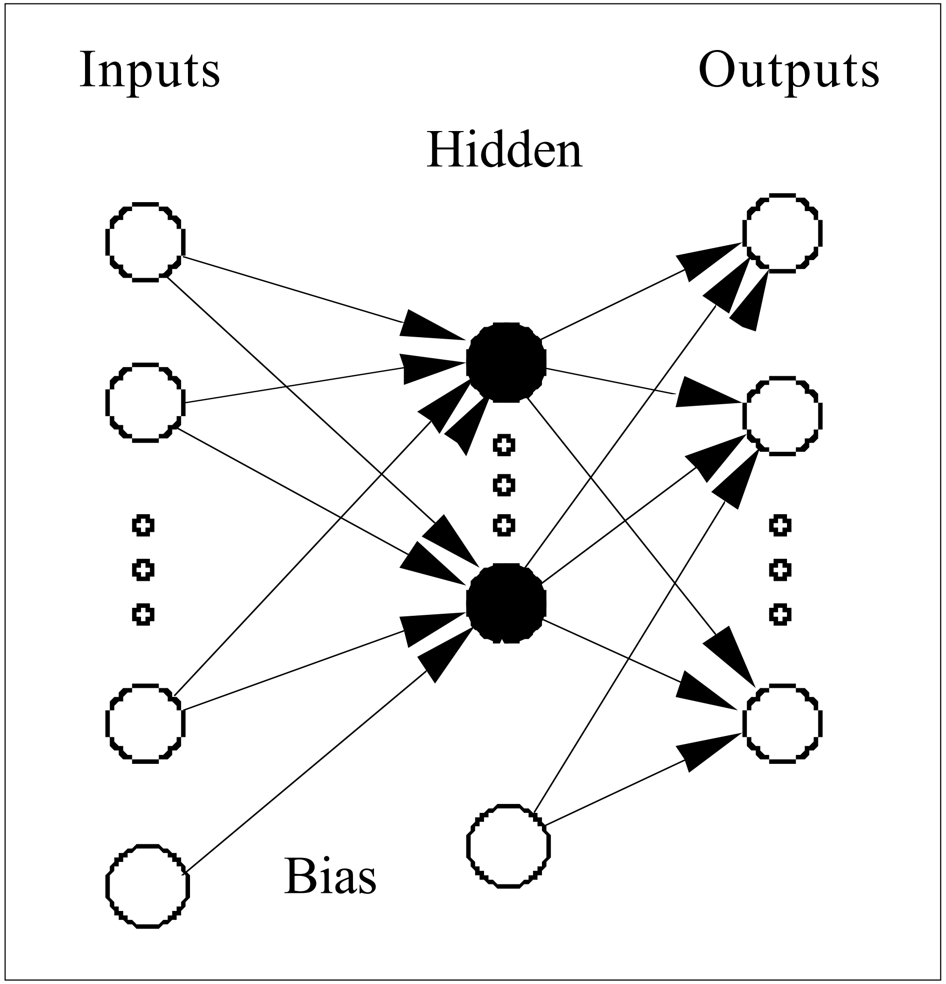
