World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:12, 2007

# Enhancing Multi-Frame Images Using Self-Delaying Dynamic Networks

Lewis E. Hibell, Honghai Liu and David J. Brown

***Abstract*— This paper presents the use of a newly created network structure known as a Self-Delaying Dynamic Network (SDN) to create a high resolution image from a set of time stepped input frames. These SDNs are non-recurrent temporal neural networks which can process time sampled data. SDNs can store input data for a lifecycle and feature dynamic logic based connections between layers. Several low resolution images and one high resolution image of a scene were presented to the SDN during training by a Genetic Algorithm. The SDN was trained to process the input frames in order to recreate the high resolution image. The trained SDN was then used to enhance a number of unseen noisy image sets. The quality of high resolution images produced by the SDN is compared to that of high resolution images generated using Bi-Cubic interpolation. The SDN produced images are superior in several ways to the images produced using Bi-Cubic interpolation.**

***Keywords*— Image Enhancement, Neural Networks, Multi-Frame.**



## I. Introduction

IMAGE enhancement using a single view of an object is severely limited by the amount of information contained in the starting image. Many techniques have been proposed for interpolation in digital images [1]–[4] which treat the pixels as values on a 2D grid. [5]–[7] deal with modifying convolution based techniques to find more optimal interpolation kernels and methods. As do Anton et al. in [8] who also use irregular sampling. However due to the fixed nature of these convolution techniques sharp edges tend to be blurred. Li and Orchard in [9] and Hong et al. in [10] focus on improving the edge information usually blurred by convolution methods. These methods all still suffer from a lack of information which can only be overcome by increasing the amount available. This is why work has been done over the past few decades in the field of data fusion which attempts to integrate other information during enhancement. Many data fusion techniques to combine information from multiple classes of image capture device have been developed to utilise the best features of each device [11], [12]. The sources can range from visible and infrared [13] to differing satellite imagery [14]–[19]. The methods used for fusion are also varied including probabilistic deconvolution [20], conditional probability networks [21] and neural networks [13]. The most common technique is wavelet based [14]–[16], [22]. In this paper the interest lies in information contained within multiple frames originating from the same capture device. Although the above mentioned image fusion techniques could be used on images from the same device, techniques such as the one pioneered by Tsai and Huang in [23] were created with multiple images from the same device in mind. Other methods for multi-frame enhancement expanding on the work by Tsai and Huang include [24]–[26] who use Bayesian, back projection and preconditioned conjugate approaches respectively. The use of Recurrent Neural Networks (RNNs) to perform multi-frame enhancement was investigated by Salari and Zhang in [27]. The choice of using RNNs allowed Salari and Zhang to pass each frame through sequentially. This sequential use of the input frames requires a network which can store and use previous inputs. The processing of any temporal pattern is best performed by a network with such capabilities. This need to process temporal patterns with neural networks has produced a number of network structures [28]. Many of which are based on observations of how time is represented in the brain [29], [30]. Elman [31] expanded on the network proposed by Jordan and produced a network with temporal memory which showed that it was possible for neural networks to learn temporal relationships. Others such as Hochreiter and Schmidhuber in [32] attempt to add functionality to nodes within a network to handle the temporal information. However this can lead to networks with an unnecessarily high number of neurons. In [33] Wang et al. state that RNNs do not function well when presented with problems requiring rate invariance. Within RNNs relationships between inputs which are separated by many time steps cannot be represented because the error signals involved in the back propagation process either blow up or vanish [32]. This problem with the feedback error was the main reason for creating the SDNs which allow multi-frame temporal data to be processed without the requirement of feedback.

L. Hibell, H. Liu and D. J. Brown are with the Institute of Industrial Research, University of Portsmouth, Hampshire, PO1 3HE, United Kingdom, (Phone: +442392842547; E-mail lewis.hibell:honghai.liu:david.j.brown@port.ac.uk)

## II. Self-Delaying Dynamic Networks

SDNs are forward flowing network structures with the ability to store inputs at time $t$ and use them at time $t + n$ (where $n$ is a number of time periods later). The value of $n$ is learned during training and can vary if necessary for each input. It is used here to allow pixels from different temporal frames to be intelligently combined to form a high resolution image. Their structure is similar to that of a feed-forward network with an input layer, an output layer and several hidden layers. However, the algorithm used to propagate the network inputs towards the network outputs is unique because it is almost completely time dependant. The values produced at the output of the network are formed using the timing of the inputs

