Ming LIU*, Hao-yuan DU, Yue-jin ZHAO, Li-quan DONG, Mei HUI

Image Small Target Detection based on Deep Learning with SNR Controlled Sample Generation

Abstract: A small target detection method based on deep learning is proposed. First, random background parts are sampled from some cloud-sky images. Then, random generated target spots are added to the backgrounds with controlled signal to background noise ratio (SNR) to generate target samples. Then training and testing results show that the performance of deep nets is superior to tradition small target detection techniques and the selection of sampling SNR has an important effect on nets training performances. SNR = 1 is a good selection for deep nets training, not only for small target detection, but also for other applications.

Keywords: small target detection; Neural Network; Deep learning; SNR control

1 Introduction

Image small target detection plays a crucial role in infrared warning and tracking systems which requires that small targets on background such as cloud-sky or sea-sky can be detected effectively. There have been many approaches to resolve this problem. Some papers provided methods by filtering or morphology. A high-pass template filter was designed for real-time small target detection by Peng and Zhou [1]. In order to detect an infrared small target, Yang et al. presented an adaptive Butterworth high-pass filter (BHPF) [2]. Wang et al. provided a real-time small target detection method based on the cubic facet model [3]. Hilliard put forward a low-pass IIR filter to predict clutter [4]. Xiangzhi Bai, et al. used top-hat transformation on small target detection [5]. Some other papers proposed methods of classifier. Wang etc., proposed a detection method based on high pass filter and least squares support vector machine [6]. Jiajia Zhao, etc., designed a detection approach based on sparse representation.

In this paper, we propose an end-to-end deep learning solution for small target detection based on deep learning which could be taken as a method belongs to classifier. Experimental results prove that our proposed method is robust and insensitive to background and target changing.

Nowadays, deep architectures with convolution and pooling are found to be highly effective and commonly used in computer vision and object recognition [7-19]. The most

*Corresponding author: Ming LIU, Beijing Key Lab. for Precision Optoelectronic Measurement Instrument and Technology, Beijing Institute of Technology, Beijing, China, E-mail: bit411liu@bit.edu.cn
Hao-yuan DU, Yue-jin ZHAO, Li-quan DONG, Mei HUI, Beijing Key Lab. for Precision Optoelectronic Measurement Instrument and Technology, Beijing Institute of Technology, Beijing, China
impressive result was achieved in 2015 ImageNet contest, where 1.2 million images in the training set with 1000 different object classes. On the test data set including 150,000 images, the deep Convolutional Neural Network (CNN) approach described in [18] achieved the top 5 error rates 3.57%, considerably lower than human recognition error rate of 5%. Furthermore, CNN has achieved superior classification accuracy on different tasks, such as handwritten digits or Latin and Chinese character recognition [8,9], traffic sign recognition [10], face detection and recognition [11], radar target recognition [19]. All facts make us believe that deep neural nets could potentially be used on many other applications, including image small target detection.

In recent years, there have been many papers about using deep neural nets on target detection [20-22]. However, their goal is to locate and recognize large objects on images. The size of the objects in the image are usually scores or hundreds of pixels, full of complex image details, which is easy for human to locate and recognize. As for small target, sized a few pixels, it can only be located, barely able to identify its classes, which is the focus of this paper. So long as we know, it is the first time that deep neural nets are used in this field.

### 2 Deep Nets Architectures and configuration

We designed the nets with an input dimension of 21×21 pixels. The small input nets are used as a moving filter window to detect small target at all positions of an image. During training, the only preprocessing we do is subtracting the mean value, computed on the training set, from each pixel. The image is passed through a stack of layers. Each layer is Full-Connected with following layer. The first layers have 128 channels each, and the last performs 2-way classification. The final layer is the soft-max transform layer. All hidden layers are equipped with the rectification non-linearity [23]. Table 1 lists the models we trained in this paper.