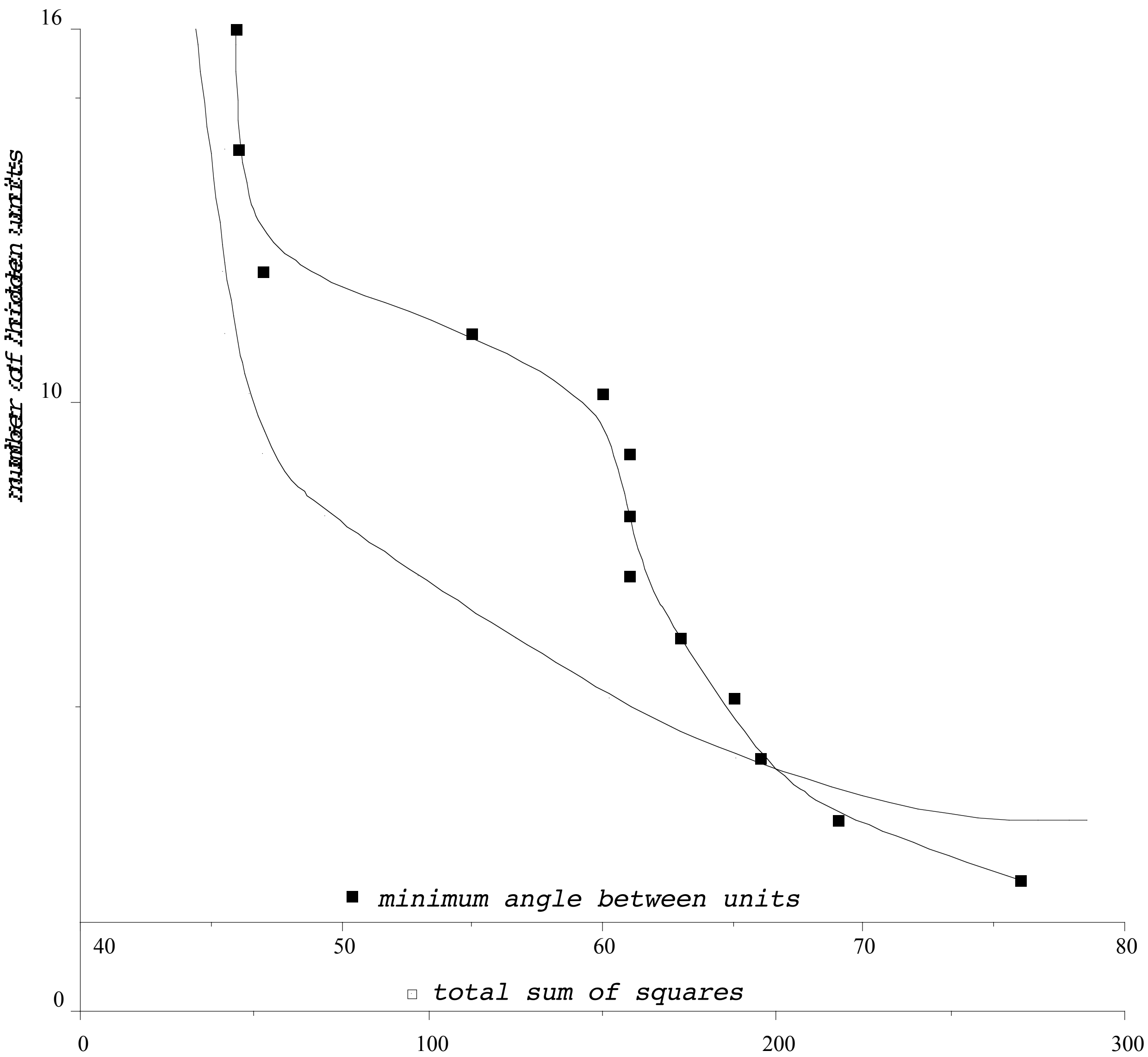
Connectionist compression

