Train a Long Short Term Memory Neural Network (LSTM) with PyTorch on Bitcoin trading data

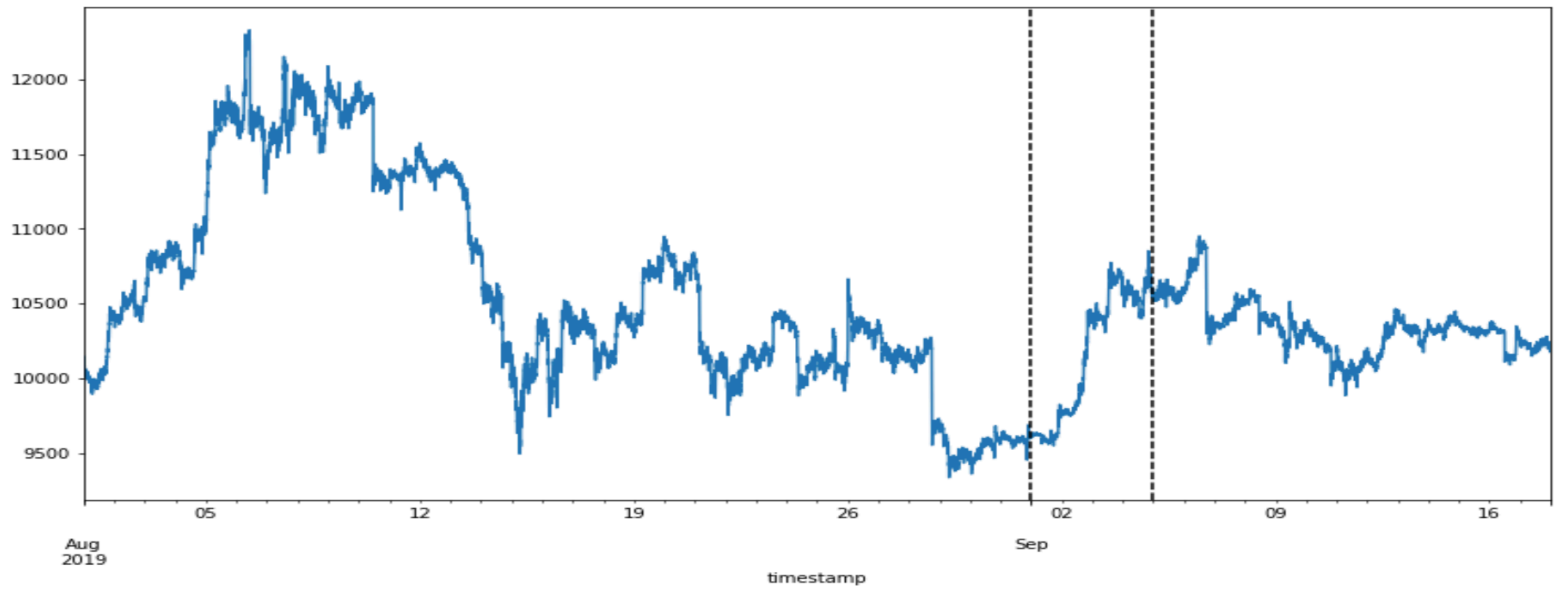
DATA

We analyze **the XBTUSD trading data** from BitMex.

Each row represents a trade:

- timestamp
- symbol of the contract traded,
- side of the trade, buy or sell,
- size represents the number of contracts (the number of USD traded),
- price of the contract,
- tickDirection: describes an increase/decrease in the price since the previous transaction,
- foreignNotional: is the amount of USD in the trade.
- We are going to use 3 columns: **timestamp, price, and foreignNotional**.

timestamp	symbol	side	size	price	tickDirection	foreignNotional
2019-08-01 00:00:03.817388	XBTUSD	Sell	849	10088.5	ZeroMinusTick	849.0
2019-08-01 00:00:03.817388	XBTUSD	Sell	1651	10088.5	ZeroMinusTick	1651.0
2019-08-01 00:00:03.950526	XBTUSD	Buy	34679	10089.0	PlusTick	34679.0
2019-08-01 00:00:03.950526	XBTUSD	Buy	35	10089.0	ZeroPlusTick	35.0
2019-08-01 00:00:03.950526	XBTUSD	Buy	35	10089.0	ZeroPlusTick	35.0



Time bars showing XBTUSD VWAP from 1st of August till the 17th of September 2019.