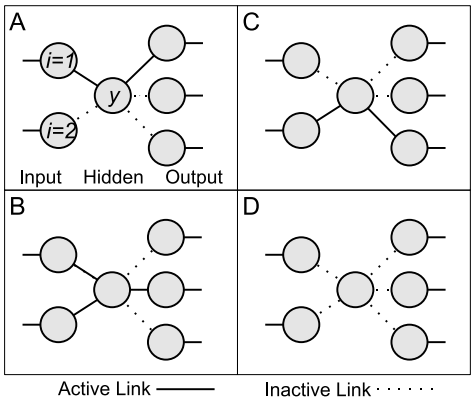


Fig. 1. During processing the network structure changes dependant on the iteration time. A: Initially network generates output 1 based on input 1. B: Network then changes links to generate output 2 based on both inputs. C: Later network generates output 3 based on input 2. D: Finally, all outputs are ready and no other links are necessary.



combined with the values themselves. They do not feature any recurrent elements but are still able to process temporal inputs due to their ability to store information. The network's dependence on its temporal structure and lack of feedback elements make it unsuitable for training with gradient descent methods. A genetic algorithm was therefore created to allow the network to evolve to a point at which it is able to complete the task of enhancing the input images.

### A. Temporal Processing

Before being presented to the network the input images are formed into small blocks of pixels. This allows the number of inputs being presented to the network to be consistent and also allows the network to be a reasonable size. The processing of an SDN happens in what has been called a processing lifecycle. This lifecycle consists of a series of iterations of an algorithm shown in Eq. (1). During this lifecycle each input frame is presented to the network for an equal number of iterations. In a two frame system the first frame is presented for the first half of the total iterations. The second frame is then presented for the second half of the iterations after the values of the first frame have been absorbed by the network. There are three types of weights in an SDN, two of which are time dependant. During each iteration the outputs from the final layer are calculated first, followed by each layer in turn back towards the input layer. This reverse approach has been adopted to allow the data stored within the nodes to be moved onwards before being replaced. Only on completion of a lifecycle have the outputs from the network been fixed. Fig. 1 shows a simplified example of a network lifecycle. The dynamic connection of the layers during a lifecycle is activated by logic with the inversons indicator functions in Eq. (1). Initially at time $A$ only two links are active. These two links allow one of the output values to be generated based on one of the inputs. At time $B$ The first two links are now deactivated and two different links become active, allowing a different output to be generated from both inputs. At time $C$ the active links change again to form a third output value. Finally at time $D$ no active links are required because all output values are now fixed.

### B. Calculation of node output

The output of node $y$ takes one of three main forms dependant on iteration $t$. The Node weight $N$ and the Link weights to the nodes in the previous layer $L_i(i = 1..n)$ are used to ascertain which of these three forms to use, Eq. (1). The node output either stays the same, is reset to zero or is recalculated based on its available inputs.

$$\begin{aligned} y_t = & \left[\Sigma_{i=1}^{n}\left[\left|\frac{\Sigma_{j=1}^{n}X_j - X_i}{n-1} - X_i\right| < T_i\right] \neq 0\right] \\ & \times \Sigma_{i=1}^{n}\frac{X_i\left[\left|\frac{\Sigma_{j=1}^{n}X_j - X_i}{n-1} - X_i\right| < T_i\right]}{\Sigma_{i=1}^{n}\left[\left|\frac{\Sigma_{j=1}^{n}X_j - X_i}{n-1} - X_i\right| < T_i\right]} \\ & \times \left[L_i < t < 4L_i\right]\left[N \leq t \leq 2N\right] \\ & + \left(1 - \left[N \leq t \leq 2N\right]\right) \times y_{t-1} \end{aligned} \quad (1)$$

Where:

| | |
|---|---|
| $t$ | is iteration time |
| $n$ | is number of nodes in the previous layer |
| $y$ | is the node being processed |
| $i$ | is one of $n$ nodes in the previous layer |
| $Y_t$ | is output of node $y$ at time $t$ |
| $X_i$,$X_j$ | are the values from nodes $i$ and $j$ |
| $T_i$ | is the Threshold from $i$ (Learned value) |
| $L_i$ | is Link weight from node $i$ (Learned value) |
| $N$ | is Node weight of node $y$ (Learned value) |
| $[\,]$ | is Inverson's indicator (1 if true, 0 if false) |

Each frame is fed into the first layer of the network for several iterations, the next frame is then fed to the first layer for an equal number of iterations. This is repeated until all frames have been fed into the network. The calculation of the node output $Y_t$ is dependent on the value of the three types of weights $T_i$,$L_i$ and $N$.

