International Journal of Engineering and Management Research
e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
www.ijemr.net
https://doi.org/10.31033/ijemr.10.1.19

Exploring the Impact of Magnitude- and Direction-based Loss Function on the Profitability using Predicted Prices from Deep Learning

Chihcheng Hsu[1] and Lichen Tai[2]
[1]Associate Professor, Department of Information Management, National Central University, TAIWAN
[2]Master, Department of Information Management, National Central University, TAIWAN

[1]Corresponding Author: khsu@mgt.ncu.edu.tw

ABSTRACT

Researches on predicting prices (as time series) from deep learning models usually use a magnitude-based error measurement (such as R^2). However, in trading, the error in the predicted direction could affect trading results much more than the magnitude error. Few works consider the impact of ill-predicted trading direction as part of the error measurement.

In this work, we first find parameter sets of LSTM and TCN models with low magnitude-based error measurement, and then calculate the profitability using program trading. Relationships between profitability and error measurements are analyzed.

We also propose a new loss function considering both directional and magnitude error for previous models for re-evaluation. Three commodities are tested: gold, soybean, and crude oil (from GLOBEX). Our findings are: with given parameter sets, if merchandise (gold and soybean) is of low averaged magnitude error, then its profitability is more stable. The proposed loss function can further improve profitability. If it is of larger magnitude error (crude oil), then its profitability is unstable, and the proposed loss function cannot improve nor stabilize the profitability.

Furthermore, the relationship between profitability and error measurement for models of LSTM and TCN with or without customized loss function is not, as commonly believed, highly positively correlated (i.e., the more precise the predicted value, the more trading profit) since the correlation coefficients are rarely higher than 0.5 in all our experiments. However, the customized loss functions perform better in TCN than in LSTM.

Keywords— Deep Learning, Price Prediction, Program Trading, Time Sequence, LSTM, TCN

I. INTRODUCTION

Predicting future values of time series is the topic of many previous works [10,14,16]. Using deep learning provides better progress than that of using linear models [9,14]. However, few works provide the guideline on applying the forecasted values in actual trading, as well as measuring the profitability concerning the error measurement (such as R^2 and MSE).

In actual trading, predicting the trading direction (i.e., to buy or to sell) incorrectly (called directional error in this paper) usually results in bigger loss than the magnitude error from the difference between the predicted and actual values. When the predicted value has a large magnitude error from the actual one, but with the correct trading direction, this may lead to profit (Fig.1.a). A smaller magnitude error with the incorrect directional prediction may lead to a larger loss (Fig. 1.b). We study how to incorporate directional error into error measurement for deep learning models in this paper.

(a) (b)

Figure 1: (a) Correct directional prediction with larger magnitude error; (b) incorrect directional prediction with smaller magnitude error

Different merchandise may need different deep learning models. LSTM is suitable for merchandise with lower volatility [5, 14], TCN may perform better on merchandises with higher volatility due to its pre-filtered local patterns [11]. Finding suitable deep learning models and parameter sets for studied merchandises is the first step to study their profitability.

Program trading can systematically and repeatedly explore the profitability of learning models. The utilized trading program should match the timing where learning models generate values to reduce unnecessary errors.

We explore the related literature in Section II. In Section III, a three-staged process is proposed to evaluate magnitude error for parameter settings of deep learning models. The specific trading program and the loss function incorporating both directional and magnitude error are also introduced in Section III. Section IV details the results of experiments with discussion. The conclusion comes in Section V.

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

II. RELATED LITERATURE

Time-series: Variables inherently ordered by time are called time series, such as stock or currency price. They are usually not expressible as linear functions since the volatility and fluctuation may change significantly along the time. With a good learning model and long enough data, a self-adaptive learning model may provide the right prediction [17]. Deep learning models is one of this kind.

Neural networks as learning models: Neural networks use neuron, layers of neurons, and adjustable weights to capture the non-linear relationship among input and output [3]. The neuron mimics the human brain to work as simple units, which are highly connected. Weights on the links may substantially affect the networks, and overfitting problem may occur quickly. The number of layers, the number of neurons in each layer, and parameters such as the dropout rate should be regulated to avoid overfitting problems.