The plot shows time bars with Volume Weighted Average Price (VWAP) in 1 minute time intervals from 1st of August till the 17th of September 2019. We use the first part of the data for the training set, part in-between for validation set, and the last part of the data for the test set (vertical lines are delimiters).

DATA PREPROCESSING

- To help the LSTM model to converge faster, it is important to scale the data. It is possible

that large values in the inputs slow down the learning.

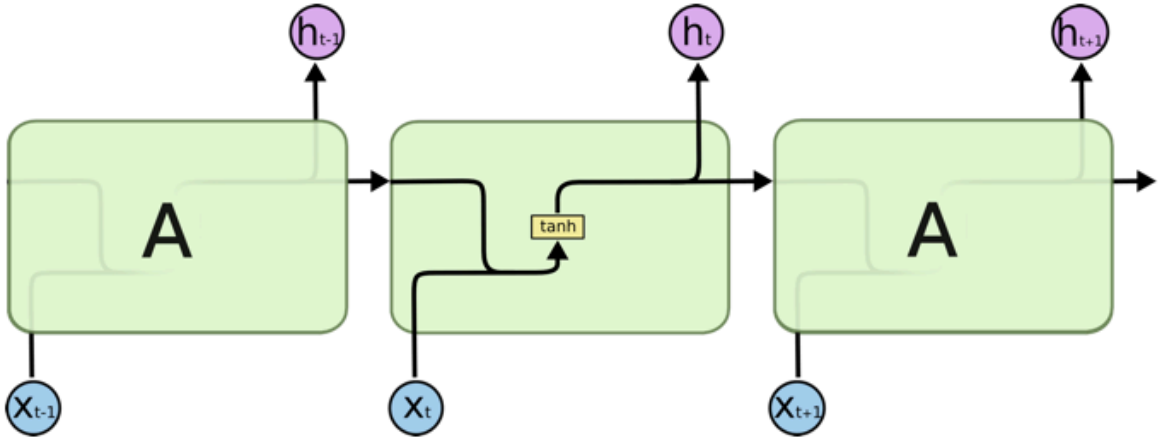
- After scaling, we need to transform the data into a format that is appropriate

for modeling with LSTM. We transform the long sequence of data into many shorter sequences (100 time bars per sequence) that are shifted by a single time bar.