*1) Node Weight $N$:* The node weight $N$ is used to ascertain when recalculation of a node output should be performed. If the current iteration $t$ is between the node weight and double node weight then a recalculation is done otherwise the output remains unchanged.

*2) Link Weight $L_i$:* The link weight ($L_i$) is used to determine if a value coming from node $i$ to node $y$ is ready to be used in the calculation of the node output. If the iteration $t$ is between the link weight and four times the link weight then the value coming from $i$ is included in the calculation otherwise it is ignored.

*3) Threshold Weight $T_i$:* The final learned value is the threshold $T_i$, this is used to judge if the value coming from node $i$ is within a certain range of the other values being used in the node.

The calculations for the nodes in the input layer are more straightforward since the only source of data for these are the input value to the network. They are allowed to go to zero

after a number of network iterations in order to stop them having any more effect on the network. Examining Eq. (1) it can be shown that if the conditionals are assumed to be true then the algorithm reduces to an averaging of the input pixels similar to bilinear interpolation.

### C. Training

Due to the nature of SDNs and their unique lifecycle structure a genetic algorithm was created for training. This evolutionary approach also allows the combination and further training of networks trained on different images by creating a population containing both solutions. This mixed population can then create new networks featuring the best elements of each solution. Crossover of the network weights is performed taking into account the organisation of the weights. Short subbranches of the network are moved together making it possible for successful sections to remain complete.

### D. Node outputs

For any number $N$ of pixels passed into a node the possible number of output values is defined by Eq. (2).

$$N + 1 + \Sigma_{r=1}^{N} \frac{N!}{(N-r)!r!} \tag{2}$$

It assumes all inputs are different and that any combination of pixel values will result in a different output value. One possible output using a 3x3 block segmented implementation of the network is to perform the equivalent to bi-linear interpolation. Whereas another possible output of the same network with different weightings it to output nearest neighbour interpolation. A third possibility is for the system to output all the pixels which are correct in the low level image in their correct location and then fill in the "unknown" values with interpolations of the correct values. The normal functioning of the network would be to perform whichever process is more suitable to the current section of the image.

## III. Comparisons

Two main numerical methods are proposed to ascertain the relative efficiency of SDNs compared to other algorithms. These results rely on the existence of a target image and therefore could not be used to rate the system on a set of frames for which a target does not exist. Sets of frames for which targets do not exist can only be assessed by observing their usefulness to the user of the system. These methods are: Average Error Per Pixel (AEPP) and Number of Perfect Pixels (NPP). Each method identifies a different property of the network which is deemed useful when attempting to generate a high resolution result. In Eq. (3) and Eq. (4) $R$ is the reconstructed image and $T$ is the target image. Also $L$,$M$ are the width and height of the image, and $N$ is the number of colour bands.

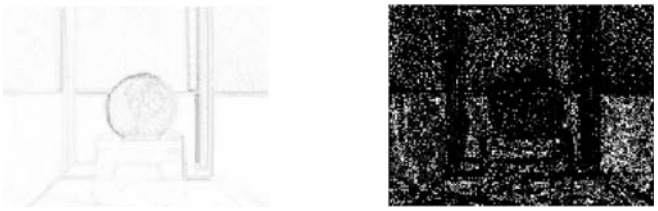

Fig. 2. The left-hand image is an EPP (Error Per Pixel) image: This image can be used to indicate to a user the locations within the scene which contain more errors. The right-hand image is an NPP Image: This image can be used to indicate which locations can be reconstructed exactly

### A. AEPP:

AEPPs are calculated according to Eq. (3). They show the average amount by which a pixel differers from its target value. A lower AEPP means that the network has successfully reorganised the information in the input frames to resemble that of the high level target. AEPPs are used in place of co-correlation for two reasons. Firstly AEPPs are more directly linked to pixel values which gives them more meaning. Secondly, the new image may be correlated correctly with the target spatially but may be consistently incorrect by several pixel levels. AEPPs will clearly show these errors whereas co-correlation would not.

