is to train a decision tree in which each node uses a neural network to make the splitting decision (Guo and Gelfand, 1992), though this has typically not been done with the primary goal of accelerating inference computations.

In the same spirit, one can use a neural network, called the **gater** to select which one out of several **expert networks** will be used to compute the output, given the current input. The first version of this idea is called the **mixture of experts** (Nowlan, 1990; Jacobs *et al.*, 1991), in which the gater outputs a set of probabilities or weights (obtained via a softmax nonlinearity), one per expert, and the final output is obtained by the weighted combination of the output of the experts. In that case, the use of the gater does not offer a reduction in computational cost, but if a single expert is chosen by the gater for each example, we obtain the **hard mixture of experts** (Collobert *et al.*, 2001, 2002), which can considerably accelerate training and inference time. This strategy works well when the number of gating decisions is small because it is not combinatorial. But when we want to select different subsets of units or parameters, it is not possible to use a "soft switch" because it requires enumerating (and computing outputs for) all the gater configurations. To deal with this problem, several approaches have been explored to train combinatorial gaters. Bengio *et al.* (2013b) experiment with several estimators of the gradient on the gating probabilities, while Bacon *et al.* (2015) and Bengio *et al.* (2015a) use reinforcement learning techniques (policy gradient) to learn a form of conditional dropout on blocks of hidden units and get an actual reduction in computational cost without impacting negatively on the quality of the approximation.

Another kind of dynamic structure is a switch, where a hidden unit can receive input from different units depending on the context. This dynamic routing approach can be interpreted as an attention mechanism (Olshausen *et al.*, 1993). So far, the use of a hard switch has not proven effective on large-scale applications. Contemporary approaches instead use a weighted average over many possible inputs, and thus do not achieve all of the possible computational benefits of dynamic structure. Contemporary attention mechanisms are described in section 12.4.5.1.

One major obstacle to using dynamically structured systems is the decreased degree of parallelism that results from the system following different code branches for different inputs. This means that few operations in the network can be described as matrix multiplication or batch convolution on a minibatch of examples. We can write more specialized sub-routines that convolve each example with different kernels or multiply each row of a design matrix by a different set of columns of weights. Unfortunately, these more specialized subroutines are difficult to implement efficiently. CPU implementations will be slow due to the lack of cache