- Autoassociative networks
  - Number of inputs = outputs
  - Values of pattern
    inputs = outputs
  - Number of hidden neurons determines degree of compression
    - input to hidden: compression phase
    - hidden to output: decompression



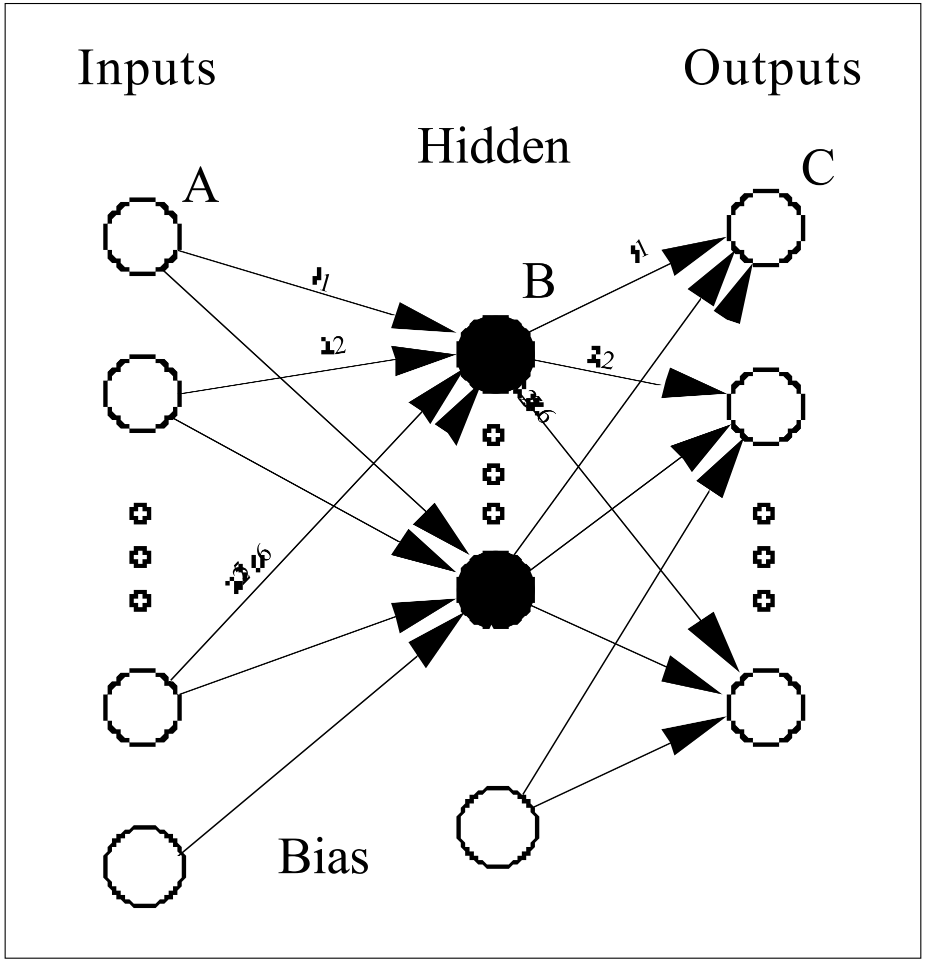
Previous work

- Pruning auto-associative net
  - Trained using backpropagation
  - Distinctiveness pruning measure
    - i.e. minimum angle between vectors
  - Image quality changes not uniform over pruning process



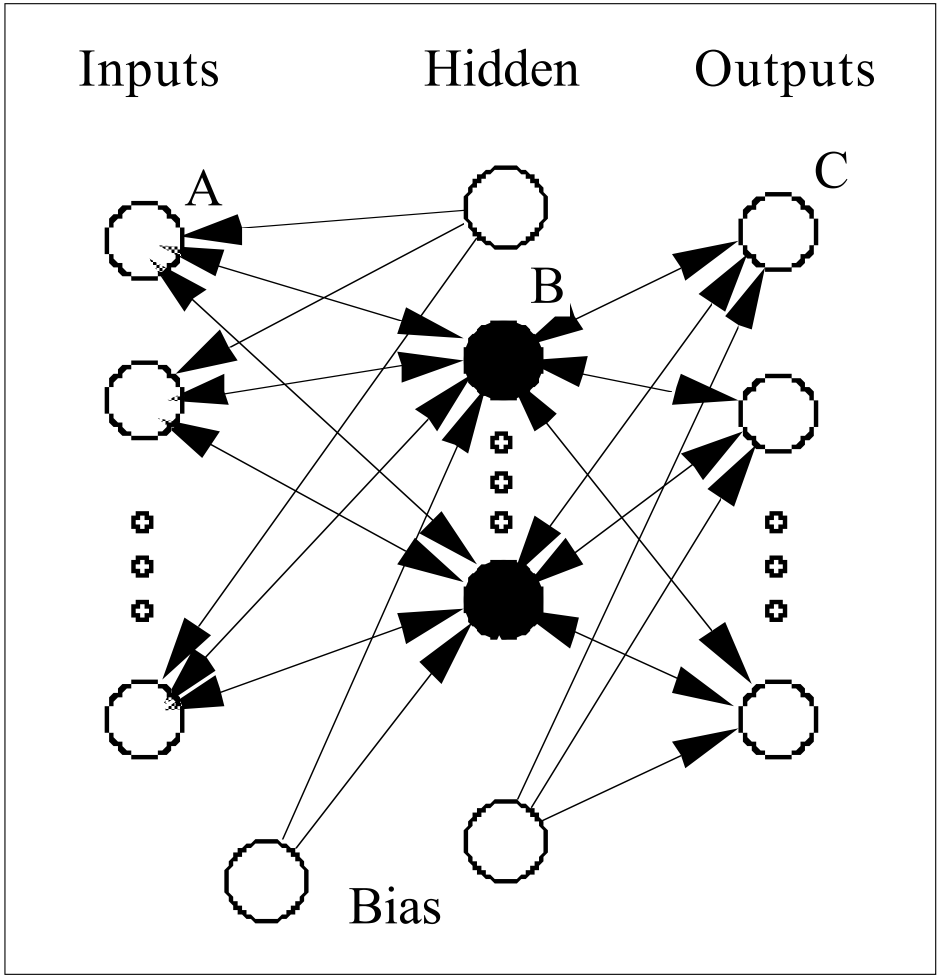
Shared Weights

- Same auto-associative topology
  - Changed meaning of weights:
    - link A → B = B → C
    - no. free parameters < no. weights
  - Reduce space of network weight configurations
    – require the compression function to be invertible.
    - harder to find function
    - expect better performance (than an approximation of an inverse function)