$$\frac{\Sigma_{i=1}^{L}\Sigma_{j=1}^{M}\Sigma_{k=1}^{N}\left(|R_{ijk} - T_{ijk}|\right)}{L\,M\,N} = \text{AEPP} \tag{3}$$

The errors attributed to each pixel can also be shown by creating an EPP (Error Per Pixel) image. The darker the regions in the left-hand figure in Fig. 2, the more difficult to be reconstructed.

### B. NPP:

Equation (4) describes a count of the number of times a pixel value was exactly the same value as that in the target. Each of the three colour bands are treated separately so that a pixel does not have to be correct in all three bands to be classed as successful. These correct pixels can also be displayed graphically as shown in the right of Fig. 2 to give an indication of areas difficult to reconstruct.

$$\frac{\Sigma_{i,j,k=1,1,1}^{LMN}\left[(R_{ijk} - T_{ijk}) = 0\right]}{L\,M\,N} \times 100 = \%\ \text{correct} \tag{4}$$

These values give a good indication of whether an image is a good rendition of the original high resolution image. The importance of each varies based on what is required by the user. If an image has a bad NPP but a good AEPP then it suggests that the image is too blurred to gain any real advantage. If an image has a good NPP but a bad AEPP then it suggests that the SDN is capable of obtaining accurate results but is inconsistent in doing so.

## IV. Results

The ability of an SDN to perform multi-frame enhancement was tested by training it on several sets of frames. Each frame set consisted of two low resolution input frames and one high

resolution target image. After training the SDN was then given several more unseen sets of input frames to enhance. The results of these enhancements were compared to an original high quality version of the scene and a bi-cubic interpolation of one of the low resolution input frames. In order to assess the ability of the SDN to handle noise the low level frames had noise added before enhancement. The SDN was given the task of learning how to produce a high resolution image. The SDN was trained using four of the available nine image sets the other five image sets were used for the assessment. Each of the four test image sets was processed by the training algorithm sequentially. This was repeated several times until only small changes to the network structure were being made by the training runs.

### A. Image Acquisition

The test image sets were generated using a Polaroid PDC4350 digital camera. First a single high resolution (2304x1728) image was taken of an object. The resolution of the camera was then lowered (1024x768) without moving either the camera or the object. A low resolution image was then taken of the same object. The object was then subject several small movements (Approximately 1mm) and a low resolution image taken at every location. This produced a set of several low resolution images and a single high resolution image of the object. The low resolution images were then resized to 80x60 and the high resolution to 159x119 in order to make human visual comparisons easier and to speed up the training process.

### B. Network and Training Parameters

The following values were discovered during experimentation to be sufficient to produce a good solution without making the network large or the training slow. There is a population size of 2000 as this allows a fairly diverse pool of solutions. There are two layers of neurons, this allows for sufficient combination of inputs but should be increased if more than two frames are used. There were no hidden layers used for these results as the nodes in the output layer contain all functionality of one hidden layer. The lifecycle of a network was set to be 20 iterations, 10 iterations per frame. The maximum threshold weight is 100, thresholds below this should be sufficient to allow a wide range of values but should not allow all pixel values to be used. There are 2000 mutations meaning that every child produced will tend to be mutated. This large value has been chosen as the storage for the network weights allows for unused values. If these values are changed by the mutation the behaviour of the network is unaffected. Also many changes to the weights within the network will have little or no effect on its ability. Having the mutate rate this high therefore allows for mutations to occur which will effect the network characteristics more frequently. 600 children per generation means that 30% of the population will be replaced with new networks in every generation. When constructing the network evolution system this value allows the process to proceed at a fairly quick steady rate. There are 12 inputs (4 pixels $\times$ 3 colours) to the network for each frame in the set which for these tests was two frames. The corresponding output is 27 values (9 pixels $\times$ 3 colours).

