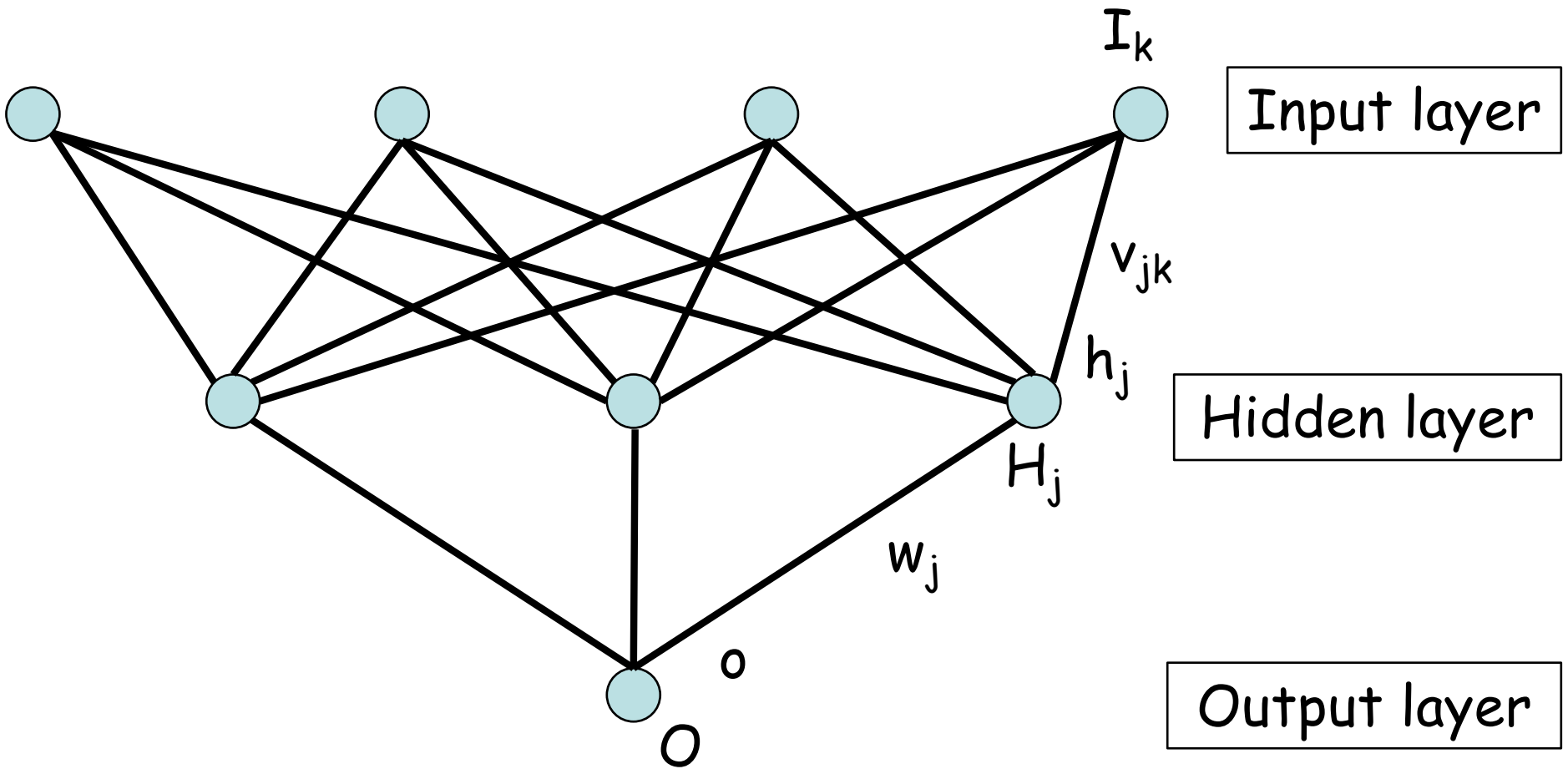
Artificial Neural Networks 3
Tricks for improved learning

