# BTC Deep Learning

## Inputs

Input = 100 x outputBars

## Outputs – 1

Price action will go up/down/(sideways)

(Buy if price will go up, Sell if price will go down, neither if indecisive.)

## Outputs – 2

Input = high, low, open, close, volume  
SEQ\_LEN = as large as possible  
Predict:

1. Price Peak within next *X* Bars
2. Price Bottom within next *X* Bars

NN Design:

Inputs may not need to be normalised, however attention layer required  
output layer has no activation, as is normal for regression (activation=’linear’)

SEQ2SEQ

Loss function:

* <https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/>
* MSE, mean squared error loss is most commonly used in this scenario. Potentially may overpunish for big differences when predicting large prices)
* MSLE, mean squared logarithmic error loss takes the natural log of predicted values then calculates MSE. Should be better for large values. Loss would likely grow for large target values.
* MAE, mean absolute error, better when there exists outliers (which there will be), doesn’t blow up for large values, however loss will be less when dealing with smaller values
* Considering the case: y1 = 100, yp1 = 200 BAD but y1=50000, yp1 = 50100 very good. I need an error which is relative to the value predicted. E.g. a percentage error.
* Quantile loss, useful when predicting an interval instead of point predictions. However, requires constant variance across values of independent variables. As price rises, variance increases.
* MAPE, mean absolute percentage error, PERFECT

Postprocessing:

* Calculate Diff1 (Peak – Current)
* Calculate Diff2 (Current – Bottom)

Spot version

* If (Diff1 >> Diff2) then Buy
* If (Diff2 >> Diff1) then Sell

Perpetual version

* If (Diff1 >> Diff2) then Buy set stop loss at Bottom x multiplier
* If (Diff2 >> Diff1) then Sell set stop loss at Bottom x multiplier

## Outputs – 3

Predict the next X bars % move. E.g. the next 2 days will be +5%.

# DNN notes

Can use feedback neurons, LSTMs. It keeps memory