### C. Results Obtained

The following figures and tables (Fig. 3 - Fig. 7) and (Table I - Table V) show results from the enhancement of five different image sets degraded with different amounts of Gaussian and drop-out noise (Drop-out noise forces a number of pixels to become set to maximum or minimum. The left-hand images are SDN results and the right-hand images bi-cubic interpolation results. Also shown are graphs illustrating the effect of increasing the amount of noise in the input images. Fig. 8 shows that as the amount of Gaussian noise is increased the SDN results gain more of an advantage over the Bi-cubic interpolation method. Fig. 9 shows that for drop-out noise the advantages of the SDN method increase greatly compared to that of Bi-cubic interpolation.

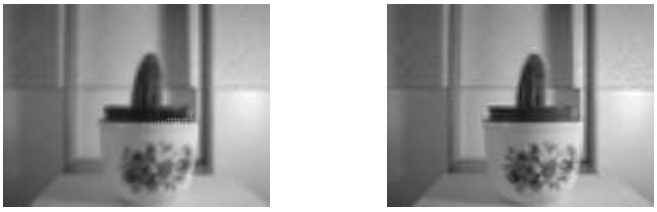

Fig. 3. Enhancement of Cactus image without added noise. The results in Table I show that the Correlation Coefficient and AEPP score higher on the bi-cubic interpolation images. However the number of correct pixels is higher using an SDN.

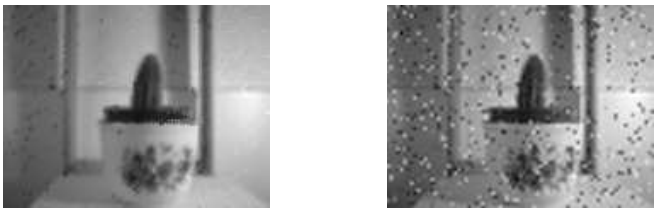

Fig. 4. Cactus image enhanced after addition of drop-out noise with variance of 0.05. SDN image on the left has removed almost all traces of the noise, whereas the Bi-cubic interpolation enhanced the noise along with the image. See table II for numerical comparisons.

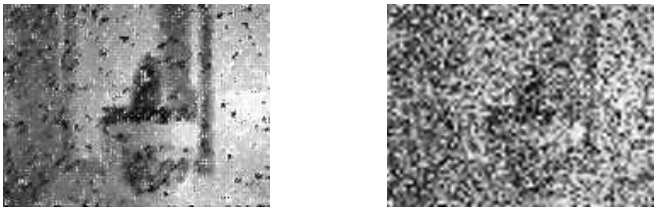

Fig. 5. Cactus image enhanced after addition of drop-out noise with variance of 0.5. With much more noise the Bi-cubic interpolation result is only just visible (right), whereas the SDN method clearly attempts to reconstruct the object (left). See table III for numerical comparisons.

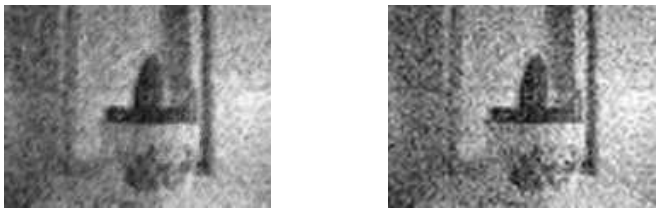

Fig. 6. Cactus image enhanced after addition of Gaussian noise with mean 0 and variance of 0.01. The SDN result (left) manages to remove more of the noise and produce a clearer image than bi-cubic interpolation (right). See table IV for numerical comparisons.

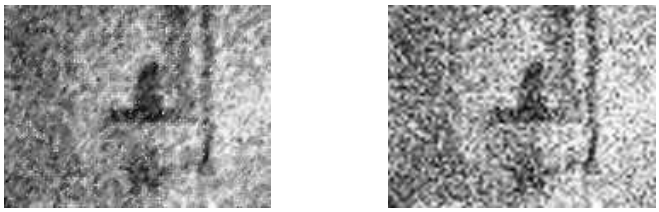

Fig. 7. Cactus image enhanced after addition of Gaussian noise with mean 0 and variance of 0.1. With more Gaussian noise both SDN and Bi-cubic interpolation struggle to produce a clear high resolution image. However the SDN result is clearer than the bi-cubic and also scores higher numerically (Table V)