Morten Nielsen
Department of Health Technology,
DTU

Network architecture

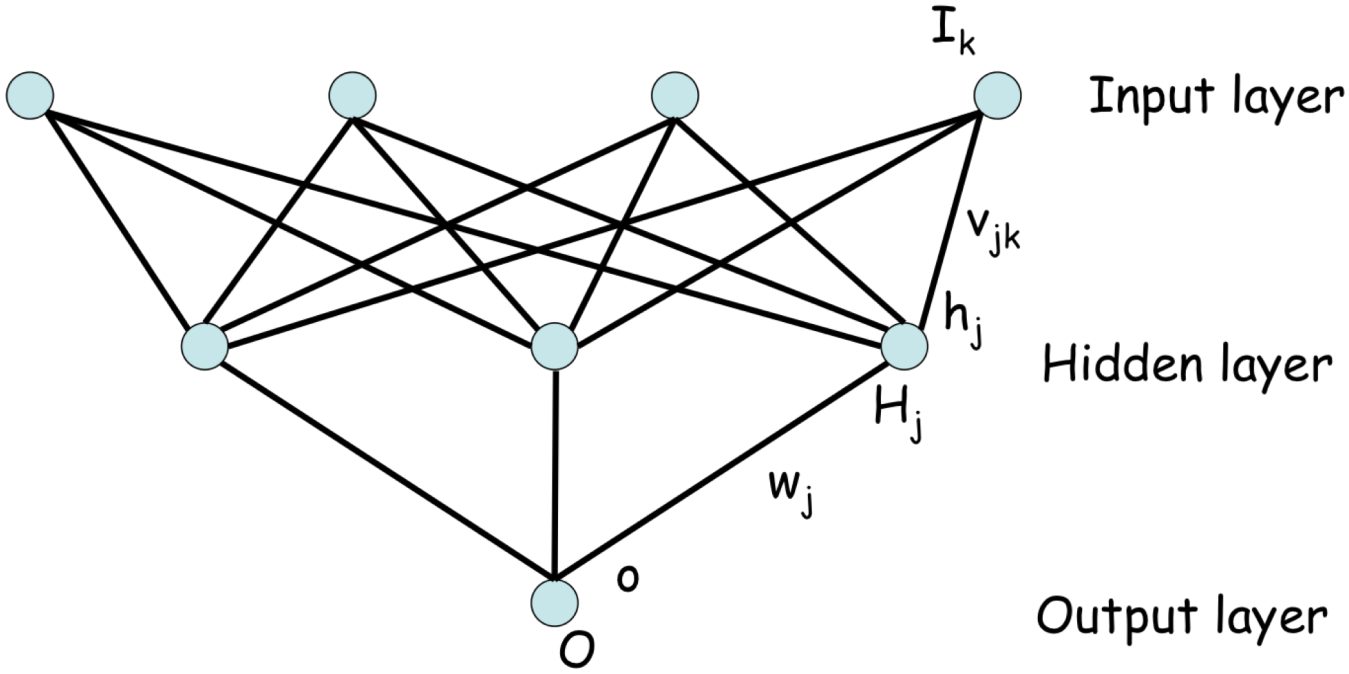


The conventional back-prop

$$\Delta w_i = -\varepsilon \cdot \frac{\partial E}{\partial w_i}$$

$$\Delta v_{jk} = -\varepsilon \cdot \frac{\partial E}{\partial v_{jk}}$$

$$E = \tfrac{1}{2} \cdot (O - t)^2$$

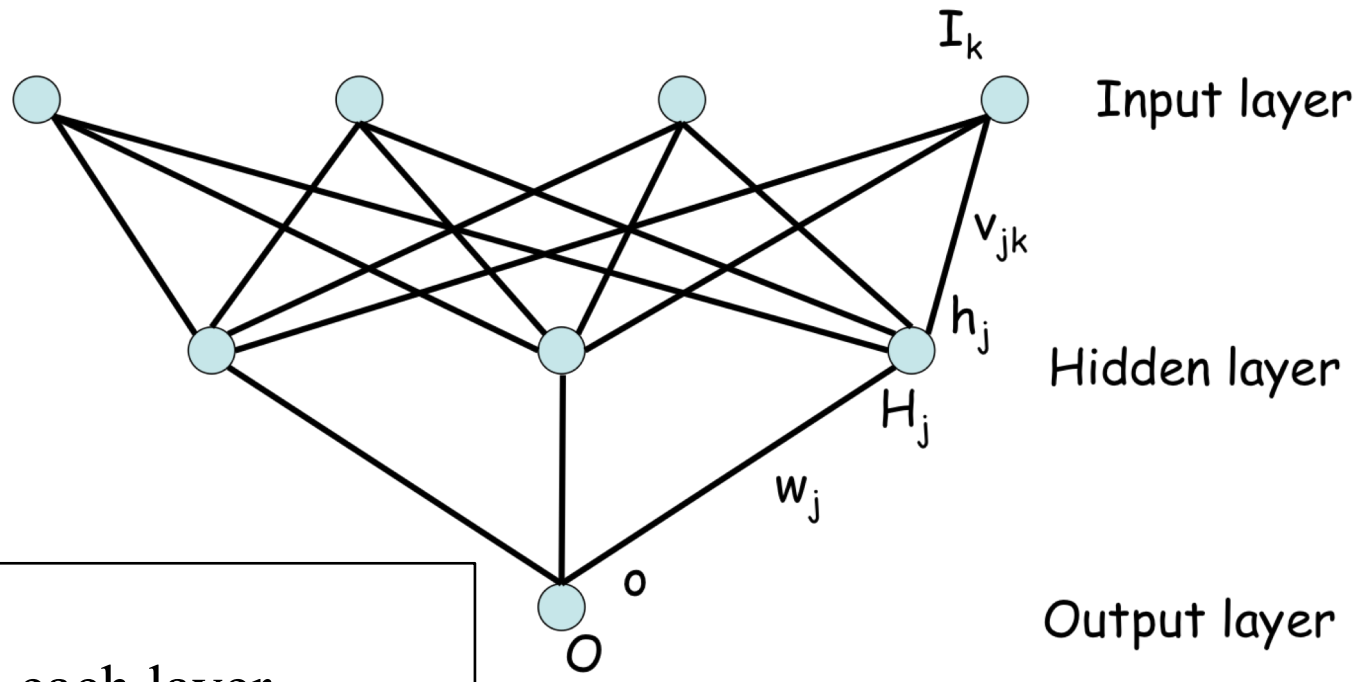


Regularization

$$\Delta w_i = -\varepsilon \cdot \frac{\partial E}{\partial w_i}$$

$$\Delta v_{jk} = -\varepsilon \cdot \frac{\partial E}{\partial v_{jk}}$$

$$E = \tfrac{1}{2} \cdot (O - t)^2 + \sum_L (\lambda_L \cdot \sum_{n_L} w_n^2)$$



Issues:

- Different lambda is each layer
- Include bias values in regularization?

Batch training

Generally each parameter update in GD is computed w.r.t a few training examples or a minibatch as opposed to a single example.

The reason for this is twofold: first this reduces the variance in the parameter update and can lead to more stable convergence, second this allows the computation to take advantage of highly optimized matrix operations that should be used in a well vectorized computation of the cost and gradient.

http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/

Changing the learning rate

Choosing the proper learning rate and schedule (i.e. changing the value of the learning rate as learning progresses) can be fairly difficult. One standard method that works well in practice is to use a small enough constant learning rate that gives stable convergence in the initial epoch (full pass through the training set) or two of training and then halve the value of the learning rate as convergence slows down. An even better approach is to evaluate a held out set after each epoch and anneal the learning rate when the change in objective between epochs is below a small threshold. This tends to give good convergence to a local optima. Another commonly used schedule is to anneal the learning rate at each iteration t as $\frac{a}{b+t}$ where a and b dictate the initial learning rate and when the annealing begins respectively. More sophisticated methods include using a backtracking line search to find the optimal update.

http://ufldl.stanford.edu/tutorial/supervised/OptimizationStochasticGradientDescent/

Using different activation functions – g(x)

- Sigmoid
- Hyperbolic tangent
- …

- Note, if you change activation function in the output layer, be sure to also rescale the target values

Momentum

One method of speeding up training per-dimension is the momentum method. This is perhaps the simplest extension to SGD that has been successfully used for decades. The main idea behind momentum is to accelerate progress along dimensions in which gradient consistently point in the same direction and to slow progress along dimensions where the sign of the gradient continues to change.