<table>
<thead>
<tr>
<th>Model Name</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input layers</td>
<td>441×128+</td>
<td>441×128+</td>
<td>441×128+</td>
<td>441×128+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RELU</td>
<td>RELU</td>
<td>RELU</td>
<td>RELU</td>
<td></td>
</tr>
<tr>
<td>Middle layers</td>
<td>1×</td>
<td>2×</td>
<td>3×</td>
<td>3×</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{128×128+</td>
<td>{128×128+</td>
<td>{128×128+</td>
<td>{128×128+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RELU}</td>
<td>RELU}</td>
<td>RELU}</td>
<td>RELU}</td>
<td></td>
</tr>
<tr>
<td>Output layers</td>
<td>441×2+</td>
<td>128×2+</td>
<td>128×2+</td>
<td>128×2+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Softmax</td>
<td>Softmax</td>
<td>Softmax</td>
<td>Softmax</td>
<td></td>
</tr>
<tr>
<td>Number of full</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>connection layers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Deep architectures for small target detection
3 Sample Set Generation

As for any neural networks to be set up, an important job is to find and establish a dataset with enough number of samples for nets training and validation. Unlike general daily life object image datasets which are publicly accessible, the datasets for small target detection have to be generated ourselves.

First, we randomly downloaded some cloud-sky images from the internet and transformed them to gray level images as a source for background generation. Figure 1a shows some of these background images. Second, background image patches are randomly cropped from background image source by a dimension of 21x21 pixels. Next, a simple program was written to add target spot at the center of half number of background patches as target images, the other half number of background images are kept unchanged as sample images of no target. The grey-scale (or grey/ intensity) distribution of spot target are generated by Eq.1~Eq.4. Figure 1b shows some generated image samples.

\[
\begin{align*}
    w_x &= \text{rand()} \times 2 + 1. \quad (1) \\
    w_y &= \text{rand()} \times 2 + 1. \quad (2) \\
    \alpha &= \text{rand()} \times \pi. \quad (3) \\
    s(x,y) &= \exp\{-(\frac{(\cos(\alpha)x - \sin(\alpha)y)}{w_x})^2 \\
               &+ (\frac{(\cos(\alpha)y + \sin(\alpha)x)}{w_y})^2\}. \quad (4)
\end{align*}
\]

Where \( w_x \) and \( w_y \) are the dimensions of gauss distribution on two perpendicular directions. \( \alpha \) is random angle between 0~\( \pi \), decides the direction of distribution. \( \text{rand()} \) generates uniform distributed random value between 0 and 1. \( s(x,y) \) is the grey level function and \( (x,y) \) represents the pixel position in a sample patch, where the center coordinate of patches is (0,0).

Figure 1. (a) Some cloud-sky images for background generation and (b) some generated samples (Random SNR between 0 and 1).
In order to train the nets effectively, the target spot intensity should be wisely selected. There are basically two kinds of strategy we used in sample sets establishing, a random intensity strategy and an intensity strategy with constant SNR. There are some considerations to be made.

First, unlike common object recognition, the sampling regions of target and no target overlap with each other under a low or unstable SNR, result in problem of over-fitting in training (Figure 2c). Besides, if the SNR is very high, there could be much space between two sampling regions, and the samples could even be separated linearly (Figure 2a). The nonlinear classification ability of neural nets would be in vain. Figure 2 illustrates the effect of SNR under a low dimensional condition.

![Image](image.png)

**Figure 2.** Samples Classification under different SNR: triangles and quadrates represent image samples with and without target in sample space respectively. The quadrate target samples are obtained by adding target vector to the corresponding triangular no target sample. The upper vector represents target vector of different amplitude for each situation. A longer target vector makes a higher SNR.

Second, in order to have a better training performance with limited number of samples, training sample points should be as close to class boundaries as possible while maintaining an ideal shape of boundary surface without mixing with other classes. A boundary approaching effect, which is illustrated by Figure 3, can be obtained by holding SNR as a specific constant in a training set.

