

6.1 *Notation*

Here we define the notation and node activation functions for the feature extraction stage of the MSNN. Let $\mathbf{I}$ denote the input image, and $t_y^h(\mathbf{x})$ $[t_y^m(\mathbf{x})]$ the hit [miss] structuring element weight connecting input node $\mathbf{x}$ to feature-map node $\mathbf{y}$.

The net input for the hit operation is defined as

$$\text{net}_{\mathbf{y}}^h=\min_{x\in D[t_{\mathbf{y}}]}[\mathbf{I}(\mathbf{x})-t_{\mathbf{y}}^h(\mathbf{x})], \tag{10}$$

and the net input for the miss operation is defined as

$$\text{net}_{\mathbf{y}}^m=\max_{\mathbf{x}\in D[t_{\mathbf{y}}]}[\mathbf{I}(\mathbf{x})-t_{\mathbf{y}}^m(\mathbf{x})]. \tag{11}$$

The output of node $\mathbf{y}$ is defined as the hit-miss transform:

$$\text{net}_{\mathbf{y}}=\text{net}_{\mathbf{y}}^h-\text{net}_{\mathbf{y}}^m. \tag{12}$$

6.2 *Learning Rule*

The learning rules for the feature extraction weights of MSNN are based on gradient descent. To use gradient descent, we need to find $\partial E/\partial t_{\mathbf{y}}^h=\partial E_1/\partial t_{\mathbf{y}}^h-\lambda\ \partial E_2/\partial t_{\mathbf{y}}^h$ and $\partial E/\partial t_{\mathbf{y}}^m=\partial E_1/\partial t_{\mathbf{y}}^m-\lambda\ \partial E_2/\partial t_{\mathbf{y}}^m$. The derivatives $\partial E_1/\partial t_{\mathbf{y}}^h$ and $\partial E_1/\partial t_{\mathbf{y}}^m$ are found in Ref. 3.

In the following paragraph we only provide the derivation of $\partial E_2/\partial t_{\mathbf{y}}^h$, since $\partial E_2/\partial t_{\mathbf{y}}^m$ is similar. Since $E_2=\sum_{i=1}^{N} E_{i2}$, we derive $\partial E_{i2}/\partial t_{\mathbf{y}}^h$ and set $\partial E_2/\partial t_{\mathbf{y}}^h=\sum_{i=1}^{N}\partial E_{i2}/\partial t_{\mathbf{y}}^h$.

Let $S_{ij}=p_{ij}\,\ln p_{ij}$. Then

$$E_{i2}=-\sum_{j=1}^{n_{\text{bins}}} p_{ij}\,\ln p_{ij}=-\sum_{j=1}^{n_{\text{bins}}} S_{ij}. \tag{13}$$

For the hit template,

$$\frac{\partial E_{i2}}{\partial t_{\mathbf{y}}^h}=-\sum_{j=1}^{n_{\text{bins}}}\frac{\partial S_{ij}}{\partial t_{\mathbf{y}}^h}=-\sum_{j=1}^{n_{\text{bins}}}\frac{\partial S_{ij}}{\partial p_{ij}}\cdot\frac{\partial p_{ij}}{\partial t_{\mathbf{y}}^h}=-\sum_{j=1}^{n_{\text{bins}}}\left[(1+\ln p_{ij})\,\frac{\partial p_{ij}}{\partial t_{\mathbf{y}}^h}\right]. \tag{14}$$

To find $\partial p_{ij}/\partial t_{\mathbf{y}}^h=(\partial p_{ij}/\partial\,\text{net}_{\mathbf{y}})\,\partial\,\text{net}_{\mathbf{y}}/\partial t_{\mathbf{y}}^h$, we assume that the features have a Gaussian distribution, so that the j'th value of the normalized feature histogram can be computed using a Gaussian function, i.e.,

$$p_{ij}=g(\text{net}_{\mathbf{y}})=c\,\frac{1}{\sqrt{2\pi}\sigma}\,e^{-(1/2)[(\text{net}_{\mathbf{y}}-\mu)/\sigma]^2}, \tag{15}$$

where $\text{net}_{\mathbf{y}}\in\text{bin}_j$ of the feature histogram, μ and σ are the average and standard deviation of feature values, respectively, and c is a normalization factor so that $\sum_{j=1}^{n_{\text{bins}}}p_{ij}=1$. We find this assumption reasonable because feature histogram plots for many sample training and testing images from the Blazer data set had a Gaussian shape (Fig. 7).