Deep learning: Deep learning is a multi-layered neural network with sophisticated routing among inter-connected neurons and layers. In traditional data mining, features are usually manually specified in advance, and this is called feature engineering. For data with inherently complex features, feature engineering is complicated. Deep learning achieves automatic feature engineering through those layers of neurons [3]. However, the quality of deep learning is profoundly affected by its training process. For example, the gradient should be managed appropriately to avoid gradient explosion or gradient vanishing problems.

Deep learning for time series: With cross-layered connectivity, recurrent neural network (RNN) allows deep learning on time series [3]. By redirecting the output value back to the input, RNN acquires the "memory" effect. Uncontrolled utilization of memory may lead to more noise. LSTM (Long Short Term Memory) maintains the memory effect of RNN but only keeps important one for future usage. Pant in [14] successfully applies LSTM to predict the currency fluctuation of the US dollar and Russian Ruble.

CNN (Convolution Neural Networks) excels at image recognition but performs less robust in time series prediction. Bai et al. (2018) proposes TCN (Temporal Convolutional Networks) for time-series prediction and get better results than LSTM in many situations. TCN, similar to CNN, acquires signals simultaneously, where RNN acquires signals sequentially. TCN regulates the sliding windows for convolution to enable time series prediction. In this paper, TCN is also explored, in addition to LSTM.

Activation functions: The purpose of activation function in neural networks is to ensure its input and output are not linearly related, as a linear relationship between input and output reduces the expressiveness of the networks significantly. Commonly used non-linear functions are Sigmoid, tanh, and ReLu [16]. Sigmoid is commonly used for classification problems as it maps the input to value between 0 and 1. Tanh and ReLu are commonly used for non-classification problems, where ReLu speeds up the training without a gradient vanishing problem [8]. Hochreiter in [6] further provides another activation function, SeLu, for time series problems. SeLu can converge well, avoid gradient explosion or vanishing problems, and perform well in deeper layered networks. Thus, both SeLu and ReLu are tested in our TCN experiments. However, due to the limitation of Cudnn in our LSTM model, only Tanh can be used for the activation function in our LSTM testing. The definition of SeLu function is provided as below:

$$SeLu:\ SeLu(x) = \lambda \begin{cases} x & if\ x > 0 \\ \alpha e^x - \alpha & if\ x \le 0 \end{cases} \quad (1)$$

Loss function: The quality of regression is improved by minimizing the loss function. The loss function usually measures the absolute or squared distance between the actual and predicted values. The functions of MAE and MSE are expressed as follows:

$$MAE: \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \ (2) \quad MSE: \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \ (3)$$

MAE tends to find local optimum when the gradient increases. MSE could be affected by outliers where normalization can reduce the problem, thus, MSE is more popular than MAE [2].

Measuring the quality of deep learning models: MSE or coefficient of determination (R^2) are used in deep learning to measure the quality of the learning. R^2 is developed from MSE with variance as the denominator to ensure its values always range between 0 and 1 and allow comparison across different features. Its function is as follow:

$$R^2(y, \hat{y}) = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \quad (4)$$

When the number of samples increases, R^2 may lose accuracy as its numerator increases un-proportionally. Hyndman and Koehler in [7] proposed the use of MASE to reduce the problem. MASE uses the naïve projection of in-sample data as the denominator to reduce the errors from out-sample data and achieves better normalization. Its function is expressed as follows:

$$MASE = mean(|q_t|) \quad (5)$$

$$q_t = \frac{e_t}{\frac{1}{n-1}\sum_{i=2}^{n}|x_i - x_{i-1}|} \quad (6)$$

$$e_t = |y - \hat{y}| \quad (7)$$

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

Program trading: Program trading uses a set of fixed rules expressed as programs to buy and sell merchandise. It can apply to merchandise with historical price data repeatedly to ensure the provided rules for specific merchandise are indeed profitable (Keith, 2000). Multicharts [12] is used in this research due to its wide user base and easy readability.

III. SYSTEM DESIGN AND IMPLEMENTATION

Our proposed system design and experiments are divided into three stages and expressed in Fig. 2. The first stage focuses on finding models and settings from LSTM and TCN with error measured by R^2 and MASE. In the second stage, the predicted values are used in the trading program to study their profitability and relationship with error measurement (R^2 and MASE). The proposed customized loss function considering both directional and magnitude error is introduced in the third stage, where the works in the first two stages are repeated for similar models with this modification on loss function inserted. Results from these experiments are analyzed to understand the potential improvements.