![Image](image.png)

**Figure 3.** Boundary approaching effect: triangles and quadrates represent negative and positive samples respectively. Samples near boundary while maintaining good surface make training easier.
All in all, we choose a strategy with fixed SNR to generate target samples and compare the net training result with the strategy of random SNR sample generation. First, a random background patch is normalized to have unit variance as Eq.5.

\[
B_n(x,y) = \frac{B(x,y) - \text{mean}(B(x,y))}{\sqrt{\sum_{(x',y')} (B(x',y') - \text{mean}(B(x,y)))^2}}
\]  

Then, target spot and background patch are added together according to SNR to make a target sample image, as Eq.6.

\[
P(x,y) = \text{SNR} \times s(x,y) + B_n(x,y).
\]

Where \(B_n(x,y)\) is normalized background image, \(\text{mean}(B(x,y))\) is the mean value of the background patch. \(P(x,y)\) is the generated target sample image. By controlling SNR, we generated 140000 sample images each time for one time training and validation. Half of the samples are positive targets and the others are negative backgrounds with no targets.

4 Experiments

We trained deep nets of different layer width and depth on training sets of different SNRs. Then evaluate the trained nets on a large sky-cloud image (sized 1024×768) which does not belong to the background source set. 100 small spot targets are generated by Eq.1~4 and randomly added on the sky-cloud background image to generate an image for final performance test. The intensity of spot is controlled in order to get an evenly-distributed local SNR ranging from 0 to 1. The performance of nets is compared by Eq.7 and 8.

Small Area Signal-to-Noise ratio gain:

\[
\text{SSNR Gain} = \frac{(S / C_s)_{\text{out}}}{(S / C_s)_{\text{in}}}
\]

Large Area Signal-to-Noise ratio gain:

\[
\text{LSNR Gain} = \frac{(S / C_l)_{\text{out}}}{(S / C_l)_{\text{in}}}
\]

Where \(S\) is the signal amplitude, \(C_s\) and \(C_l\) are the standard deviations within a local small area (21×21pixel) and a large area (201×201pixel) respectively. SSNR reflects the ability of signal enhancement while LSNR reflects the ability of background noise suppression. Both indices with larger value imply better performance. Different from other papers, we do not use the indices of background suppression factor (BSF) due to its dependency on signal amplification level.

All training and tests were conducted on matlab platform with matconvnet [24] toolbox. A laptop computer with Intel Core i7-2630QM CPU and GTX650m Nvidia Graphic Card was used.
We trained all nets in Table 1 by training set of different SNRs, then test and compare their performances with traditional method of Max-mean and Max-median filtering. Table 2 lists all test results.

**Table 2. Performance of different methods and architectures**

<table>
<thead>
<tr>
<th>Models and Method</th>
<th>Train SNR</th>
<th>Mean SSNR Gain</th>
<th>Mean LSNR Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2.0781</td>
<td>14.3569</td>
</tr>
<tr>
<td></td>
<td>Random</td>
<td>2.0781</td>
<td>14.3569</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>3.0178</td>
<td>18.4606</td>
</tr>
<tr>
<td></td>
<td>Random</td>
<td>2.5196</td>
<td>16.3126</td>
</tr>
<tr>
<td>C</td>
<td>16</td>
<td>3.9680</td>
<td>21.5455</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4.0517</td>
<td>21.6317</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4.1437</td>
<td>21.8808</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4.2621</td>
<td>22.2605</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.3197</td>
<td>22.8432</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>3.9907</td>
<td>22.6232</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>3.6274</td>
<td>21.2948</td>
</tr>
<tr>
<td></td>
<td>0.125</td>
<td>2.9646</td>
<td>17.3168</td>
</tr>
<tr>
<td></td>
<td>Random</td>
<td>2.9282</td>
<td>16.1848</td>
</tr>
<tr>
<td>D</td>
<td>16</td>
<td>4.7604</td>
<td>24.5610</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>4.6380</td>
<td>24.1185</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4.7964</td>
<td>24.2731</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4.8474</td>
<td>24.4911</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>4.9587</td>
<td>25.0740</td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>4.4921</td>
<td>25.5707</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>3.7134</td>
<td>22.0043</td>
</tr>
<tr>
<td></td>
<td>0.125</td>
<td>3.1878</td>
<td>16.5554</td>
</tr>
<tr>
<td></td>
<td>Random</td>
<td>3.9263</td>
<td>17.5989</td>
</tr>
<tr>
<td>Max-mean</td>
<td></td>
<td>1.8351</td>
<td>7.0192</td>
</tr>
<tr>
<td>Max-median</td>
<td></td>
<td>2.0031</td>
<td>8.5107</td>
</tr>
</tbody>
</table>
As shown in Table 2, the performances of deep nets are significantly better than traditional filtering methods of Max-mean and Max-median. Deeper nets get better performances in case the number of full connection layers are less than or equal to 5. The training process did not converge for nets with more than 5 full connection layers. And the nets trained by samples with constant SNR (SNR=1) achieves best training performance, better than random sampling, except for the linear classification model (type A). The performance of linear classification model does not change with SNR.

