distribution are outputs of the network, with the mixture weight probabilities produced by a softmax unit, and the variances parametrized so that they are positive. Stochastic gradient descent can be numerically ill-behaved due to the interactions between the conditional means μ_i and the conditional variances σ_i^2. To reduce this difficulty, Uria *et al.* (2013) use a pseudo-gradient that replaces the gradient on the mean, in the back-propagation phase.

Another very interesting extension of the neural auto-regressive architectures gets rid of the need to choose an arbitrary order for the observed variables (Murray and Larochelle, 2014). In auto-regressive networks, the idea is to train the network to be able to cope with any order by randomly sampling orders and providing the information to hidden units specifying which of the inputs are observed (on the right side of the conditioning bar) and which are to be predicted and are thus considered missing (on the left side of the conditioning bar). This is nice because it allows one to use a trained auto-regressive network to *perform any inference problem* (i.e. predict or sample from the probability distribution over any subset of variables given any subset) extremely efficiently. Finally, since many orders of variables are possible ($n!$ for n variables) and each order o of variables yields a different $p(\mathbf{x} \mid o)$, we can form an ensemble of models for many values of o:

$$p_{\text{ensemble}}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^{k} p(\mathbf{x} \mid o^{(i)}). \tag{20.84}$$

This ensemble model usually generalizes better and assigns higher probability to the test set than does an individual model defined by a single ordering.

In the same paper, the authors propose deep versions of the architecture, but unfortunately that immediately makes computation as expensive as in the original neural auto-regressive neural network (Bengio and Bengio, 2000b). The first layer and the output layer can still be computed in $O(nh)$ multiply-add operations, as in the regular NADE, where h is the number of hidden units (the size of the groups h_i, in figures 20.10 and 20.9), whereas it is $O(n^2h)$ in Bengio and Bengio (2000b). However, for the other hidden layers, the computation is $O(n^2h^2)$ if every "previous" group at layer l participates in predicting the "next" group at layer $l+1$, assuming n groups of h hidden units at each layer. Making the i-th group at layer $l+1$ only depend on the i-th group, as in Murray and Larochelle (2014) at layer l reduces it to $O(nh^2)$, which is still h times worse than the regular NADE.