To be appeared at: The 16th Australian Joint Conference on Artificial Intelligence - AI'03, Dec. 2003, Perth, Australia

Convolutional neural networks for image processing: an application in robot vision

Matthew Browne, Saeed Shiry Ghidary

GMD-Japan Research Laboratory,
Collaboration Center,
2-1 Hibikino, Wakamatsu-ku,
Kitakyushu-city, JAPAN
Email: matthew.browne@gmd.gr.jp, saeed.ghidary@gmd.gr.jp

Phone: +81 93 695-3085
Fax: +81 93 695-3525

Keywords: neural networks, computer vision, applications

Convolutional neural networks (CNNs) represent an interesting method for adaptive image processing, and form a link between general feedforward neural networks and adaptive filters. Two dimensional CNNs are formed by one or more layers of two dimensional filters, with possible non-linear activation functions and/or down-sampling. Conventional neural network error minimization methods may be used to optimize convolutional networks in order to implement quite powerful image transformations. CNNs possess key properties of translation invariance and spatially local connections (receptive fields). CNNs are an interesting alternative when the the input is spatially or temporally distributed, and the desired output of a system may be specified. The present paper presents a description of the convolutional network architecture, and an application to a practical image processing application on a mobile robot. As a formal CNN framework has not yet been specified in the literature, we describe CNNs in some detail, conceptually and formally. A CNN is used to detect and characterize cracks on an autonomous sewer inspection robot. Although cracks are relatively easy to detect by a human operator, autonomous sewer inspection necessitates the detection of pipe damage using computer vision methods. This is an appropriate application for trainable data-based computer vision methods, since prior specification of appropriate of the filtering / detection method is quite difficult. The The CNN architecture used involved a total of five layers: a single input and output map, and three hidden layers. The filter sizes used in all cases were 5x5, and the common activation function used was a log-sigmoid. The number

of feature maps used in the three hidden layers was, from input to output, 4, 3, 2. Thus, the number of neural weights to be optimized was 624 while the input to the network was a square region with side lengths of 68 pixels, yielding a total of 4624 pixel inputs to the network. The network was trained using a dataset of 39 still image 320x240 pixel frames sampled from a pre-recorded sewer pipe inspection video. Although development of a CNN system for civil use is ongoing, the results support the notion that data-based adaptive image processing methods such as CNNs are useful for image processing, or other applications where the input arrays are large, and spatially / temporally distributed. Further refinements of the CNN architecture, such as the implementation of separable filters, or extensions to three dimensional (ie video) processing, are suggested.

Convolutional neural networks for image processing: an application in robot vision

No Author Given

GMD-Japan Research Laboratory

1 Abstract

Convolutional neural networks (CNNs) represent an interesting method for adaptive image processing, and form a link between general feedforward neural networks and adaptive filters. Two dimensional CNNs are formed by one or more layers of two dimensional filters, with possible non-linear activation functions and/or down-sampling. CNNs possess key properties of translation invariance and spatially local connections (receptive fields). The present paper presents a description of the convolutional network architecture, and an application to a practical image processing application on a mobile robot. A CNN is used to detect and characterize cracks on an autonomous sewer inspection robot. The filter sizes used in all layers of the CNN used here was 5x5, and the common activation function used was a log-sigmoid. The network was trained using a dataset of 39 still image 320x240 pixel frames sampled from a pre-recorded sewer pipe inspection video. The results support the notion that data-based adaptive image processing methods such as CNNs are useful for image processing, or other applications where the input arrays are large, and spatially / temporally distributed. Further refinements of the CNN architecture, such as the implementation of separable filters, or extensions to three dimensional (ie video) processing, are suggested.

2 Introduction

The term *convolutional network* (CNN) is used to describe an architecture for applying neural networks to two-dimensional arrays (usually images), based on spatially localized neural input. This architecture has also been described as the technique of shared weights or local receptive fields [1–3] and is the main feature of Fukushima's *neocognitron* [4, 5]. Le Cun and Bengio [6] note three architectural ideas common to CNNs: local receptive fields, shared weights (weight averaging), and often, spatial down-sampling. Processing units with identical weight vectors and local receptive fields are arranged in an spatial array, creating an architecture with parallels to models of biological vision systems [6]. A CNN image mapping is characterized by the strong constraint of requiring that each neural connection implements the same local transformation at all spatial translations. This dramatically improves the ratio between the number of degrees of freedom in the system and number of cases, increasing the chances of generalization [7]. This advantage is significant in the field of image processing, since without the use of appropriate constraints, the high dimensionality of the input data generally leads to ill-posed problems. To some extent, CNNs reflect models of biological vision systems [8]. CNNs take raw data, without the need for an initial separate pre-processing or feature extraction stage: in a CNN the feature extraction and classification stages occur naturally within a single framework.