Bidirectional net

- Allow mapping: outputs to inputs
  - Network – weights same both dirs.
    - Input neurons bias weights for use only in reverse dir.
  - Training – backprop. for 1 direction at a time
    - Reverse: use output neurons as inputs
      - Values propagate to the ‘input’, error propagates back (to the ‘output)
    - Same patterns from training set
      - mapping is 1 to 1, input = target



Experiment description

- 64 x 64 greyscale image –'Lena'
  - Cut 16 non-overlapping patches (as in previous work)
  - Patches sufficiently different to require network to learn a suitably general compression function
  - Original image

- Overall objective
  - Compare compression schemes
  - Impose constraints on the allowed network weights
    - shared weights: rigidly enforce an invertible compression function
    - bidirectional training: a functional symmetry is imposed (less rigid)

Results – backprop.

- ◊ Network trained 2,000 epochs
- ◊ Select hidden unit to prune
- ◊ Network retrained 600 epochs
- ◊ Technique of cutting images into non-overlapping patches
  - ∆ Increase ease exposition / generalisation
  - ∆ Introduces extra edge effects on output
    - > necessarily impacts on image quality
    - > does not reflect degree of compression available using neural techniques

Results – shared wts.

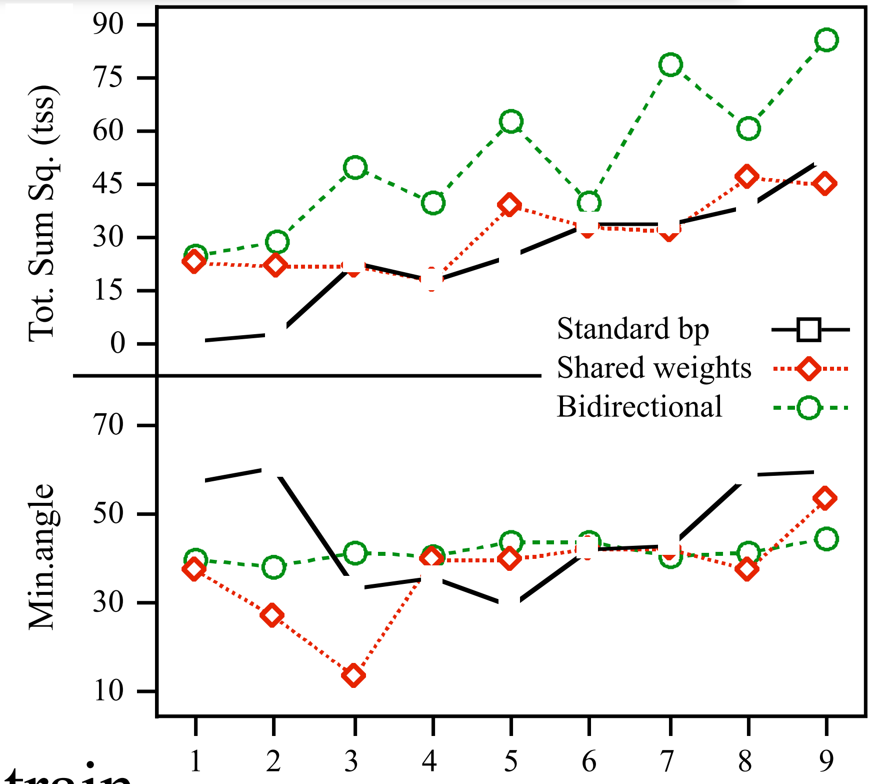
- ◊ Same wts., train 2000 - prune – train 600
- ◊ 1st image slightly lower quality than initial backprop. image.
  - ∆ Not surprising: no. weights same, but no. free parameters is (approx.) halved
  - ∆ Constrains the functions that network can implement
- ◊ Function more robust – degradation in image quality is slower
  - ∆ Final image better than backprop.