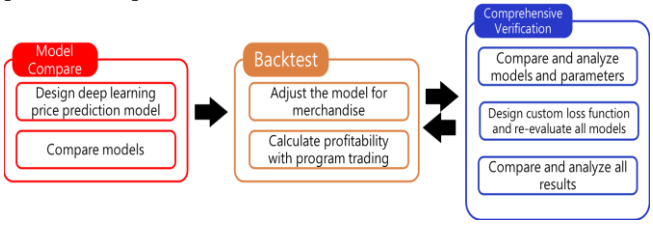

Figure 2: Three-staged system design and experiments

Use trading programs with the predicted next-day's close prices: Most works on deep learning models highlight their contribution by measuring the improvement on R^2 for the proposed models. Our research further pursues the trading profitability of predicted values as another possible measurement for the quality of learning models. The utilized trading model should match how the next-day close price is predicted. The trading model should be kept simple enough and not introduce other factors to dilute the effect of the predicted next-day close price from the learning models. As shown in Fig. 3, we propose to enter the market five minutes before the close of today and exit the position before the end of the next trading day. A buy position is entered when the predicted close price for the next day is proportionally higher than today's close price. To reduce unexpected risk, a percentage stop-loss is used to liquidate position if a certain percentage of loss relative to price is reached. The exact proportion to today's close price for entry and stop-loss is found by the optimization mechanism from program trading [15]. The short position is done with the exact opposite rules. If this simple trading program can profit, we can attribute most of the success to the precision of the predicted next-day close value.

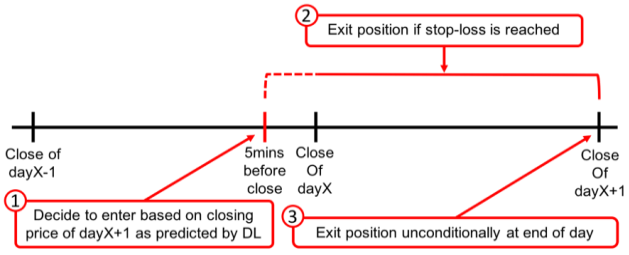

Figure 3: A trading model utilizes the predicted next-day close price at around today's close

In existing researches, today's close price is always used to predict the next-day close price [9, 14]. In reality, when today's close price is available for the prediction, the market is already closed for trading. This problem should be fixed when trading profitability is studied. We use the price five minutes before the close to predicting the next-day close price in the proposed trading program. To verify the feasibility of this modification, the fluctuation of the last five minutes before close for the three merchandise studied in our experiments are analyzed. The average price fluctuation for these merchandise in the last five minutes is less than 0.01% as shown in Table 1. We further use the actual close price and the price five minutes before the close as two different inputs for the same LSTM models. The goal is to know how different the predicted next-day values are from the actual next-day close. As shown in Table 2, the R^2 for these two different inputs are almost the same.

TABLE I
THE PRICE DIFFERENCE BETWEEN TODAY'S CLOSE AND FIVE MINUTES BEFORE THE CLOSE

	SOYBEAN	CRUDE OIL	GOLD
AVERAGE	0.075(0.007%)	0.002(0.003%)	-0.023(-0.002%)
MAXIMUM	6.0(0.544%)	0.2(0.267%)	6.0(0.444%)
MINIMUM	-3.75(-0.340%)	-0.19(-0.253%)	-11.5(-0.851%)
STANDARD DEVIATION	1.046	0.039	0.769

TABLE 2
R^2 OF PREDICTED AND ACTUAL VALUES, USING ACTUAL CLOSE OR PRICE FIVE MINUTES BEFORE CLOSING AS INPUTS

	SOYBEAN	CRUDE OIL	GOLD
CLOSE PRICE	0.945	0.972	0.967
5 MIN BEFORE CLOSE	0.945	0.976	0.963

The next-day exit trading program: The corresponding trading program, based on Fig. 3, in Multicharts' PowerLanguage is shown in Fig. 4. This next-

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

day exit trading program has parameters SL and SS as the percentage of stop-loss for buy and short. numSD specifies the multiple of the standard deviation to be exceeded for the difference between the current price and the predicted price when determining whether to initiate buy (or short) position. *Length* specifies the period to measure the standard deviation. In the trading program, D2price is the predicted price from deep learning models and is input as the second data source (called "data2" in Multicharts). S is the standard deviation, ok2buynextbar and ok2sellshortnextbar are the Boolean variables track whether buy or short decision is activated.