In the CNN architecture, the 'sharing' of weights over processing units reduces the number of free variables, increasing the generalization performance of the network. Weights are replicated over the spatial array, leading to intrinsic insensitivity to translations of the input - an attractive feature for image classification applications. CNNs have been shown to be ideally suited for implementation in hardware, enabling very fast real-time implementation [9]. Although CNN have not been widely applied in image processing, they have been applied to handwritten character recognition [2, 9–11] and face recognition [7, 8, 12]. CNNs may be conceptualized as a system of connected feature detectors with non-linear activations. The first layer of a CNN generally implements non-linear template-matching at a relatively fine spatial resolution, extracting basic features of the data. Subsequent layers learn to recognize particular spatial combinations of previous features, generating 'patterns of patterns' in a hierarchical manner. If downsampling is implemented, then subsequent layers perform pattern recognition at progressively larger spatial scales, with lower resolution. A CNN with several downsampling layers enables processing of large spatial arrays, with relatively few free weights.

The present paper presents an application of CNN to an applied problem in mobile robotics. In particular, the visual system of a KURT2 [13] autonomous mobile robot designed for sewer inspection and equipped with an infra-red video camera. The task of the the robot is the online detection of cracks and other faults in sewer pipes. The cracks are defined by a relatively distinctive space-frequency structure. However, they are often embedded in a variety of complex textures and other spurious features. Lighting and reflection effects present an additional source of difficulty. The implementation and performance of the CNN architecture is discussed in terms of this application. The CNN is also applied to certain artificial image processing problems.

Figure 1 shows the architecture of a CNN with two layers of convolution weights and one output processing layer. Neural weights in the convolution layers are arranged in an 2-D filter matrices, and convolved with the preceding array. In figure 1, a single layer 1 neural filter is shown operating at two different spatial translations on the input array. The output (shaded pixel) forms a spatially distributed feature map, which is processed by the second convolutional layer, and passed to the output array. Downsampling of the feature array may be implemented between the convolution layers. For fixed filter sizes, this has the effect of increasing the spatial range of subsequent layers, while reducing the level of spatial resolution. As with most neural networks, the exact architecture of a CNN should depend on the problem at hand. This involves determination

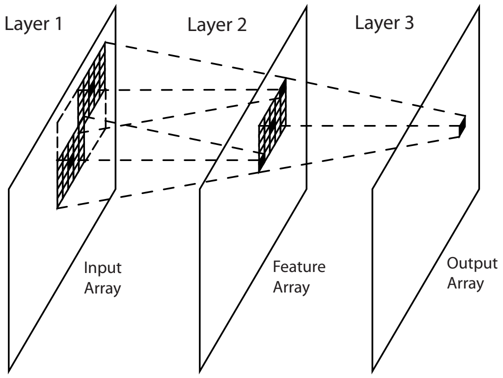

Fig. 1. Architecture of a CNN with a single convolutional neuron in two layers. A 5x5 filter at two different translations is shown mapping to shaded pixels in the first feature array. Shaded pixels in turn are part of the local feature information which is mapped to the output (or second feature) array by a second 5x5 filter.

of: the mixture of convolutional and downsampling layers (if any), filter-weight sizes, number of neurons, and interconnectivity between layers.

Figure 2 shows an example two layer CNN with multiple neurons in each layer,with down-sampling being implemented by the second layer.

Figure 3 displays the application of a simple CNN to a toy problem of detecting road markers from a camera image mounted above an intersection. It may be seen that the feature arrays in layer 1 capture simple features, while the feature arrays in the second hidden layer capture have more complex responses. The final output of the system does a good job of detecting road markers despite low degrees of freedom, a high degree of noise, and the presence of many distractors such as cars or road signs.