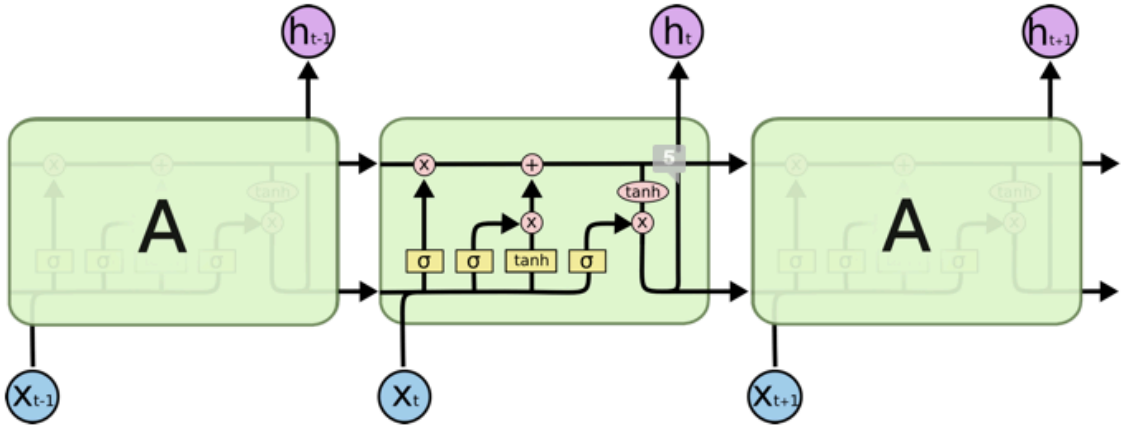
LSTM

- The Long Short Term Memory neural network is a type of a Recurrent Neural Network (RNN).
- RNNs use previous time events to inform the later ones. RNNs work well if the problem requires only recent information to perform the present task.
- If the problem requires long term dependencies, RNN will struggle to model it.
- The LSTM was designed to learn long term dependencies. It remembers the information for long periods.
- LSTM also prevent the problem of vanishing gradients as it provides a highway for the gradients to flow backwards and update the weights.

RNN vs LSTM



The repeating module in a standard RNN contains a single layer.

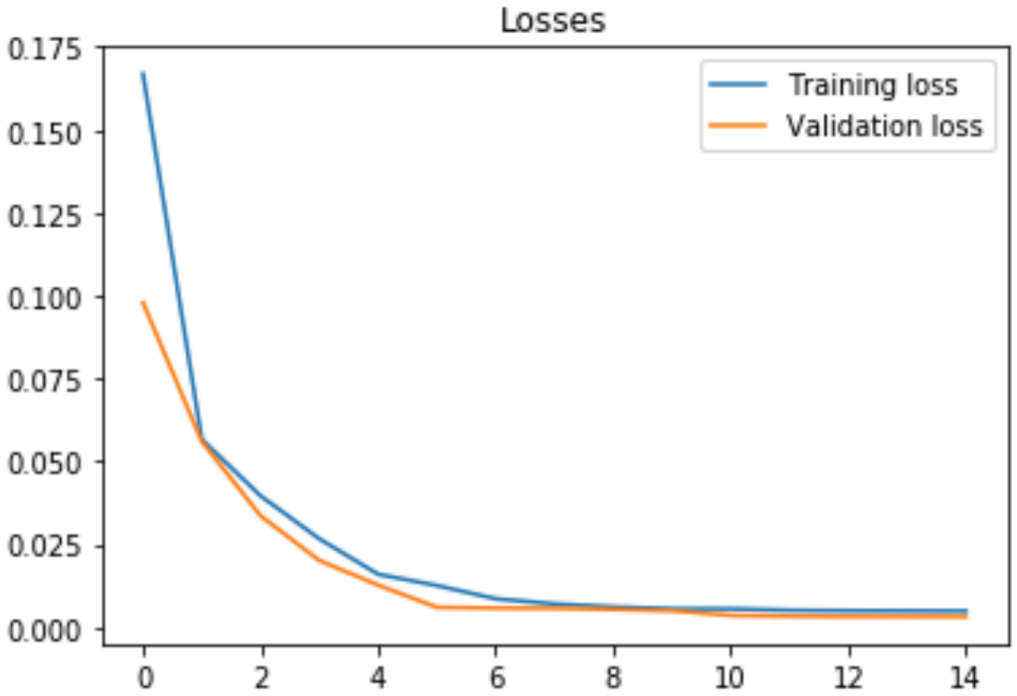


The repeating module in an LSTM contains four interacting layers.

Training the network

- The network has 21 hidden units.
- A low number of hidden units is used to prevent the network from overfitting.
- Loss function : Mean squared error
- Optimizer : Adam Optimizer
- Learning rate is set to 0.001, and it decays every 5 epochs.
- Train the model with 100 sequences per batch for 15 epochs.