TABLE I
NO NOISE ADDED

| Image | Cubic CC | SDN CC | Cubic AEPP | SDN AEPP | Cubic NPP | SDN NPP |
|---|---|---|---|---|---|---|
| Set 1 | 0.99 | 0.98 | 3.86 | 4.41 | 14.88 | 17.69 |
| Set 2 | 0.99 | 0.98 | 5.16 | 6.75 | 13.59 | 9.58 |
| Set 3 | 0.98 | 0.97 | 4.78 | 6.05 | 12.61 | 14.80 |
| Set 4 | 0.99 | 0.98 | 3.93 | 5.00 | 15.85 | 14.52 |
| Set 5 | 0.99 | 0.98 | 4.77 | 5.76 | 14.59 | 15.73 |
| Average | **0.99** | **0.98** | **4.50** | **5.59** | **14.30** | **14.47** |

TABLE II
DROP-OUT NOISE WITH VARIANCE 0.05

| Image | Cubic CC | SDN CC | Cubic AEPP | SDN AEPP | Cubic NPP | SDN NPP |
|---|---|---|---|---|---|---|
| Set 1 | 0.84 | 0.97 | 10.95 | 5.17 | 12.00 | 16.47 |
| Set 2 | 0.88 | 0.97 | 11.96 | 7.46 | 11.14 | 9.30 |
| Set 3 | 0.86 | 0.96 | 11.65 | 6.68 | 10.36 | 13.96 |
| Set 4 | 0.87 | 0.98 | 10.73 | 5.70 | 12.77 | 13.94 |
| Set 5 | 0.90 | 0.97 | 11.40 | 6.47 | 11.95 | 14.71 |
| Average | **0.87** | **0.97** | **11.34** | **6.30** | **11.64** | **13.68** |

TABLE III
DROP-OUT NOISE WITH VARIANCE 0.5

| Image | Cubic CC | SDN CC | Cubic AEPP | SDN AEPP | Cubic NPP | SDN NPP |
|---|---|---|---|---|---|---|
| Set 1 | 0.28 | 0.71 | 58.09 | 23.33 | 2.81 | 8.83 |
| Set 2 | 0.33 | 0.76 | 59.54 | 24.86 | 2.63 | 5.98 |
| Set 3 | 0.29 | 0.71 | 59.51 | 25.25 | 2.41 | 7.27 |
| Set 4 | 0.31 | 0.74 | 58.58 | 23.84 | 3.01 | 7.94 |
| Set 5 | 0.35 | 0.75 | 60.43 | 26.54 | 2.84 | 7.28 |
| Average | **0.31** | **0.73** | **59.23** | **24.77** | **2.74** | **7.46** |

TABLE IV
GAUSSIAN NOISE OF VARIANCE 0.01

| Image | Cubic CC | SDN CC | Cubic AEPP | SDN AEPP | Cubic NPP | SDN NPP |
|---|---|---|---|---|---|---|
| Set 1 | 0.81 | 0.87 | 23.72 | 17.77 | 1.36 | 1.93 |
| Set 2 | 0.86 | 0.90 | 23.65 | 19.20 | 1.32 | 1.80 |
| Set 3 | 0.83 | 0.87 | 23.80 | 19.00 | 1.41 | 1.78 |
| Set 4 | 0.84 | 0.89 | 23.32 | 18.25 | 1.39 | 1.84 |
| Set 5 | 0.88 | 0.91 | 23.62 | 18.81 | 1.44 | 1.82 |
| Average | **0.84** | **0.89** | **23.62** | **18.61** | **1.38** | **1.83** |

TABLE V
GAUSSIAN NOISE OF VARIANCE 0.1

| Image | Cubic CC | SDN CC | Cubic AEPP | SDN AEPP | Cubic NPP | SDN NPP |
|---|---|---|---|---|---|---|
| Set 1 | 0.50 | 0.60 | 49.59 | 33.87 | 0.66 | 1.03 |
| Set 2 | 0.58 | 0.67 | 48.21 | 34.67 | 0.67 | 1.00 |
| Set 3 | 0.52 | 0.61 | 49.32 | 34.94 | 0.64 | 1.00 |
| Set 4 | 0.54 | 0.64 | 48.51 | 34.39 | 0.66 | 0.95 |
| Set 5 | 0.62 | 0.71 | 48.32 | 34.73 | 0.68 | 0.96 |
| Average | **0.55** | **0.65** | **48.79** | **34.52** | **0.66** | **0.99** |