3 Convolutional Neural Networks

CNNs perform mappings between spatially / temporally distributed arrays in arbitrary dimensions. They appear to be suitable for application to time series, images, or video. CNNs are characterized by:

- translation invariance (neural weights are fixed with respect to spatial translation)
- local connectivity (neural connections only exist between spatially local regions)
- an optional progressive decrease in spatial resolution (as the number of features is gradually increased).

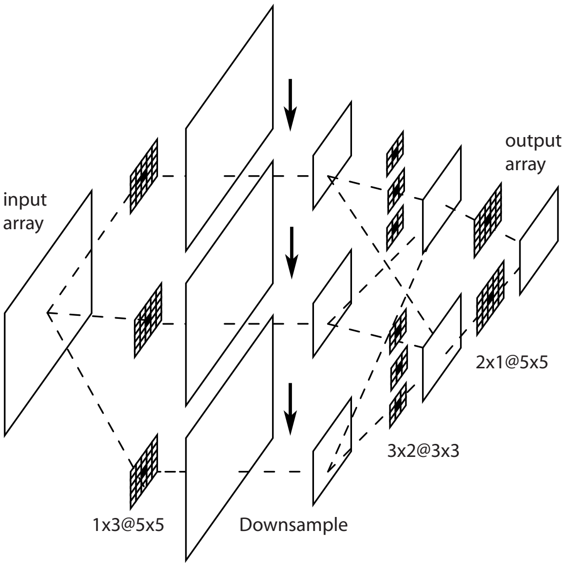

Fig. 2. Architecture of a fully interconnected CNN with multiple neurons in the convolution layer, and a factor of two translation in the second layer.

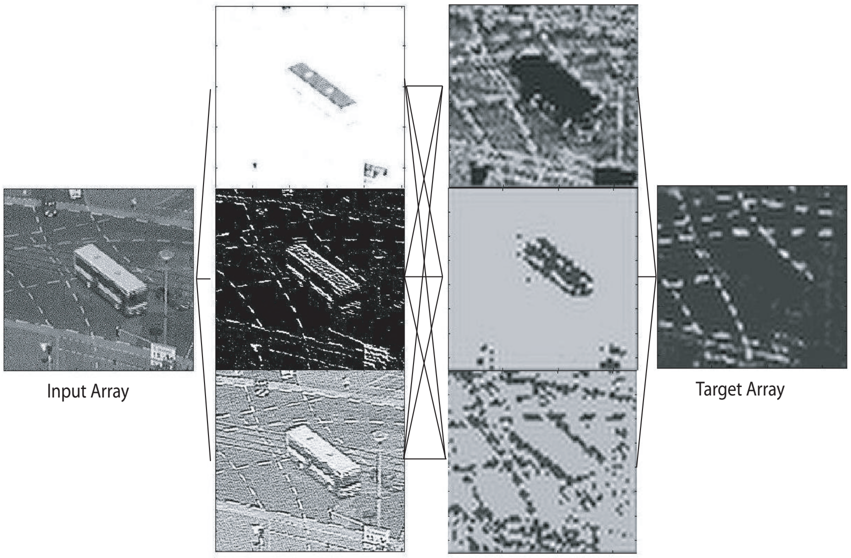

Fig. 3. Input, feature, and output arrays of a convolution network applied to detecting road markers.

These constraints make a CNN operate like a system of interconnected filters, and profitable comparisons may be made between other filtering systems, since the neural weights of a CNN operate like the taps of a system of finite impulse response (FIR) or wavelet filters. Thus a trained CNN may be thought of trainable filter system, custom made for a certain function mapping application. Finally, CNNs allow the processing of large spatially distributed arrays without a correspondingly large number of free parameters, increasing the chances of minima avoidance and generalization.

