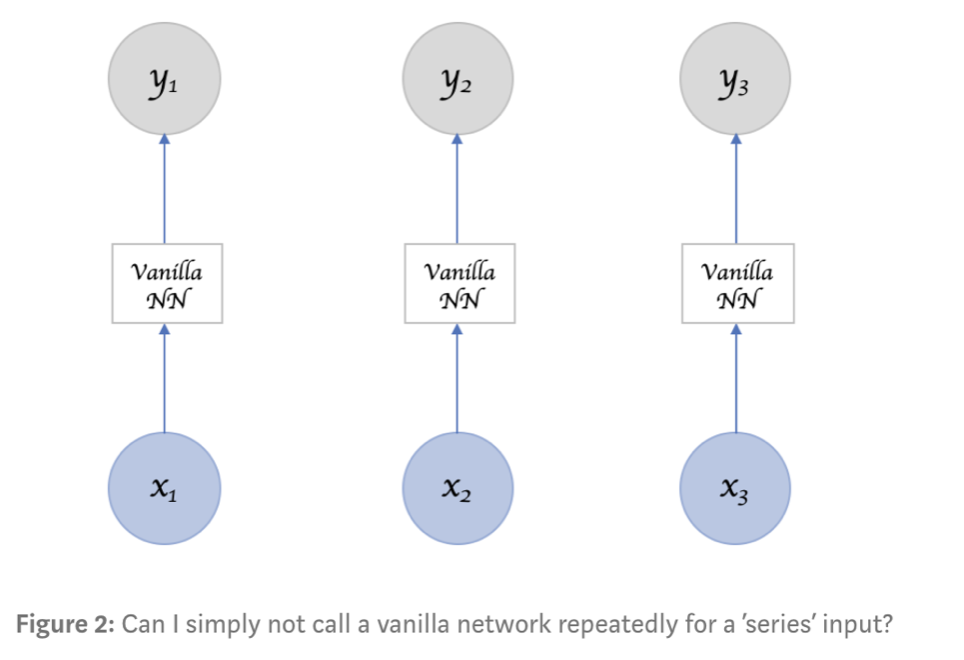
III. Theoretical & Conceptual Study of RNN

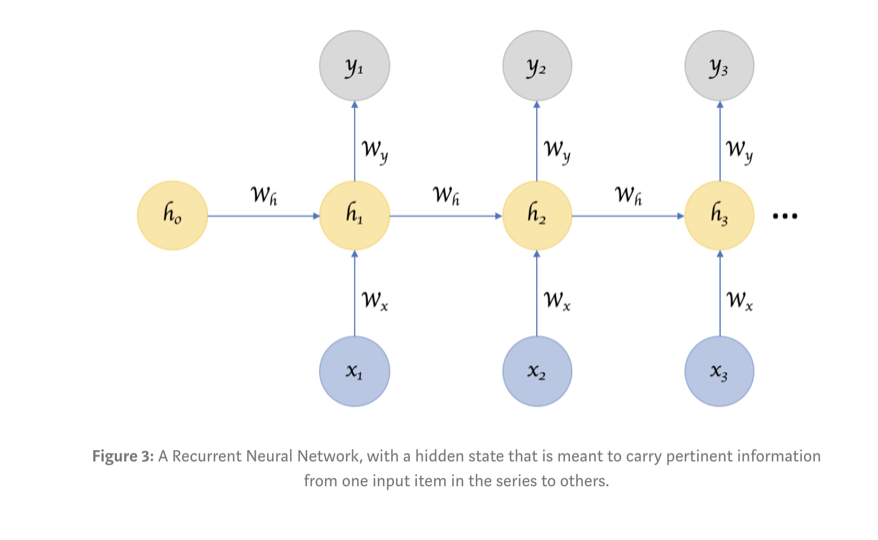
The algorithm we have chosen to implement is a Recurrent Neural Network for time series prediction. A generic time series model is based on the idea that future behavior can be predicted based on past behavior. In this case, we chose an RNN to learn how to predict future stock prices based on the stock prices in the past.

A common misconception is that if we take a singular Neural Network, and call it repeatedly, then this serves the same purpose as an RNN. This, however, is inaccurate, as we realized with further research, since the inputs and outputs in a regular NN are independent, as can be seen in the image below.



The inputs are independent in the sense that once one iteration has completed, calling the neural net a second time will simply repeat the same process again, with no influence from the previous call.

A key component of RNN, however, is that it can take in more than one input vector, and also output more than one output vector. This feature of the RNN is what allows the network to retain its “memory”, and make predictions not simply based off of what it has learned during training, but from the prior inputs and outputs as well, which can be seen in the image below.