Results – bidir. train

- ◊ Same wts., train 2000 - prune – train 600
- ◊ 1st image even lower quality than using the shared weights method
  - ▵ Surprising: expected result between the backprop and shared weights results
  - ▵ Weaker constraint on the function the neural network implements
- ◊ Degradation in image quality on pruning similar to backprop.

Comparison of results

- ◊ Min. angle values not related to image quality, especially for bidirectional training
- ◊ Image 6 most similar on tss



- ◊ backprop. shared wts bidir. train



Further experiments

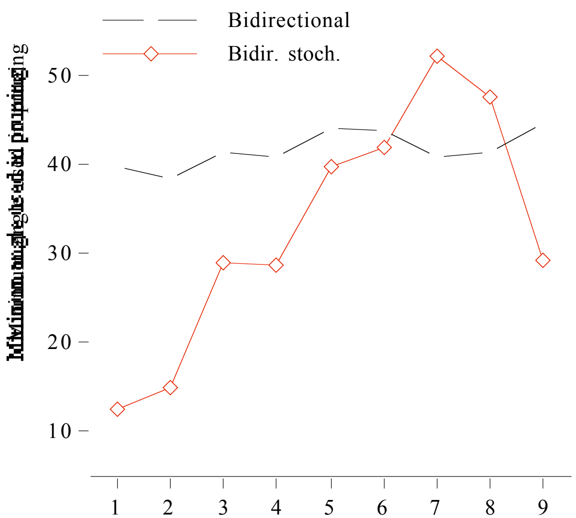
- Length of training
  - Double training on backprop.
  - No improvement found
- Observation: bidir. train tss values oscillate
  - ? a relationship between changes of direction and tss oscillation ? (no)
  - Bidir. training: switch direction each pattern rather than epoch
    - Repeat same pattern in both directions
    - Double training on each pattern

Bidir. – stochastic

◊ Switch direction on each pattern

◊ Substantial improvement in quality

◊ The min. angle measure now shows more of expectable variation



Bidir. stoch. repeat pat.

- ◊ Network trained on each pattern in both directions (no. epochs halved)
- ◊ Results are worse, but still better than epoch bidir. train
- ◊ Effect of repeating patterns is to reduce the stochastic effect (cf analogy of momentum with stochastic backprop.)

Bidir. stoch. double pat.

- ◊ Network trained on a pattern twice then switch direction (epochs $^1/_2$'ed)
- ◊ Results are better than our original bidirectional stochastic example
  - ▵ Increase local effect of individual patterns on weights
  - ▵ (Three presentations of patterns gives slight reduction in quality)

Summary

- Summary of results (image 6)
- Advantages for generalisation for shared weights and bidirectional training probably derive from:
  - Reduction in free parameters
  - Faster training of input to hidden weights
    - the weights receive stronger feedback than with standard backprop. training

backprop.

2*backprop.

shared wts.

bidir. train

bidir. stoch.

bidir.2*sto.

Is it gradient descent?

- Yes, of the total error surface

$$E_{TOT} = E_F + E_B$$

- Standard stochastic backprop. approximates gradient descent – i.e. approximate by $\frac{\partial E_F}{\partial w_{i,j}}(\boldsymbol{w}) = \sum_X \frac{\partial E_F^X}{\partial w_{i,j}}(\boldsymbol{w})$

$$\frac{\partial E_F}{\partial w_{i,j}}(\boldsymbol{w}) \approx \frac{\partial E_F^{X_1}}{\partial w_{i,j}}(\boldsymbol{w}) + \frac{\partial E_F^{X_2}}{\partial w_{i,j}}(\boldsymbol{w} + \Delta \boldsymbol{w}_1) + \frac{\partial E_F^{X_3}}{\partial w_{i,j}}(\boldsymbol{w} + \Delta \boldsymbol{w}_1 + \Delta \boldsymbol{w}_2) + \ldots$$

where Δw_i is change in weight vector after pattern X_i is trained

It is gradient descent.

- Bidirectional stochastic training are either working out $\frac{\partial E_F^{X_k}}{\partial w_{i,j}}$ or $\frac{\partial E_B^{X_k}}{\partial w_{i,j}}$
- Both are partial derivatives of $\frac{\partial E_{TOT}^{X_k}}{\partial w_{i,j}}$
- However: $\min E_{TOT} \geq \min E_F + \min E_B$

  so local minima E_{TOT} and E_F may not be located at same point.
- Also at min E_F, E_B can be large.
- For auto-associative network, can chose minimum of E_F and E_B!