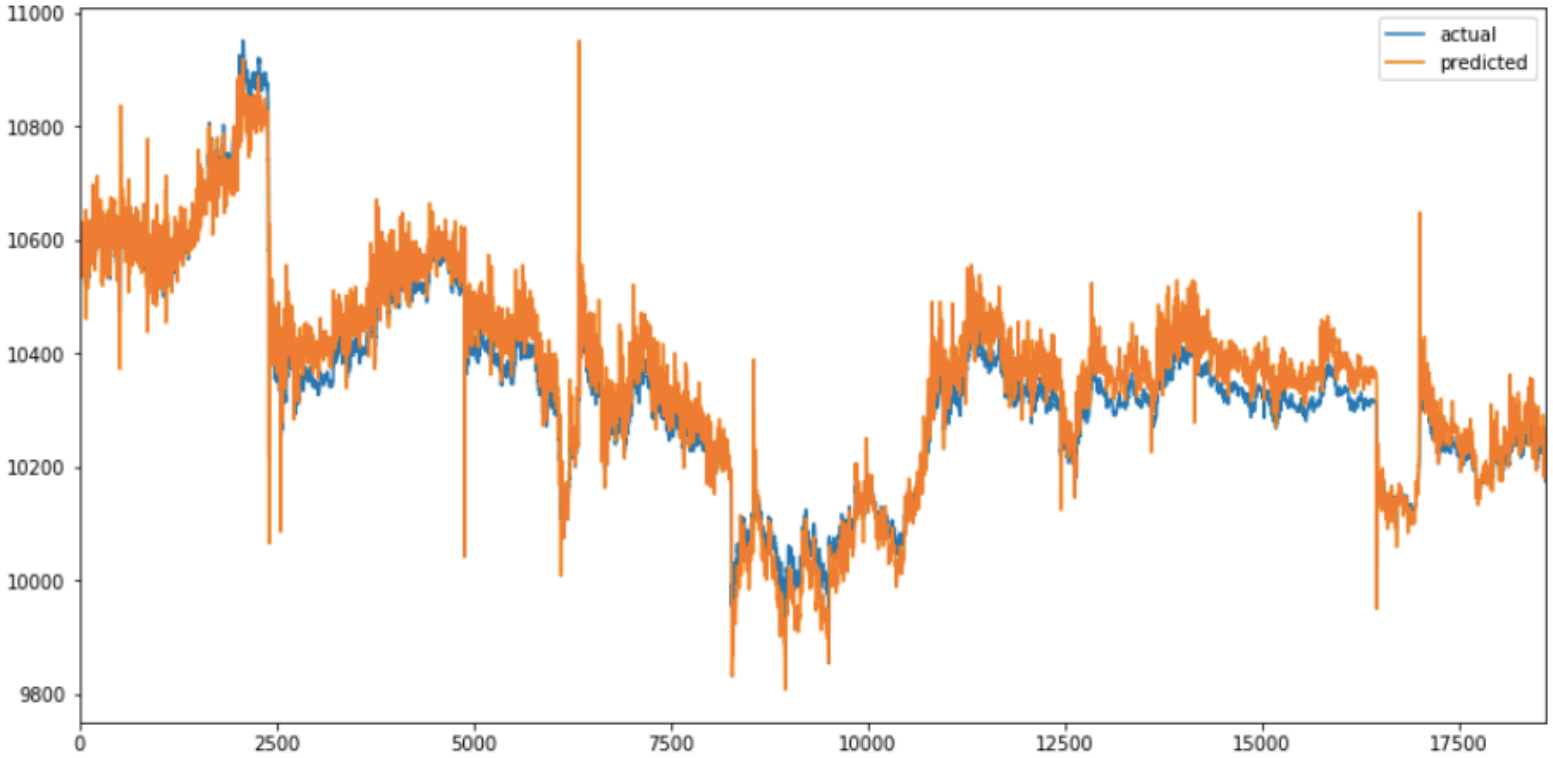
Training and Validation Loss



Training and validation loss.

In this plot we observe that training and validation loss converge after the sixth epoch.

Actual Test Data VS. Predicted Values



Actual and predicted VWAP on the test set. We can observe that predicted values closely match the actual values of VWAP