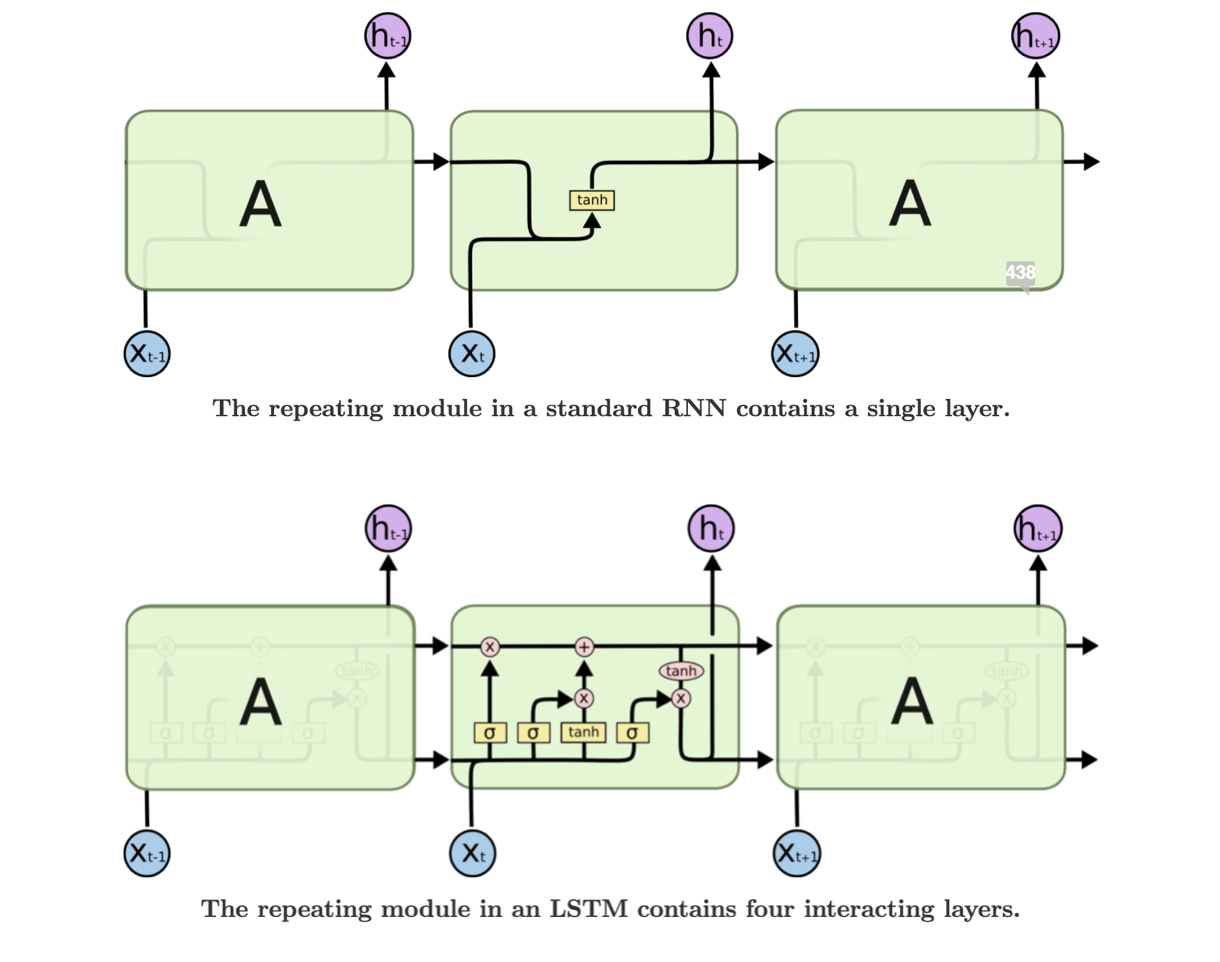


Therefore, in the case of an RNN, we would be able to input a data into the network, receive an output, and then have the network store that in its memory, so that if we were to input the same data again into the RNN, it would then use what it has previously learned to more accurately predict the future value.

While RNNs typically have far more advantages than disadvantages when it comes to running prediction models, we did find that RNNs are more suitable for running “short term” predictions. To run more data through a regular neural network, more layers can be added to create a deep neural net. With an RNN, on the other hand, since keeping track of the previous inputs in memory is necessary, if the prediction is simply based on more recently stored information, then the RNN will be capable, but if the information is stored further back in memory, then the network may struggle.

Through our research, we have chosen to go forward with the LSTM (Long Short Term Memory) model, which is a special type of RNN model that does allow for the network to learn based off of long term dependencies.

To gauge a better idea of the difference between a regular RNN and the LSTM model, we can take a look at the images below.



As we can see, whereas the RNN on the top only has one layer, the LSTM model on the bottom has 4 components working together : the forget gate, the input gate, the Cell itself, and the output gate.

The forget gate is the first layer in the model, and this gate determines which data elements to keep and pass on to the cell, and which elements to get rid of entirely. The Sigmoid function is used in this layer, and a 0 or 1 is output to indicate whether the data will be used or not. Data disregarded at this layer cannot be retrieved for future use.

The next part of the model is the input gate. The first part of this layer uses another Sigmoid function, which allows the relevant information which we will update to be passed on to the next part of the layer, which then uses the Tanh function to create the vector of values to be updated. This vector of values is then used to update the cell.

Finally, the last component of LSTM is the output layer. In this layer, the output from the cell is filtered to represent the necessary information in order to complete the desired prediction.