loss functions, the gradient can be computed efficiently (Vincent *et al.*, 2015), but the standard cross-entropy loss applied to a traditional softmax output layer poses many difficulties.

Suppose that $\boldsymbol{h}$ is the top hidden layer used to predict the output probabilities $\hat{\boldsymbol{y}}$. If we parametrize the transformation from $\boldsymbol{h}$ to $\hat{\boldsymbol{y}}$ with learned weights $\boldsymbol{W}$ and learned biases $\boldsymbol{b}$, then the affine-softmax output layer performs the following computations:

$$a_i = b_i + \sum_j W_{ij} h_j \quad \forall i \in \{1, \ldots, |\mathbb{V}|\}, \tag{12.8}$$

$$\hat{y}_i = \frac{e^{a_i}}{\sum_{i'=1}^{|\mathbb{V}|} e^{a_{i'}}}. \tag{12.9}$$

If $\boldsymbol{h}$ contains n_h elements then the above operation is $O(|\mathbb{V}|n_h)$. With n_h in the thousands and $|\mathbb{V}|$ in the hundreds of thousands, this operation dominates the computation of most neural language models.

12.4.3.1 Use of a Short List

The first neural language models (Bengio *et al.*, 2001, 2003) dealt with the high cost of using a softmax over a large number of output words by limiting the vocabulary size to 10,000 or 20,000 words. Schwenk and Gauvain (2002) and Schwenk (2007) built upon this approach by splitting the vocabulary $\mathbb{V}$ into a **shortlist** $\mathbb{L}$ of most frequent words (handled by the neural net) and a tail $\mathbb{T} = \mathbb{V} \backslash \mathbb{L}$ of more rare words (handled by an n-gram model). To be able to combine the two predictions, the neural net also has to predict the probability that a word appearing after context C belongs to the tail list. This may be achieved by adding an extra sigmoid output unit to provide an estimate of $P(i \in \mathbb{T} \mid C)$. The extra output can then be used to achieve an estimate of the probability distribution over all words in $\mathbb{V}$ as follows:

$$\begin{aligned} P(y = i \mid C) =& 1_{i \in \mathbb{L}} P(y = i \mid C, i \in \mathbb{L})(1 - P(i \in \mathbb{T} \mid C)) \\ &+ 1_{i \in \mathbb{T}} P(y = i \mid C, i \in \mathbb{T}) P(i \in \mathbb{T} \mid C) \end{aligned} \tag{12.10}$$

where $P(y = i \mid C, i \in \mathbb{L})$ is provided by the neural language model and $P(y = i \mid C, i \in \mathbb{T})$ is provided by the n-gram model. With slight modification, this approach can also work using an extra output value in the neural language model's softmax layer, rather than a separate sigmoid unit.

An obvious disadvantage of the short list approach is that the potential generalization advantage of the neural language models is limited to the most frequent