Line 7 of Fig .4 determines whether it is five minutes before the close, Line 9 decides whether to buy or short based on how far D2Price deviates from the current price, Line 16 restricts the number of trades to once a day, Line 24 executes the possible stop-loss, and finally Line 31 forces exit at the end of day.

Optimization in program trading is also conducted to seek the maximum potential profit as the profitability deep learning models (with specific parameters) can achieve.

```
1  input:sl(2),ss(0),numSD(0),Length(10);
2  variables:D2price(0),S(0),ok2buynextbar(0),ok2sellshortnext
3  bar(0);
4  D2price = close of data2;
5  S = StandardDev(close, Length, 1); //S
6  value1 = numSD * S; //where to buy
7  if time=(sess1endtime - barinterval) then begin
8          ok2buynextbar = 0; ok2sellshortnextbar = 0;
9          if D2price > (close + value1) then begin
10                 ok2buynextbar = 1;
11         end;
12         if D2price < (close - value1) then begin
13                 ok2sellshortnextbar = 1;
14         end;
15 end;
16 if EntriesToday(date) < 1 then begin
17         if ok2buynextbar = 1 then begin
18                 buy next bar at market;
19         end;
20         if ok2sellshortnextbar = 1 then begin
21                 sellshort next bar at market;
22         end;
23 end;
24 if marketposition>0 then begin
25         sell next bar at entryprice*(1-(sl*0.01)) stop;
26 end;
27 if marketposition<0 then begin
28         buytocover next bar at entryprice*(1+(ss*0.01))
29 stop;
30 end;
31 setexitonclose;
```

Figure 4: The code for next-day exit trading program

Data collection of the three traded merchandise: Based on [14] and the data collected, three commonly traded commodities, gold, soybean, and crude oil are used for the experiments in this paper with their data specification provided in Table 3.

TABLE 3
DATA SPECIFICATION OF THREE MERCHANDISES

	Soybean	Crude oil	Gold
Trading days	Monday - Friday	Monday - Friday	Monday – Friday
Trading hours	19:00 of prior day ~ 13:20	17:00 of prior day~16:00	18:00 of prior day~17:00
Big point value	50 USD	1000 USD	100 USD
Data source	GLOBEX soybean Futures provided by e-signal	GLOBEX crude-oil Futures provided by e-signal	GLOBEX gold Futures provided by e-signal
Period	2006/01/01 ~ 2018/04/30	2007/03/01 ~ 2018/04/30	2007/03/01 ~ 2018/04/30

As explained, both the daily close price and the price of five minutes before daily close are input for our deep learning models. All the data collected for the trading program is one-minute price data since the program in Fig. 4 needs to be executed on one-minute data for correctness. Eighty percent of data is used as a training set and twenty percent as a test set.

Parameter settings for LSTM and TCN models: The CudnnLSTM in Cudnn library of GPU [13] is used for the LSTM model in this work where the ranges of parameter setting are shown in Table 4. Those for the TCN model are shown in Table 5.

TABLE 4
PARAMETER SETTING FOR LSTM

HIDDEN LAYER	1					2				
# OF NEURON	3	6	7	50	80	3	6	7	50	80

TABLE 5
PARAMETER SETTING FOR TCN

Activation functions	ReLu, SeLu
Filters	32, 64, 128, 256, 512, 1024
Dilations	[1,2,4,8], [1,2,4,8,16]. [1,2,4,8,16,32], [1,2,4,8,16,32,64], [1,2,4,8,16,32,64,128], [1,2,4,8,16,32,64,128,256], [1,2,4,8,16,32,64,128,256,512]
Dropout_rate	0, 0.2, 0.5, 0.7

Measuring errors for the prediction models: The coefficient of determination (R^2) from the scikit-learn library where R^2=r2_score(y_test,y_(test_pred)) is directly used in our study. The MASE(train,test,pred) from [4] is also used in our study and calculated based on the following steps: (1) calculating the length n of train data set, (2) using the diff function from numpy to mimic naïve prediction

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

(calculate continuous residuals), (3) summing up the mimicked naïve prediction, taking absolute value, and dividing by n-1 to get the denominator of MASE, (4) similarly calculating the residuals among test and pred, taking absolute value, using mean to sum and average, and finally dividing by the denominator as acquired in (3) to get MASE.