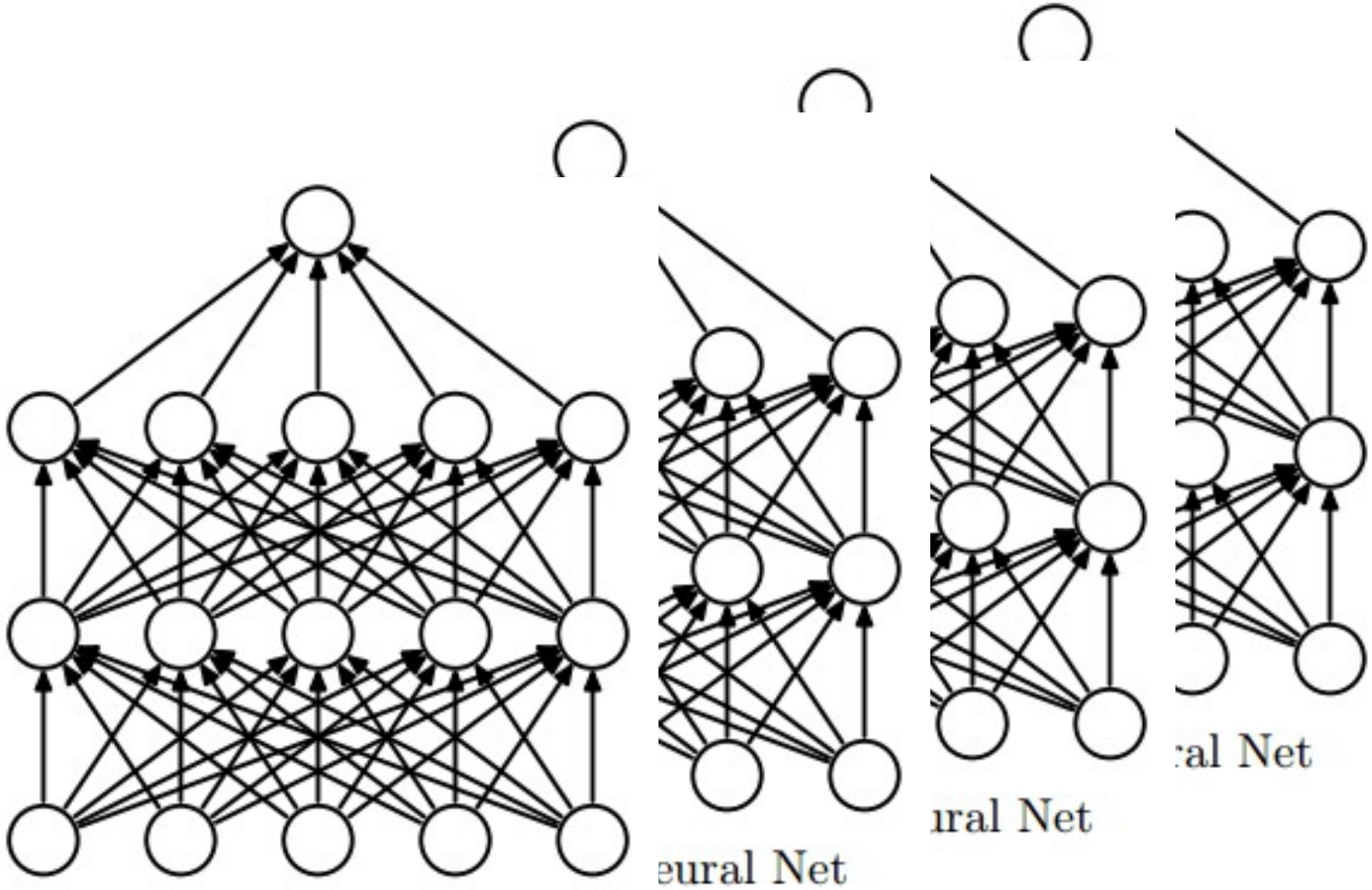
$$\Delta w_i^t = -\varepsilon \cdot \frac{\partial E}{\partial w_i} + \rho \cdot \Delta w_i^{t-1}$$