Initially, we will describe the case of one dimensional input and a single hidden layer. Extensions may be then be made to multiple dimensions and multiple layers, with possible operation of downsampling. We wish to obtain the formula for changing the weight and bias parameters of a CNN given a training set of input-output pairs $\xi^{\mu}_{k,r}, \zeta^{\mu}_{i,p}$. The indexes $\{i, j, k\}$ refer respectively to neuron arrays in the output, hidden, and input layers. In a CNN neurons are 'replicated' with respect to spatial translation, although they share the same weight and bias vectors. Indices $\{p, q, r\}$ are to used to as a spatial index for each layer. A property of CNNs is that translated neurons $h^{\mu}_{j,q}$ receive only *local* connections from the previous layer - a CNN is not a fully interconnected network. So, when calculating the net input to a particular translated neuron, it is convenient to index the spatially distributed weights separately, using indices s, t, u. The weights of a CNN are invariant to spatial translation, so it is natural to think of the set of weights $w^{t}_{j,k}, t = \{-T, .., 0, ..., T\}$ as a filter that connects input array $\xi^{\mu}_{k,r}$ to feature array $V^{\mu}_{j,q}$. $2T + 1$ is size of the region surrounding each translation point for which network weights exist, i.e., the filter size. It is constrained to be of an odd - numbered length, so it is symmetrical about q.

A summary of the indices used in the present paper is shown in table 1.

	output	hidden	input
array label	O	V	ξ
array index	i	j	k
spatial index	p	q	r
weight index	s	t	

Table 1. Indexes and array terms, organized with respect to layer and data type.

Given input pattern μ hidden unit j, q receives net input

$$h^{\mu}_{j,q} = \sum_{k} \sum_{t} w^{t}_{j,k} \xi^{\mu}_{k,q+t} + b_j \tag{1}$$

where the index to ξ^{μ}_{k} is clamped to spatially local positions, centered at translation q, by setting $r = q + t$. The term b_j refers to the usual constant bias. The neural output forms the hidden feature arrays, produced by the transfer function

$$V^{\mu}_{j,q} = g\big(h^{\mu}_{j,q}\big). \tag{2}$$

The neuron at translation p in the i^{th} array in the output layer receives net input

$$h^{\mu}_{i,p} = \sum_{j}\sum_{s} w^{s}_{i,j} V^{\mu}_{j,p+s} + b_i \tag{3}$$

where, as before, $s = \{-S, .., 0, ..., S\}$, and $2S + 1$ describes the length of the filter in the output layer, and relative indexing has been substituted for absolute indexing; $q = p + s$. Final output of the network

$$O^{\mu}_{i,p} = g\big(h^{\mu}_{i,p}\big) = g\bigg(\sum_{j}\sum_{s} w^{s}_{i,j} V^{\mu}_{j,p+s} + b_i\bigg) \tag{4}$$

Identical to the effect of applying two convolution filters sequentially, CNNs with hidden layers result in progressively larger portions of the input contributing to the function, $O_p = f(\xi_{p-S-T}, ..., \xi_p, ..., \xi_{p+S+T})$.

4 Delta rule for CNNs

Although CNNs make use of the same weight update rule as normal neural networks, some care should be taken in implementation, particularly with differential indexing of spatial translation and feature maps.

As with normal ANNs, the cost function of a CNN is

$$E = \frac{1}{2}\sum_{\mu,i,p}\big[\zeta^{\mu}_{i,p} - O^{\mu}_{i,p}\big]^2. \tag{5}$$

We find the derivative of the error with respect to the s^{th} weight of the filter connecting the j^{th} feature array to the i^{th} output array

$$\frac{\partial E}{\partial w^{s}_{i,j}} = -\sum_{\mu,p}\big[\zeta^{\mu}_{i,p} - g\big(h^{\mu}_{i,p}\big)\big] g'\big(h^{\mu}_{i,p}\big) V^{\mu}_{j,p+s} \tag{6}$$

which used in combination with the familiar gradient descent weight update rule yields

$$\varDelta w^{s}_{i,j} = \eta \sum_{\mu,p} \delta^{\mu}_{i,p} V^{\mu}_{j,p+s} \tag{7}$$

where

$$\delta^{\mu}_{i,p} = \big[\zeta^{\mu}_{i,p} - g\big(h^{\mu}_{i,p}\big)\big] g'\big(h^{\mu}_{i,p}\big). \tag{8}$$

In order to find the weight change of the input to hidden connections $w^s_{j,k}$ the delta rule is applied with a change of indices

$$\mathit{\Delta} w^s_{j,k} = \eta \sum_{\mu,q} \delta^\mu_{j,q} \xi^\mu_{k,q+t} \tag{9}$$

where

$$\delta^\mu_{j,q} = \sum_s g'\big(h^\mu_{j,q}\big) \sum_i \delta^\mu_{i,q-s} w^s_{i,j}. \tag{10}$$

Bias terms b_i and b_j are treated as normal weights with constant input, and may be likewise updated using (10) and (11).