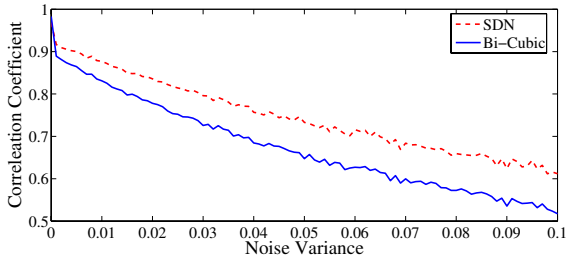


Fig. 8. As Gaussian noise variance increases the ability of the bi-cubic method to reconstruct the image falls away faster than the ability of the SDN.

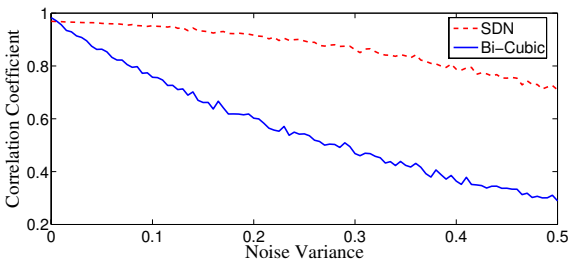


Fig. 9. As the amount of drop-out noise increases the benefit of the SDN becomes more clear as the difference between it and the bi-cubic interpolation increases.

## V. Conclusion

The focus of this paper was to ascertain if SDNs could be used to perform multi-frame enhancement to an acceptable level. It has been shown that they can outperform Bi-cubic interpolation of a single frame when noise is present. As the level of noise increases the SDNs remain the better enhancement technique. SDNs can successfully be trained to rely on information from two different frames in order to reconstruct a high resolution image successfully. The full range of uses for SDNs is not yet known. Given their abilities with temporal information they could be used for many time related applications. Their abilities with multi-frame image resolution enhancement could also be taken further. Further work with SDNs and image enhancement include utilising more than two frames for enhancement. Modifications to SDNs allowing the extra information available in stereo images to be used for enhancement should also be possible.

## References

[1] E. H. W. Meijering, "A chronology of interpolation," in *Proceedings of the IEEE*, vol. 90, no. 3, March 2002, pp. 319–342.

[2] T. M. Lehmann, C. Gonner, and K. Spitzer, "Survey: Interpolation methods in medical image processing," *IEEE Trans. Med. Imag.*, vol. 18, no. 11, pp. 1049–1075, November 1999.

[3] S. Battiato, G. Gallo, M. Mancuso, G. Messina, and F. Stanco, "Analysis and characterization of super-resolution reconstruction methods," in *SPIE Electronic Imaging 2003, Sensors, Cameras and Applications for Digital Photography*, vol. 5017B-37, 2003.

[4] E. H. W. Meijering, W. J. Niessen, and M. A. Viergever, "Quantitative evaluation of convolution-based methods for medical image interpolation," *Medical Image Analysis*, vol. 5, no. 2, pp. 111–126, June 2001.

[5] E. H. W. Meijering, "Spline interpolation in medical imaging: Comparison with other convolution based approaches," in *Signal Processing X: Theories and Applications, EUSIPCO 2000*, M. Gabbouj and P. Kuosmanen, Eds., vol. 4. The European Association for Signal Processing, Tampere, 2000, pp. 1989–1996.

[6] M. Unser, A. Aldroubi, and M. Eden, "Enlargement or reduction of digital images with minimum loss of information," *IEEE Trans. Image Processing*, vol. 4, no. 3, pp. 247–258, 1995.

[7] N. A. Dodgson, "Quadratic interpolation for image resampling," *IEEE Trans. Image Processing*, vol. 6, no. 9, pp. 1322–1326, September 1997.

[8] F. Anton, D. Mioc, and A. Fournier, "2d image reconstruction using natural neighbour interpolation," in *The Eighth International Conference on Computer Graphics and Visualization in Central Europe (WSCG' 2000)*, Febuary 2000, pp. 263–269.

[9] X. Li, "New edge-directed interpolation," *IEEE Trans. Image Processing*, vol. 10, pp. 1521–1527, October 2001.