Moreover, we give our explanations on why SNR≈1 is the best choice. Figure 4 provides a sampling space with only two dimensions, each axis of the Figure represents a pixel gray level of a sample image which has only two pixels here. Each image, whether it is a target of background, could be expressed as a point in this Figure. Then, the random background images with equal intensity would form a circle centered at the original point on the Figure. The circle radius equals to the intensity of background. On the other hand, if a target is added to these background, we would get a new circle formed by target images. As there are more than one type of targets (There are 3 types of targets on Figure (4), we would get a set of circles, each circle set in the point represent a type of target with given intensity. Then Figure 4 shows three conditions with SNR>1, SNR=1 and SNR<1 in (a) (b) and (c) respectively.

As shown in Figure 4a, as for background noise of totally random, a sampling SNR>2, a large gap exists between background and target samples. The samples could even be divided linearly which is obviously inaccurate. As Figure 4b shows, sampling with SNR = 2 gets a perfect classification between classes. There are many samples located near the classification surface while no mixture between them, which meet the conditions of boundary approaching effect. In Figure 4c, with a sampling SNR>2, samples of two classes mixed with each other and some background samples are misclassified as target samples. Usually, there are far more background pixels than target pixels in an image and this misclassification could lead to pool performance in test. In a word, SNR=2 is the best choice for training in small target sample generation.

This conclusion does not coincide with the result of experiments at a first glance. However, the key idea to understand this inconformity is to notice the different definitions of SNR. The intensity of signal used in sample generation experiments is the peak value of signal pixels, while in Figure 4, it is the root-mean-square of signal pixels. For a signal spot of gauss shape, the ratio between peak value and root-mean-square is about 2, which causes the SNR inconformity.
Figure 4. Training Result with Different Sampling mean square root SNR: (a) SNR > 2; (b) SNR = 2; (c) SNR < 2
5 Conclusions

In our work, a new small target detection method is proposed based on deep learning. The performance of deep nets is significantly better than traditional filtering method. It is found that, the nets trained by samples of specific constant SNR, gets better performance than that trained by samples of random SNR. A reasonable choice of SNR is SNR≈1 (peak signal value SNR) which gets the best test performance in all tests with different sampling SNRs. As it has been well known that the addition of noise to the input data of a neural network during training can lead to significant improvements in generalization performance [25], our conclusion is that adding too much noise would do more harm than good in training for the application to detect signal from noise background. And we provide a simple explanation on why sampling SNR≈1(peak signal value SNR) gets the best performance in all tests. And SNR = 2 (mean square root SNR) might get even better performance, which needs a further testify. This conclusion might also be useful for training of nets on general objects recognition and detection which needs a further research. The next work would be to reconstruct the nets by convolutions to achieve a parallel faster even real-time small target detection algorithm.

Acknowledgement: This research was financially supported by the National Science Foundation of China (No.61301190)

References