5 Subsampling

Often when applying CNNs we wish to progressively reduce spatial resolution at each layer in the network. For example, a CNN may be used for classification where an image is mapped to a single classification output. Given fixed filter sizes, reducing spatial resolution has the effect of increasing the effective spatial range of subsequent filters. In a CNN with subsampling in each layer, the outcome is a gradual increase in the number of features used to describe the data, combined with a gradual decrease in spatial resolution. Because the change in coordinate system is accomplished in a nonlinear, incremental, hierarchical manner, the transformation can be be made insensitive to input translation, while incorporating information regarding the relative spatial location of features. This provides an interesting contrast to methods such as principle components analysis, which make the transition from normal coordinate space to feature space in a single linear transformation.

We can rewrite the previous formulas for calculating the output of the network, given that both layers incorporate spatial subsampling. This has been previously accomplished using a seperate 'averaging' layer with fixed neural weights [2]. However, it is described below by increasing the shift indexes by a factor of two, thus combining adaptive and downsampling functions. Since the averaging layer in the method of Le Cun [2] may be specified by a 'double shift' layer with a filter size of 2, it may be shown that the present formalism is essentially similar, albiet more general, and allowing for adaption of previously fixed averaging weights.

$$h^\mu_{j,q} = \sum_k \sum_t w^t_{j,k} \xi^\mu_{k,2q+t} + b_j \tag{11}$$

$$h^\mu_{i,p} = \sum_j \sum_s w^s_{i,j} V^\mu_{j,2p+s} + b_i \tag{12}$$

The output of the network is then

$$O^\mu_{i,p} = g\big(\sum_j \sum_s w^s_{i,j} g\big(h^\mu_{j,2p+s}\big) + b_i\big) \tag{13}$$

$$= g\big(\sum_j \sum_s w_{i,j}^s g\big(\sum_k \sum_t w_{j,k}^t \xi_{k,4p+2s+t}^\mu + b_j\big) + b_i\big). \tag{14}$$

For a general CNN with N layers, being some combination of non-subsampling and subsampling layers, and filter sizes being given by F_n, the local region of input contributing to the output is given by the recursive formulas

$$R_{n+1} = R_n + F_n - 1 \text{ if nonsubsampling} \tag{15}$$

$$R_{n+1} = 2(R_n + F_n) - 3 \text{ if subsampling} \tag{16}$$

given $R_1 = F_1$. Given fixed filter sizes F_n, it is clear that the input 'window' of CNN may grow rapidly as the number of subsampling layers increase. Most wavelet transforms utilize a similar nested series of downsampling operations, achieving significant computational savings. In fact, given a tree-like connectivity between feature arrays, the sub-sampled CNN may be profitably conceptualized as a wavelet transform with adaptable filters.

For downsampling layers, the weight update rules are identical to [7] and [9], with the shift indices p and q increased by a multiple of two.

6 Method

CNNs are investigated in the current work for use as an image processing system on an autonomous mobile robot (see [14, 15] for details). The task of the system is autonomous detection and characterization of cracks and damage in sewer pipe walls. The robot scans the pipe wall using a monochrome CCD camera, which is digitally converted at a resolution of 320x240 pixels per frame. The task of the CNN is to perform filtering of the raw pixel data, identifying the spatial location of cracks, enabling subsequent characterization of the length, width, etc of the damage. Figure 4 displays a sample input frame, along with the ideal output of the CNN. Although the cracks are easily identifiable by eye, the image processing task is quite complex, as variability in lighting and orientation, width and type of crack, along with the presence of other crack-like structures (such as joins between pipe sections), combine to make a challenging computer vision task.

A representative data-set of 37 frames from an on-line recording session were manually classified for training and validation of the network. A training data set was generated using 20 of the images, sampling 100 crack and 200 non-crack pixel locations, yielding a total of 6000 input-target pairs. Although all pixel locations had associated targets, not every pixel was used for training because: a) computational expense b) the low proportion of 'crack' to 'clean' training samples tended to bias the network towards classifying all samples as 'clean'. Alternatively, it would be possible to use a biased learning procedure, where the error generated by the rarer class would be weighted in inverse proportion to the ratio of occurance.