Customized loss function: The built-in loss function in Keras can only reference y_true and y_pred (Keras, 2018), and cannot be used to accommodate our required directional error. However, the loss function in Keras can be replaced as long as the new one is of the Backend format from either TensorFlow or Theano.

Our customized loss function is modified from MSE. The design principle of the proposed loss function is to (a) maintain the original value at best, (b) reduce the original penalty when the predicted direction is the same as the actual, and (c) increate penalty when the predicted is different from the actual. The exact procedure (as shown in Fig. 5) includes (1) using tf.manip.roll function to copy the Tensor for both actual values (y_true) and predicted values (y_pred), and performing roll function to get both the actual value as rolled by one day (y_true_1) and the predicted value as rolled by one day (y_pred_1) as the two required prices for the previous day, (2) getting the Tensor difference between (y_true, y_pred) and (y_true_1, y_pred_1), (3) using the difference in (2) to know whether the actual direction (trueD = y_true – y_true_1) and predicated directions (predD = y_pred – y_pred_1) is the same or not, (4) multiplying *trueD* with *predD*, which will be negative if the two directions are different and positive if the two directions is the same, (5) and finally increasing or reducing penalty based on (4).

The 0.01 in the last line of the loss function is the adjusting factor for the penalty where 1, 0.1, 0.01, and 0.001 are tested initially. Values of 0.001 and 1 are discarded as too little or too heavy the penalty is introduced. 0.1 is also omitted as the process of deep learning cannot converge, and 0.01 is the final choice as shown here. This adjusting factor only affects the speed of converging and using a multiple of 10 is sufficient.

```
def New_Loss_Func(y_true,y_pred):
    y_true_1 = tf.manip.roll(y_true, shift=1, axis=0)
    y_pred_1 = tf.manip.roll(y_pred, shift=1, axis=0)
    d = tf.sign((y_true - y_true_1) * (y_pred - y_pred_1))
    #取符號(y = sign(x) = -1 if x < 0; 0 if x == 0; 1 if x > 0.)
    return K.mean(K.square((y_pred - y_true) - (0.01*d)), axis=-1)
```

Figure 5: Customized loss function

IV. EXPERIMENTS AND DISCUSSION

Using the parameter sets for LSTM in Section III and the error measurement of R^2 and MASE, the results of three merchandises in Table 3 are shown in Table 6 and Fig. 6. Eighty percent of data is used as a training set, and twenty percent as a test set. All three merchandises show high R^2 in the training set but R^2 drop for some parameter settings in their test sets. MASE and R^2 reveal similar trends. For Fig. 6 (only results from test sets are plotted), more points at the right lower corner are better for merchandise since for R^2 the more close to 1 the better, and for MASE, the more close to 0 the better. Models for both gold and soybean reveal a excellent learning effect in LSTM, but not the case for crude oil.

TABLE 6
COMPARING RESULTS OF LSTM

Hidden layer		1	1	1	1	1	2	2	2	2	2
neuron		3	6	7	50	80	3	6	7	50	80
Gold	train	0.994	0.994	0.994	0.994	0.991	0.993	0.993	0.992	0.989	0.988
	test	0.965	0.967	0.966	0.968	0.945	0.963	0.965	0.965	0.94	0.94
	MASE	0.764	0.746	0.759	0.733	1.028	0.802	0.775	0.758	1.087	1.085
Soybean	train	0.975	0.981	0.977	0.977	0.981	0.97	0.978	0.978	0.981	0.982
	test	0.911	0.921	0.946	0.947	0.946	0.784	0.886	0.87	0.945	0.951
	MASE	0.68	0.647	0.497	0.492	0.491	1.196	0.822	0.887	0.499	0.462
Crude Oil	train	0.992	0.992	0.993	0.993	0.993	0.99	0.992	0.993	0.992	0.991
	test	0.903	0.955	0.974	0.97	0.976	0.858	0.907	0.956	0.978	0.98
	MASE	1.614	1.224	0.921	0.954	0.873	1.963	1.573	1.102	0.874	0.849