We take

$$\frac{\partial p_{ij}}{\partial\,\text{net}_{\mathbf{y}}}=\begin{cases}g'(\text{net}_{\mathbf{y}}) & \text{if } \text{net}_{\mathbf{y}}\in\text{bin}_j,\\ 0 & \text{otherwise,}\end{cases} \tag{16}$$

Table 1 Performance of the entropy-trained and non-entropy-trained MSNNs on the Blazer data set.

Network pair no.	Entropy-trained MSNN			Non-entropy-trained MSNN	
	Detections	False Alarms	λ	Detections	False Alarms
1	30	25	0.1	26	31
2	36	20	0.01	35	26
3	34	22	0.0001	29	23
4	32	23	0.0001	30	22
5	36	26	0.0001	35	24
Average	33.6	23.2	—	31	25.2

which makes

$$\frac{\partial E_{i2}}{\partial t_{\mathrm{y}}^{h}} = -(1+\ln p_{ij})\, g'(\mathrm{net}_{\mathrm{y}})\, \frac{\partial\ \mathrm{net}_{\mathrm{y}}}{\partial t_{\mathrm{y}}^{h}}. \tag{17}$$

The expression for $\partial\ \mathrm{net}_{\mathrm{y}}/\partial t_{\mathrm{y}}^{h}$ is derived in Ref. 3.

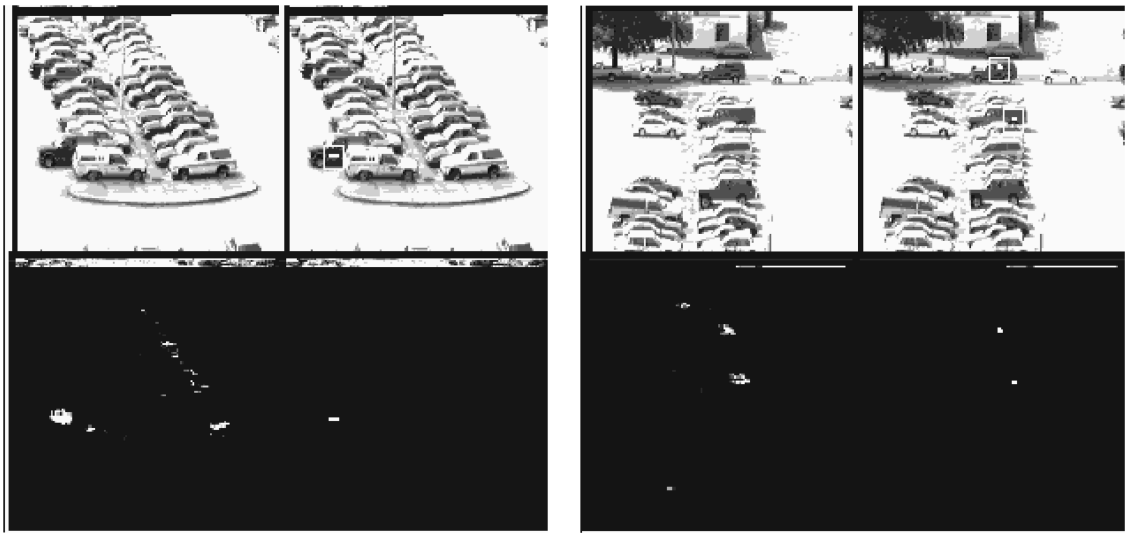
7 Experimental Results

We conducted two sets of experiments. We used the Blazer data set in the first set, and the MSTAR data set in the second. The following subsections describe the results.