The CNN architecture used involved a total of five layers: a single input and output map, and three hidden layers. The filter sizes used in all cases were 5x5, and the common activation function used was a log-sigmoid. The number of feature maps used in the three hidden layers was, from input to output, 4, 3, 2. Thus, the number of neural weights to be optimized was $(5^2 + 1)(1 * 4 + 4 * 3 + 3 * 2 + 2 * 1) = 624$ while the input to the network was a square region with side lengths of $(((5 * 2 + 4) + 4) + 4) = 68$ pixels, yielding a total of 4624 pixel inputs to the network. These figures are indicative of the degrees-of-freedom saving achieved by the weight sharing method. Training was conducted using standard weight updated rules, as described about, for 10,000 epochs, using a learning rate $\eta = 0.05$, requiring approximately two hours of compute time. The network was implemented in C++.

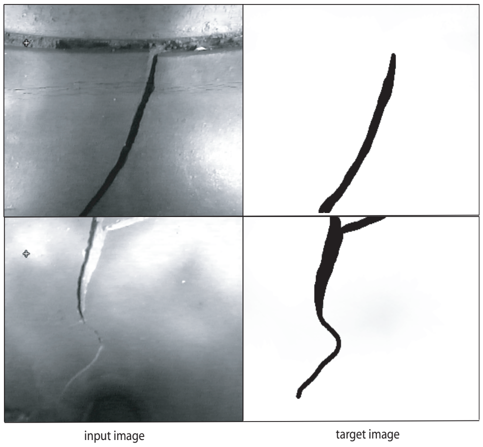

Fig. 4. Example input and target images for large cracks on a concrete pipe. Note the horizontal feature in the upper left image is a pipe joint, not a crack. Differentiating between pipes and joints, accounting for shadows and lighting effects are significant challenges for a detection / filtering system.

7 Results

Approximately 93% of pixels in the validation set were correctly classified. However, as the current application relates to an image processing / filtering task, numerical results (i.e. percentage of pixels correctly / incorrectly classified) are less informative than graphical results. Figure 5 displays three example frames, that were representative of the data set, including crack present, no crack and pipe joint, and crack and joint together. The network appears to have successfully ignored the presence of joints, and attenuated lighting effects while enhancing the cracks. In the context of the application to sewer pipe defect detection and characterization, the output may be profitably used by a subsequent crack detection algorithm. However, the present system is in it's early stages, and we believe that further refinement may provide better results.

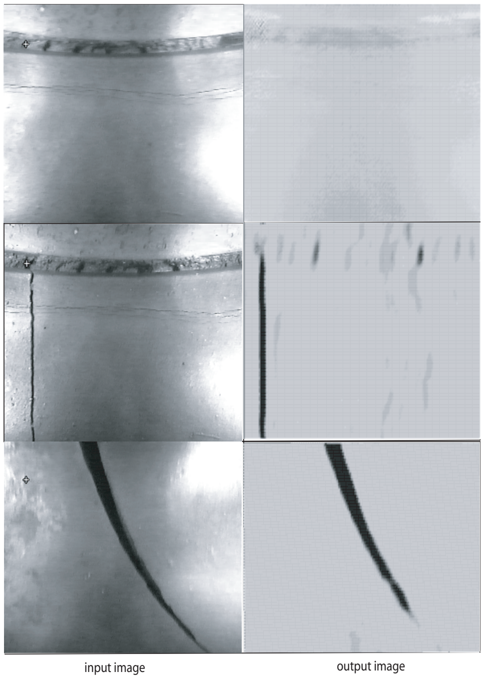

Fig. 5. Example input and CNN output frames Note that because of the 68 pixel window required by the network, the output image represents is subregion of the input.

8 Discussion

We have described and applied a general CNN architecture to a 'real-world' problem of some interest to the civil robotics community. CNNs may be expected to achieve significantly better results than standard feed-forward networks for many tasks because they impose appropriate constraints on the way the function mapping is learnt. The key characteristics of local connectivity, and translation invariant weight sharing, are appropriate when the input data is spatially or temporally distributed. In addition, implementing down-sampling allows the network to progressively trade-off resolution for a greater input range.

