**Exact Line Rule Override Grid**

A long-short term memory model, adapted from the original figure in [14]. Learned weights control how data enter and leave and are deleted through the use of gates.

The cell state vector activation is given by the following equation:where  represents the Hadamard product. Finally, the output gate vector activation is given by the following equation:

As it has been already stated, LSTM gated cells in RNNs have internal recurrence, besides the outer recurrence of RNNs. Cells store an internal state, which can be written to and read from them. There are gates controlling how data enter and leave and are deleted from this cell state. Those gates act on the signals they receive, and, similar to a standard neural network, they block or pass on information based on its strength and importance using their own sets of weights. Those weights, as the weights that modulate input and hidden states, are adjusted via the recurrent network’s learning process. The cells learn when to allow data to enter and leave or be deleted through the iterative process of making guesses, backpropagating error, and adjusting weights via gradient descent. This type of model architecture allows successful learning from long sequences, helping to capture diverse time scales and remote dependencies. Practical aspects on the use of LSTMs and other deep learning architectures can be found in [18].

2.2. Unsupervised Learning

Unsupervised learning aims towards the development of models that are capable of extracting meaningful and high-level representations from high-dimensional sensory unlabeled data. This functionality is inspired by the visual cortex which requires very small amount of labeled data.

Deep Generative Models such as Deep Belief Networks (DBNs) [19, 20] allow the learning of several layers of nonlinear features in an unsupervised manner. DBNs are built by stacking several Restricted Boltzmann Machines (RBMs) [21, 22], resulting in a hybrid model in which the top two layers form a RBM and the bottom layers act as a directed graph constituting a Sigmoid Belief Network (SBN). The learning algorithm proposed in [19] is supposed to be one of the first efficient ways of learning DBNs by introducing a greedy layer-by-layer training in order to obtain a deep hierarchical model. In this greedy learning procedure, the hidden activity patterns obtained in the current layer are used as the “visible” data for training the RBM of the next layer. Once the stacked RBMs have been learned and combined to form a DBN, a fine-tuning procedure using a contrastive version of the wake-sleep algorithm [23] is applied.

For a better understanding, the theoretical details of RBMs are provided in the following equations. The energy of a joint configuration  can be calculated as follows:where  represent the model parameters.  are the “visible” stochastic binary units, which are connected to the “hidden” stochastic binary units . The bias terms are denoted by  for the visible units and  for the hidden units.

The probability of a joint configuration over both visible and hidden units depends on the energy of that joint configuration and is given by ([10](https://www.hindawi.com/journals/js/2017/3296874/#EEq12)), where  represents the partition function (see ([11](https://www.hindawi.com/journals/js/2017/3296874/#EEq12))):

The probability assigned by the model to a visible vector  can be computed as expressed in the following equation:

The conditional distributions over hidden variables  and visible variables  can be extracted using ([13](https://www.hindawi.com/journals/js/2017/3296874/#EEq15)). Once a training sample is presented to the model, the binary states of the hidden variables are set to 1 with probability given by ([14](https://www.hindawi.com/journals/js/2017/3296874/#EEq15)). Analogously, once the binary states of the hidden variables are computed, the binary states of the visible units are set to 1 with a probability given by ([15](https://www.hindawi.com/journals/js/2017/3296874/#EEq15)).where  is the logistic function.

For training the RBM model, the learning is conducted by applying the Contrastive Divergence algorithm [22], in which the update rule applied to the model parameters is given by the following equation:where  is the learning rate,  represents the expected value of the product of visible and hidden states at thermal equilibrium, when training data is presented to the model, and  is the expected value of the product of visible and hidden states after running a Gibbs chain.

Deep neural networks can also be utilized for dimensionality reduction of the input data. For this purpose, deep “autoencoders” [24, 25] have been shown to provide successful results in a wide variety of applications such as document retrieval [26] and image retrieval [27]. An autoencoder (see Figure [4](https://www.hindawi.com/journals/js/2017/3296874/fig4/)) is an unsupervised neural network in which the target values are set to be equal to the inputs. Autoencoders are mainly composed of an “encoder” network, which transforms the input data into a low-dimensional code, and a “decoder” network, which reconstructs the data from the code. Training these deep models involves minimizing the error between the original data and its reconstruction. In this process, the weights initialization is critical to avoid reaching a bad local optimum; thus some authors have proposed a pretrained stage based on stacked RBMs and a fine-tuning stage using backpropagation [24, 27]. In addition, the encoder part of the autoencoder can serve as a good unsupervised nonlinear feature extractor. In this field, the use of Stacked Denoising Autoencoders (SDAE) [25] has been proven to be an effective unsupervised feature extractor in different classification problems. The experiments presented in [25] showed that training denoising autoencoders with higher noise levels forced the model to extract more distinctive and less local features.