7.1 *Experiments on the Blazer Data Set*

Five MSNN pairs were trained using the 36 training images of the Blazer data set. The first MSNN of each pair was trained with the added entropy maximization term, and the second was trained without it. We used five MSNN pairs to obtain a more accurate comparison and to reduce statistical errors. Each network pair was initialized with the same random weights and used the same training parameters, i.e., learning rate and momentum. All networks took about the same number of epochs to train (about 790 epochs) and attained similar RMS errors when fully trained. All networks took a 76×46 image as input and had one feature-extraction layer with two feature maps. The downsampling rate from the input image to the feature layer was 2, i.e., each feature map had a size of 38×23. The sizes of the hit-miss kernels of both feature maps were 5×5. The feed-forward stages of the networks were composed of one hidden layer with three nodes and one output layer with two nodes. This network architecture was chosen because it produced the best results for the *non-entropy-trained* MSNN among various configurations we experimented with.

We tested the performance of these networks on the 52 testing images of the Blazer data set. Table 1 gives a summary of the results. As can be seen, the entropy-trained MSNNs detected more Blazers and had, in general, fewer false alarms. Figure 9(a) shows an example where an entropy-trained MSNN detected the Blazer even though it



(a) (b)

Fig. 9 Sample output planes for (a) a correct detection and (b) a false alarm. The upper left quadrant is the input image, the lower left quadrant is the output plane of the network, the lower right quadrant is the result of the TAP selection algorithm, and the upper right quadrant is a superposition of the TAP squares on the input image.

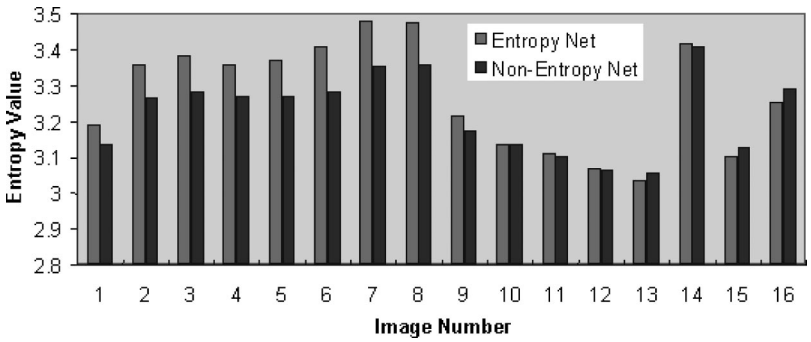


Fig. 10 Entropy values for sample training images from the Blazer data set using an entropy-trained and a non-entropy-trained MSNN.

was half occluded. Figure 9(b) shows an example where the same network produced a false alarm. Notice that in this example the Blazer is hardly visible (parked behind the white van in the middle of the image). The white boxes indicate the positions of the TAPs.

For comparison, we calculated the entropy values for the features generated by one entropy-trained and one non-entropy-trained MSNN for all training and testing images of the Blazer data set. Entropy values for sample training images are plotted in Fig. 10. Except for three training images and one testing image, the features generated by the entropy-trained MSNN had higher entropy values. This suggests that features generated by entropy-trained MSNNs carry more information than those generated by non-entropy-trained MSNNs and may explain the better performance of the former.

7.2 *Experiments on the MSTAR Data Set*

We tried two different network structures. The feature extraction stage of the first structure consisted of a 100×100 input and a single feature extraction layer with four feature maps. The downsampling rate from the input to the feature layer was 4, i.e., each feature map had a size of 25×25. The sizes of the hit-miss structuring elements were 5×5, 5×5, 3×3, and 3×3. The feedforward stage had two hidden layers with two and three hidden units and a two-unit output layer. This network structure was chosen because it produced the best results for the non-entropy-trained MSNN among various configurations we experimented with. The second structure we tried was the same as the first except that the downsampling rate from the input to the feature layer was 2, i.e., each feature map had a size of 50×50. We tried this second structure to see the effect of the added entropy maximization term on the MSNN performance using a ''neutral'' structure. We also wanted to see the effect of a more complex structure on the MSNN performance.

For each network structure we trained five MSNN pairs. The first MSNNs of the pairs were trained with the added entropy maximization term, and the second MSNNs were trained without it. We used five pairs for each structure to obtain a more accurate comparison and to reduce statistical errors. The MSNNs of each pair were initialized with the same set of weights. All networks were trained for 500 epochs and used the same learning rate of 0.001 and the same momentum of 0.8.