In training the present CNN, some issues became apparent. Training using bitmap type images results in an over abundance of training sample points. It appears that, in order to maximize the available computational resources, that not all pixel-centre points of the training set should be used. Rather, a representative sub-sample would be more appropriate. In the present application, this might mean over-sampling joints as opposed to flat-wall regions, or thin cracks as opposed to thick cracks. This may be done manually by hand-coding multiple targets, and sampling each one equally. Alternatively, some statistical criteria might be developed for selection of a representative data subset. Also, we have not yet done work on deciding the best CNN architecture for a given application - the present architecture was simply chosen as appearing reasonable for the current application.

As yet, the utility of CNNs does not appear to have been fully realized, and applications to a wide variety of data-types and function mapping problems (ie physiological recordings, financial time-series analysis, automatic sateliite image processing) remain to be explored. In particular, through the implementation of 3-D filters, CNNs may represent a computationally feasible method of adaptive video processing. Refinements in the CNN architecture remain to be explored. For example, sequential CNN layers comprising 1xN and Nx1 filters may be used to learn separable NxN filter functions. There are clear links between CNNs and finite impulse response filters, adaptive filters, and wavelet transforms, and theoretical work bridging these disciplines would be of significant interest. As a final point, the development of methods for adapting the CNN architecture via discriminant or entropy-based cost functions, similar to those developed for wavelet packet analysis, would be of significant interest.

References

1. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representation by error propagation,” in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., vol. 1, pp. 318–362. Cambridge, MA: MIT Press, 1986.
2. Y.B. Le Cun, J.S. Boser, D. Denker, R.E. Henderson, W. Howard, W. Hubbard, and L.D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, vol. 4, no. 1, pp. 541–551, 1988.
3. K.J. Lang and G.E. Hinton, “Dimensionality reduction and prior knowledge in e-set recognition,” in *Advances in Neural Information Processing Systems*, D.S. Touretzky, Ed., pp. 178–185. Morgan Kauffman, San Marteo, CA, 1990.
4. K. Fukushima, S. Miyake, and T. Ito, “Neocognitron: a neural model for a mechanism of visual pattern recognition,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 826–834, 1983.

5. Kunihiko Fukushima, “Neocognitron: A hierachical neural network capable of visual pattern recognition,” *Neural Networks*, vol. 1, no. 2, pp. 119–130, 1988.
6. Y. Le Cun and Y. Bengio, “Convolutional networks for images, speech, and time series,” in *The Handbook of Brain Theory and Neural Networks*, M.A. Arbib, Ed., pp. 255–258. MIT Press, Cambridge, MA, 1995.
7. Steve Lawrence, C. Lee Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural network approach,” *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.
8. B. Fasel, “Robust face analysis using convolutional neural networks,” in *Proceedings of the International Conference on Pattern Recognition (ICPR 2002), Quebec, Canada*, 2002.
9. E. Sackinger, B. Boser, J. Bromley, and Y. LeCun, “Application of the anna neural network chip to high-speed character recognition,” *IEEE Transactions on Neural Networks*, vol. 3, pp. 498–505, 1992.
10. Y. Le Cun, “Generalization and network design strategies,” Tech. Rep. CRG-TR-89-4, Department of Computer Science, University of Toronto, 1989.
11. Y. Bengio, Y. Le Cun, and D. Henderson, “Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and Hidden Markov Models,” in *Advances in Neural Information Processing Systems*, Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, Eds. 1994, vol. 6, pp. 937–944, Morgan Kaufmann Publishers, Inc.
12. Beat Fasel, “Facial expression analysis using shape and motion information extracted by convolutional neural networks,” in *Proceedings of the International IEEE Workshop on Neural Networks for Signal Processing (NNSP 2002), Martigny, Switzerland*, 2002.
13. F. Kirchner and J. Hertzberg, “A prototype study of an autonomous robot platform for sewerage system maintenance,” *Autonomous Robots*, vol. 4, no. 4, pp. 319–331, 1997.
14. M.Browne, M. Dorn, R. Ouellette, and S. Shiry, “Wavelet entropy-based feature extraction for crack detection in sewer pipes.,” in *6th International Conference on Mechatronics Technology, Kitakyushu, Japan*, 2002.
15. M. Browne, S. Shiry, M. Dorn, and R. Ouellette, “Visual feature extraction via pca-based parameterization of wavelet density functions,” in *International Symposium on Robots and Automation, Toluca, Mexico*, 2002.