[10] K. P. Hong, J. K. Paik, H. J. Kim, and C. H. Lee, "An edge-preserving image interpolation system for a digital camcorder," *IEEE Trans. Consumer Electron.*, vol. 42, no. 3, pp. 279–284, 1996.

[11] L. W. Leung, B. King, and V. Vohora, "Comparison of image data fusion techniques using entropy and ini," in *22nd Asian Converence of Remove Sensing*. Centre for Remote Imaging, Sensing and Processing CRISP, November 2001.

[12] G. Piella, "A general framework for multiresolution image fusion: from pixels to regions," *Information Fusion*, pp. 259–280, 2003.

[13] A. Waxman, A. Gove, D. Fay, J. Racamato, J. Carrick, M. Seibert, and D. Savoye, "Color night vision: Opponent processing in the fusion of visible and ir imagery," in *Neural Networks*, vol. 10, no. 1, 1997.

[14] H. Li, B. S. Manjunath, and S. K. Mitra, "Multi-sensor image fusion using the wavelet transform." in *ICIP (1)*, 1994, pp. 51–55.

[15] J. Nunez, X. Otazu, O. Fors, A. Prades, V. Pala, and R. Arbiol, "Multiresolution-based image fusion with additive wavelet decomposition," *IEEE Trans. Geosci. Remote Sensing*, vol. 37, pp. 1204–1211, 1999.

[16] A. Garzelli, "Wavelet-based fusion of optical and sar image data over urban area," in *PCV02*, 2002, p. B: 59.

[17] Y. Zhang and R. Wang, "Multi-resolution and multi-spectral image fusion for urban object extraction," in *Proceedings of XXth ISPRS Congress, Commission III*, 2004, pp. 960–966.

[18] P. S. Chavez, C. S. Stuart, and J. A. Anderson, "Comparison of three different methods to merge multiresolution and multispectral data: Landsat tm and spot panchromatic," *Photogrammetric Engineering and Remote Sensing*, vol. 57, pp. 295–303, 1991.

[19] C. K. Munechika, J. S. Warnick, and C. Salvaggo, "Resolution enhancement of multispectral image data to improve calssification accuracy," *Photographic Engineering and Remote Sensing*, vol. 59, no. 1, pp. 67–72, Januray 1993.

[20] F. Sroubek and J. Flusser, "Image fusion via probabilistic deconvolution," in *Third International Workshop on Pattern Recognition in Remote Sensing*, 2004, pp. 1–5.

[21] H. Kiiveri, "Image fusion with conditional probability networks for monitoring salinisation of farmland," 1998.

[22] G. Pajares and J. Manuel de la Cruz, "A wavelet based image fusion tutorial," *Pattern Recognition*, vol. 37, pp. 1855–1872, 2004.

[23] R. Tsai and T. Huang, *Multiframe Image Restoration and Registration*, R. Tsai and T. Huang, Eds. JAI Press, 1984, vol. 1.

[24] M. Irani and S. Peleg, "Super resolution from image sequences," *10th ICPR*, vol. 2, pp. 115–120, 1990.

[25] R. Schultz and R. Stevenson, "Extraction of high resolution frames from video sequences," *IEEE Trans. Image Processing*, vol. 5, no. 6, pp. 996–1011, 1996.

[26] S. Kim, N. Bose, and H. Valenzuela, "Recursive reconstruction of high resolution image from noisy undersampled multiframes," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 6, 1990.

[27] E. Salari and S. Zhang, "Integrated recurrent neural network for image resolution enhancement from multiple image frames," in *IEE Proceedings on Vision, Image and Signal Processing*, vol. 150, no. 5, 2003.

[28] M. Van Veelen, J. Nijhuis, and B. Spaanenburg, "Neural network approaches to capture temporal information," in *CASYS 1999*, 2000, pp. 361–371.

[29] R. Ivry and R. Spencer, "The neural representation of time," *Current Opinion in Neurobiology*, vol. 14, no. 2, pp. 225–232, 2004.

[30] A. Herz, "How is time represented in the brain," to Appear: Problems in Systems Neuroscience.

[31] J. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, pp. 179–211, 1990.

[32] S. Hochreiter and J. Schmidhuber, "Long short term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[33] D. Wang, X. Liu, and S. Ahalt, "On temporal generalization of simple recurrent networks," *Neural Networks*, vol. 9, pp. 1099–1118, 1996.