The MSNNs were tested using all 10 testing scenes. Figure 11 shows the detection output produced by an entropy-trained MSNN on testing scene 1, and Fig. 12 shows the output of a non-entropy-trained MSNN on the same scene. In both figures, the upper left quadrant is the input image, the lower left quadrant is the output plane of the network, the lower right quadrant is the result of the TAP selection algorithm, where the small *T*'s indicate the positions of the detected targets, and the upper right quadrant is a superposition of the TAP squares on the input image.

Figure 13 shows the average number of detections per scene achieved by the entropy-trained and non-entropy-trained MSNNs using different threshold values in the TAP selection algorithm for the first and second structures. Figure 14 shows the average number of false alarms per scene for the first and second structures. Figure 15 shows the relationship between the average number of detections per scene and the average number of false alarms per scene for the first and second structures.

As can be seen from Figs. 13–15, the entropy-trained MSNNs of the first structure performed almost the same as the non-entropy-trained MSNNs. However, the entropy-trained MSNNs of the second structure produced more detections and less false alarms than the non-entropy-trained MSNNs.

The first structure we tried was the one that allowed the best detection results for the non-entropy-trained MSNN. The addition of the entropy maximization term to the objective function did not help the performance of the MSNN, nor did it hurt it. This structure can be viewed as the least complex structure that is able to solve the problem. The entropy maximization of the MSNN can be seen as a regularization problem where the entropy maximization term added to the objective function plays the rule of a stabilizer. It has been shown in previous publications that regularization techniques would have little or no effect on the least complex system that can solve a problem. The second structure we tried was not the most efficient, and hence the effect of the entropy maximization technique was visible in there being more detections and less false alarms.

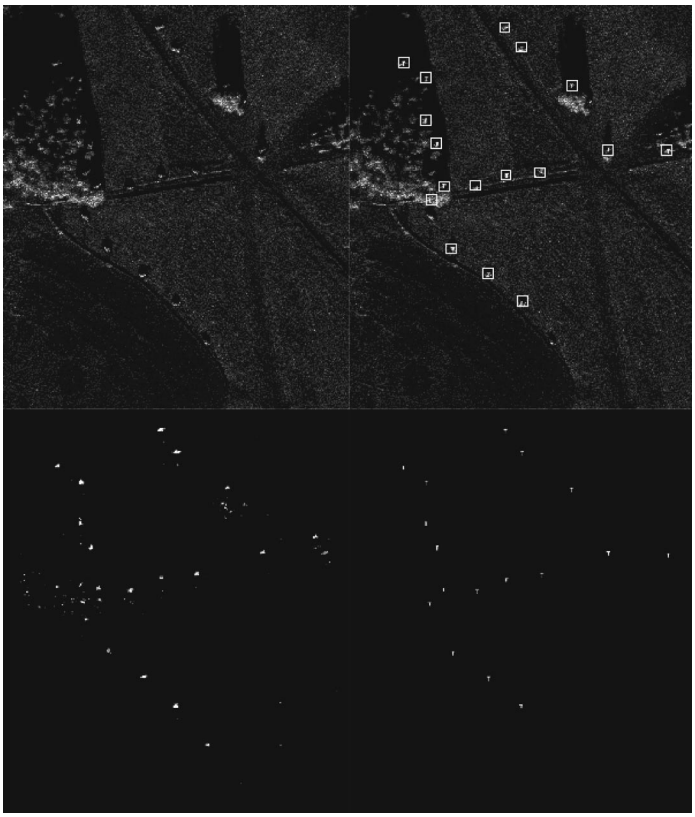


Fig. 11 Detection output of an entropy-trained MSNN on scene 1 (15 detections and 2 false alarms).