$$\Delta v_{jk}^t = -\varepsilon \cdot \frac{\partial E}{\partial v_{jk}} + \rho \cdot \Delta v_{jk}^{t-1}$$

$$E = \tfrac{1}{2} \cdot (O - t)^2$$

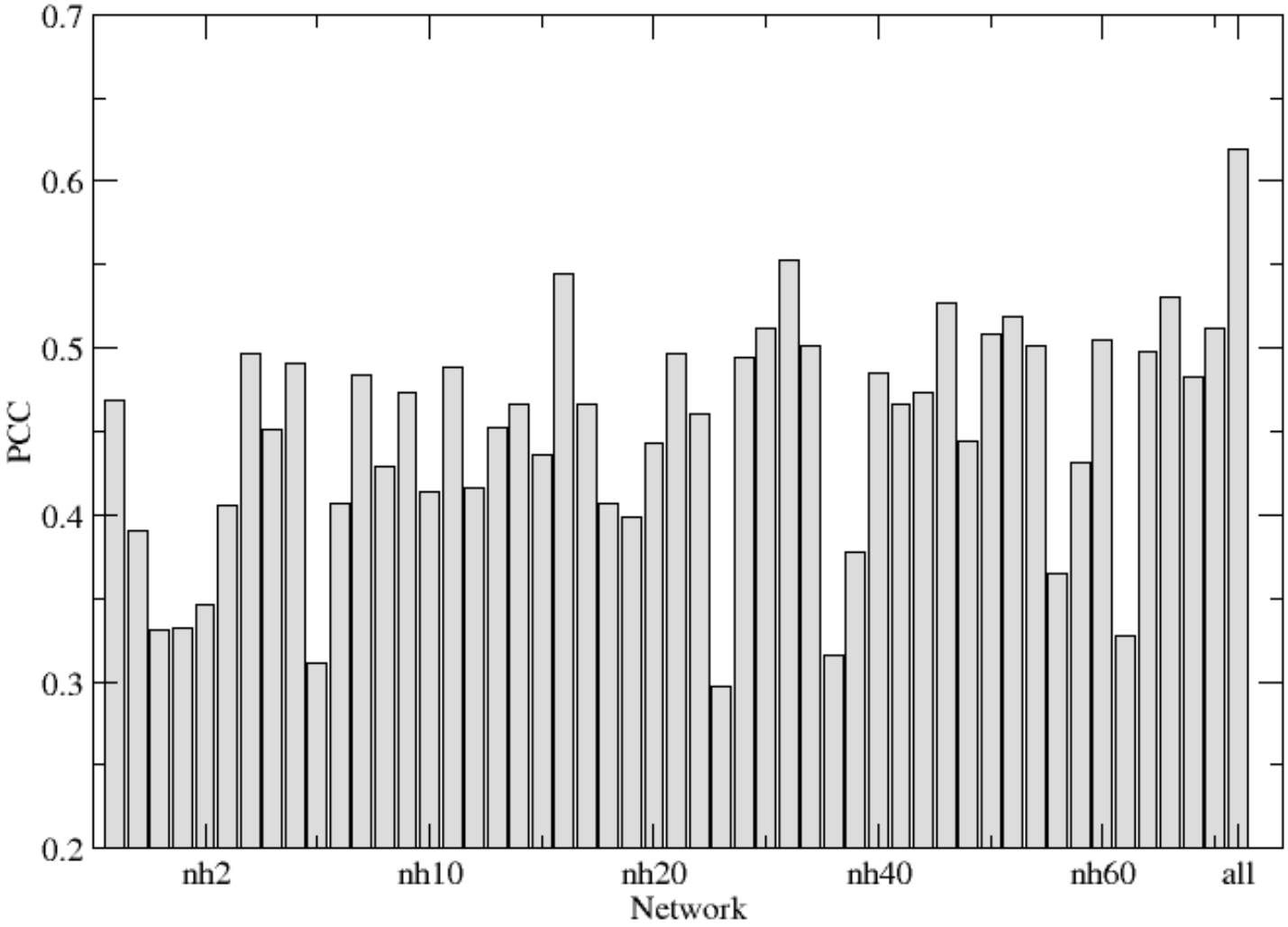
ADADELTA: AN ADAPTIVE LEARNING RATE METHOD, Matthew D. Zeiler

