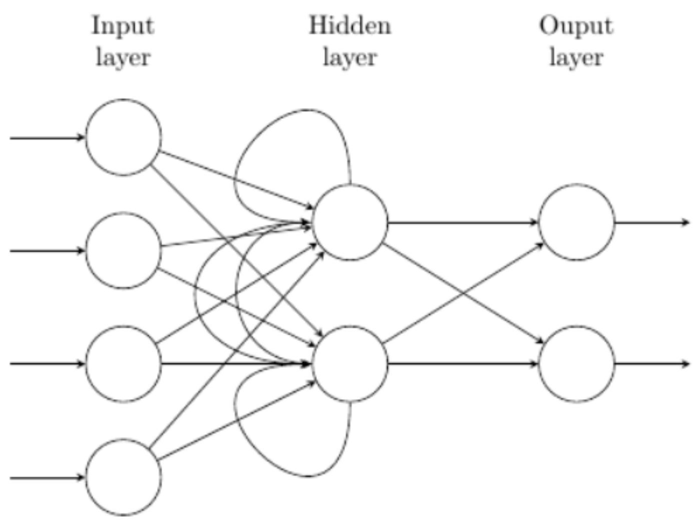
Recurrent Neural Network Implementation Issues

A recurrent neural network (RNN) is a class of Artificial Neural Network where connections between nodes form a directed graph along a sequence. Unlike feedforward neural networks, RNNs can use their internal state to process sequence of inputs. This makes RNNs applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition. RNNs can be applied in machine translation. For instance, you can feed the network an English sentence paired with its French translation. With enough training you can give the network an English sentence and it will translate it to French! This model is called a Sequence 2 Sequences model or Encoder Decoder model. This model uses a word embedding layer (GloVe or Word2Vec) that takes a bunch of words and creates a weighted matrix that allows similar words to be correlated to each other. This helps make the RNN more accurate.

The following are the implementation issues associated with RNNs:

- RNN faces the problem of gradient vanishing and exploding problems. It makes training of RNN difficult since it cannot process very long sequences if using tanh as its activation function and it is very unstable if using ReLU (rectified linear unit) as its activation function.
- RNNs cannot be stacked into very deep models. This is mainly because of the saturated activation function used in RNN models, making the gradient decay over layers.
- RNNs are very difficult to train. They have the capacity to learn from long sequences to retain information about their hidden state for a long time. It is complex to get them to efficiently use

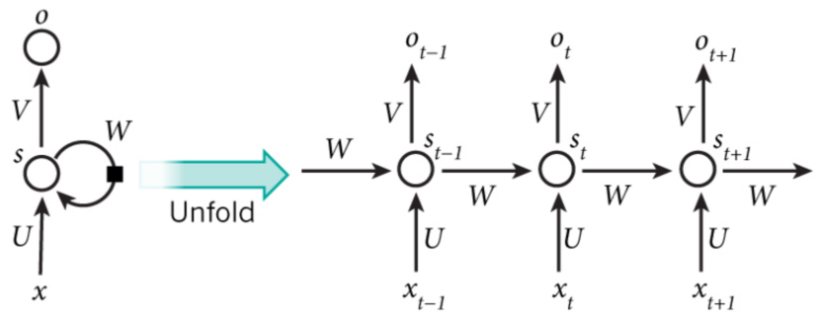
their ability.



The figure above depicts the working of an RNN with the recurrence in the hidden layer

- RNNs fail to understand the context behind an input. Something that was said long before, cannot be recalled when making predictions in the present.
- RNN remembers things for just small durations of time, i.e. if we need the information after a small time it may be reproducible, but once a lot of words are fed in, this information gets lost somewhere.
- RNNs get stuck in local minima during gradient descent, thus discovering sub-optimal solutions and facing network stagnancy (Bianchini, Gori, Maggini, 1994).
- RNNs faces slow speed of convergence being not definite to find the global minimum of the error function since during gradient descent it may get stuck in local minima (Nawi, Khan, Rehman, Chiroma, Herawan).

- Unitary RNN requires more research and comparative study.



The figure above depicts an RNN and the unfolding in time of the computation involved in its forward computation

- A major constraint with the attention mechanism is that only one context vector is produced for the entire time-series data. The entire sequence of data must be read into the model before the context vector can be generated. In other words, the attention mechanism is static.
- Deep RNN increases computational complexity due to a greater number of parameters compared to an RNN. Deeper networks are more susceptible to vanishing of gradients.
- Bidirectional RNN needs to know both the start and end sequence and increases the computational complexity due to a larger number of parameters compared to a vanilla RNN.
- Multi-Dimensional RNN has a larger computational complexity when compared to RNNs. It significantly increases memory requirements for training and testing due to multiple recurrent connections (Salehinejad, Sankar, Barfett, Colak, Valaee).
- Hierarchical Subsampling RNN is sensitive to sequential distortions and also requires tuning window size.

Bianchini, M. Gori, M. Maggini, M. On the Problem of Local Minima in Recurrent Neural Networks. 1994. Retrieved from: https://dl.acm.org

Nawi, N. Khan, A. Rehman, M. Chiroma, H. Herawan, T. Weight Optimization in Recurrent Neural Networks with Hybrid Metaheuristic Cuckoo Search Techniques for Data Classification. Retrieved from: https://www.hindawi.com

Salehinejad, H. Sankar, S. Barfett, J. Colak, E. Valaee, S. Recent Advances in Recurrent Neural Networks. Retrieved from: https://arxiv.org