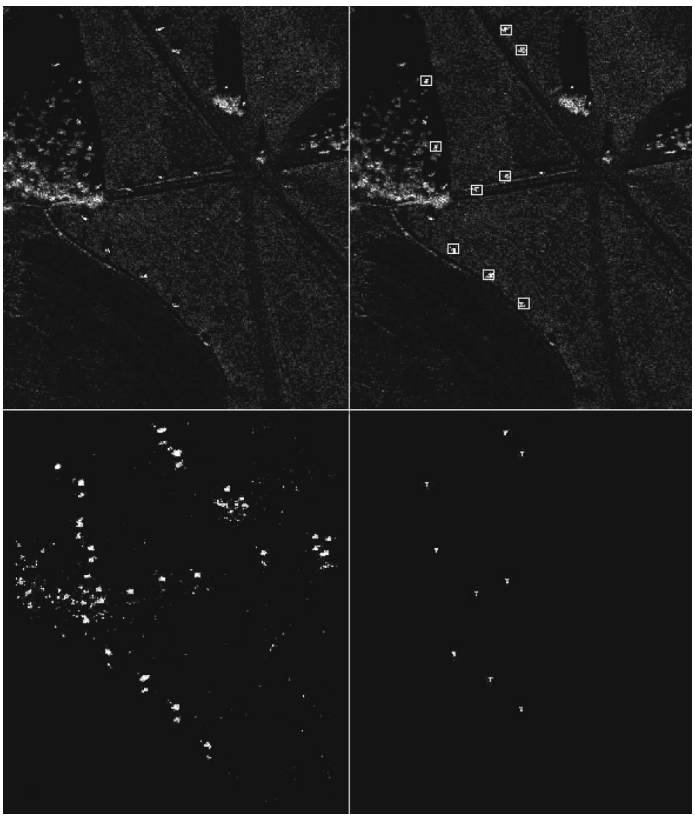


Fig. 12 Detection output of a non-entropy-trained MSNN on scene 1 (9 detections and 0 false alarms).

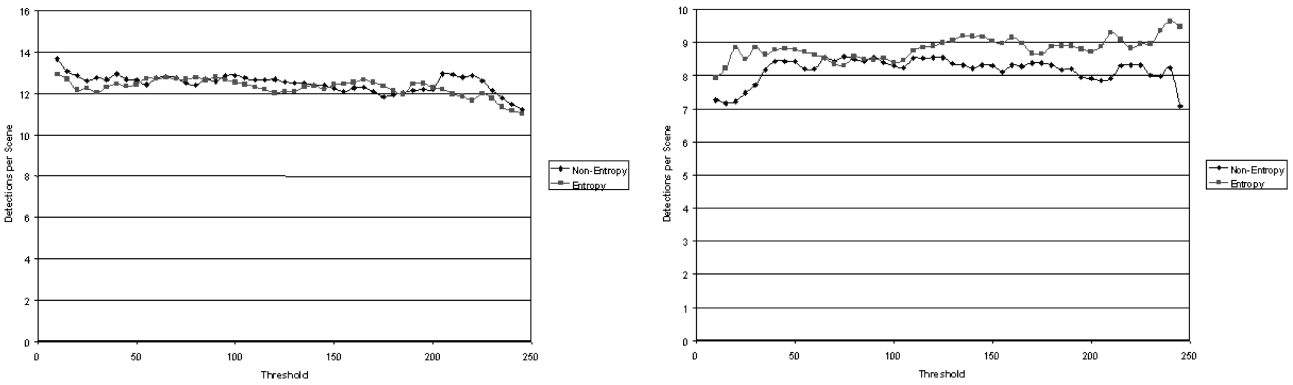


Fig. 13 Average number of detections per scene versus thresholds for entropy-trained and non-entropy-trained MSNNs: (a) first structure, (b) second structure.