Network ensembles

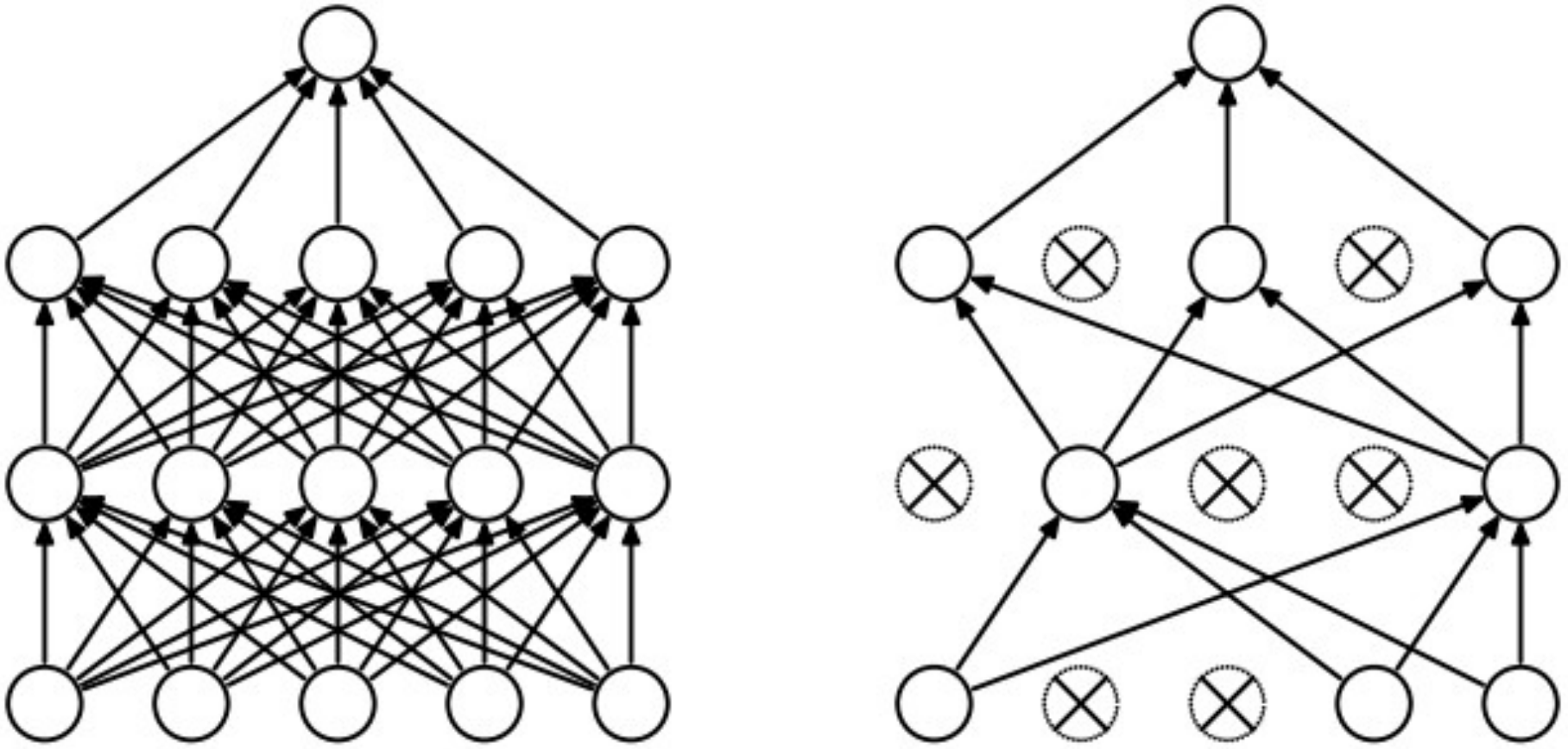


(a) Standard Neural Net

Network ensembles



Drop out (one network containing the ensemble)



(a) Standard Neural Net

(b) After applying dropout.