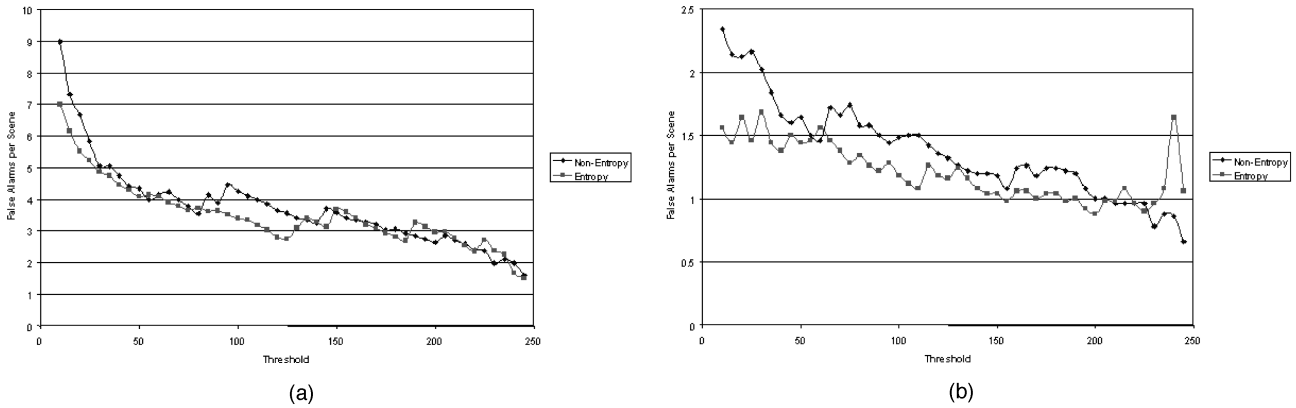


Fig. 14 Average number of false alarms per scene versus thresholds for entropy-trained and non-entropy-trained MSNNs: (a) first structure, (b) second structure.

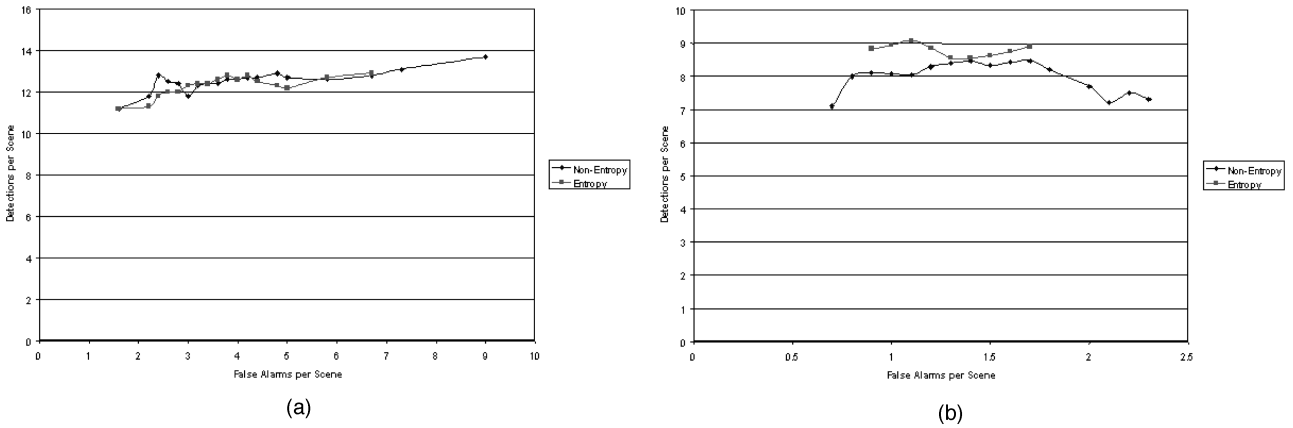


Fig. 15 Average number of detections per scene versus average number of false alarms per scene for entropy-trained and non-entropy-trained MSNNs: (a) first structure, (b) second structure.

8 Conclusion

We derived a new learning rule for MSNN based on an entropy-maximization objective function. We compared the performance of entropy-trained and non-entropy-trained MSNNs at detection problems using two different data sets.

In the experiments using the Blazer data set, the entropy-trained MSNNs detected more targets and produced, on average, less false alarms than the non-entropy-trained MSNNs. The features generated by the entropy-trained MSNNs seem to carry more information and hence may explain the better detection rates. In the experiments using the MSTAR data set, the entropy-trained MSNNs produced more detections and less false alarms than the non-entropy-trained MSNNs when the network structure was not the least complex to solve the problem. However, when the structure was optimized for the non-entropy-trained MSNN, the entropy-maximization technique did neither improve or hurt the performance of the MSNN.

The MSNN was reasonably fast at scanning input scenes. Implemented on a DEC workstation, it took the MSNN about 0.3 s to scan a 256×256 Blazer scene and 23 s to scan a 1784×1474 MSTAR scene.

Future work includes investigating the existence of an optimal value of λ and the implementation of the MSNN on a parallel computer for real-time detection. We are also looking into the application of the entropy maximization approach to the SSNN. Since the feature entropy values for the non-entropy-trained MSNN were already near the highest possible value ($\ln 100 \approx 4.605$ for $n_{\text{bins}} = 100$), the opportunity for improvement was somehow limited. In contrast, the SSNN entropy values were low and the opportunity for improvement is large (Fig. 8).

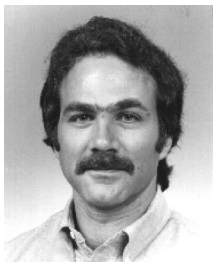
Acknowledgment

This effort is partially sponsored by the Wright Laboratory/MNGA, Air Force Materiel Command, USAF, under grant F08630-96-1-0005. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes not withstanding any copyright notation thereon.

References

1. P. Gader, J. Miramonti, Y. Won, and P. Coffield, ''Segmentation free shared weight networks for automatic vehicle detection,'' *Neural Networks* **8**, 1457–1473 (1995).
2. P. Gader, Y. Won, and M. Khabou, ''Image algebra networks for pattern classification,'' in *Image Algebra and Morphological Image Processing V*, San Diego, CA, July, *Proc. SPIE* **2300**, 157–168 (1994).
3. Y. Won, P. Gader, and P. Coffield, ''Morphological shared-weight networks with applications to automatic target recognition,'' *IEEE Trans. Neural Netw.* **8**, 1195–1203 (1997).
4. Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, ''Back-propagation applied to handwritten ZIP code recognition,'' *Neural Comput.* **1**(4), 541–551 (1989).
5. Y. Le Cun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L. Jackel, and H. Baird, ''Constrained neural network for unconstrained handwritten digit recognition,'' in *Proc. First Int. Workshop on Frontiers in Handwriting Recognition*, Montreal (April 1990).
6. E. Dougherty, *An Introduction to Morphological Image Processing*, SPIE, Bellingham, WA (1992).
7. A. Dixon, ''MBVLab data homepage,'' World Wide Web page http://www.mbvlab.wpafb.af.mil/public/MBVDATA/.
8. P. Gader and M. Khabou, ''Automatic feature generation for handwritten digit recognition,'' *IEEE Trans. Pattern. Anal. Mach. Intell.* **18**(12), 1256–1262 (1996).
9. E. Jaynes, ''On the rationale of maximum entropy methods,'' *Proc. IEEE* **70**(9), 939–952 (1982).
10. D. Mensa, *High Resolution Radar Cross-Section Imaging*, Artech House (1991).
11. N. Theera-Umpon, M. Khabou, P. Gader, J. Keller, H. Shi, and H. Li, ''Detection and classification of MSTAR objects via morphological shared-weight neural networks,'' in *Algorithms for SAR Imagery V*, Orlando, FL, April, *Proc. SPIE* **3370**, 530–540 (1998).

Mohamed A. Khabou received his master's degree in electrical engineering from the University of Missouri–Columbia in 1993. He is currently working on his PhD in electrical engineering at the same university. His research interests include handwriting recognition, automatic target recognition, mathematical morphology, and regularization theory.



Paul D. Gader received his PhD in applied mathematics from the University of Florida in 1986. Since then he worked as a senior research scientist at Honeywell Systems and Research Center, an assistant professor of mathematics at the University of Wisconsin–Oshkosh, and a research engineer and manager at the Environmental Research Institute of Michigan (ERIM). He is currently an associate professor in the Department of Computer Engineering and Computer Science at the University of Missouri–Columbia. Dr. Gader has worked on a wide variety of basic and applied research problems, including automatic target recognition, land-mine and unexploded-ordnance detection, handwriting recognition and document analysis systems, mathematical morphology in image processing and object recognition, fuzzy sets in computer vision, medical imaging, and applied mathematics.

Hongchi Shi is an assistant professor of computer engineering and computer science at the University of Missouri-Columbia. He received his PhD degree in computer and information sciences from the University of Florida in 1994. His research interests include parallel and distributed computing, image processing and computer vision, and neural networks. He has served on the program committee of several international conferences. He chairs the SPIE annual conference on Parallel and Distributed Methods for Image Processing.