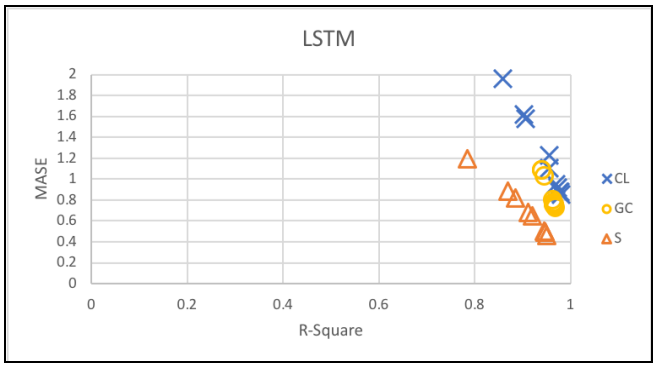

Figure 6: Error measurement for the test set from LSTM

Parameter sets for the TCN models in Section III are also applied to three merchandises. TCN's parameter setting includes activation function, filters, dilations, and dropout rate. They are evaluated individually.

SeLu and ReLu, as activation functions, are compared, as shown in Table 7. All three merchandises show high R^2 in the training set. SeLu performs better than ReLu in test sets for both soybean and crude oil, where both SeLu and ReLu perform similarly in gold. The same trend is also observed in MASE. Here the setting for Filter is 256, [32] for Dilations, and 0.2 for the dropout rate. These

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

values for Filter, Dilations, and dropout_rate are fixed as these values show better results when being tested individually. A similar approach is used when comparing the error measurement for different parameter settings. The whole purpose here is to know which parameter setting in TCN generates models of lower error measurement on the three merchandise.

Filters for TCN are compared, as shown in Table 7. For test sets, gold has good results on all Filter parameters, Crude oil performs well at around 128 to 512, and soybean performs well at most values except 1024. The same trend is also observed in MASE. Here the setting for other parameters is: SeLu, Dilations:[32], and Dropour_rate:0.2.

Dilations for TCN are compared, as shown in Table 8. For test sets, gold has good results on all Dilations parameters, crude oil performs well at 16, 32, and 256, and soybean performs well at 8, 16, and 32. The same trend is also observed in MASE. Here the setting for other parameters is SeLu, Filters: 256, and Dropour_rate:0.2.

Dropout_rate for TCN is compared as shown in Table 9. For test sets, gold has good results at 0.2 and 0.5, and crude oil and soybean both perform well at 0.2. The same trend is also observed in MASE. Here the setting for other parameters is SeLu, Filters: 256, and Dilations:[32].

All the above parameter settings are applied to the trading program to calculate its profitability to analyze the relationship between traditional error measurement (R^2 and MASE) and profitability.

TABLE 7
COMPARING RESULTS FOR ACTIVATION FUNCTIONS OF TCN

Activation function		SeLu	ReLu
Gold	train	0.988	0.985
	test	0.966	0.963
	MASE	0.753	0.797
Soybean	train	0.975	0.975
	test	0.94	0.909
	MASE	0.526	0.652
Crude Oil	train	0.992	0.979
	test	0.974	0.484
	MASE	0.95	4.132

TABLE 8
COMPARING RESULTS FOR PARAMETER SETS OF FILTERS OF TCN

Filters		32	64	128	256	512	1024
Gold	train	0.993	0.991	0.988	0.988	0.984	0.985
	test	0.969	0.969	0.968	0.966	0.962	0.956
	MASE	0.704	0.705	0.731	0.753	0.806	0.887
Soybean	train	0.953	0.968	0.973	0.975	0.975	0.969
	test	0.924	0.919	0.925	0.94	0.924	0.878
	MASE	0.638	0.631	0.628	0.526	0.614	0.834
Crude Oil	train	0.988	0.988	0.991	0.992	0.992	0.987
	test	0.681	0.852	0.928	0.974	0.947	0.742
	MASE	2.426	2.132	1.597	0.95	1.467	2.933

TABLE 9
COMPARING RESULTS FOR PARAMETER SETS OF DILATIONS OF TCN

Dilations		8	16	32	64	128	256	512
Gold	train	0.99	0.988	0.988	0.986	0.985	0.978	0.976
	test	0.968	0.966	0.966	0.966	0.963	0.957	0.953
	MASE	0.724	0.749	0.753	0.744	0.796	0.87	0.925
Soybean	train	0.976	0.974	0.975	0.966	0.96	0.954	0.94
	test	0.949	0.921	0.94	0.91	0.928	0.885	0.897
	MASE	0.48	0.595	0.526	0.721	0.612	0.846	0.771
Crude Oil	train	0.992	0.992	0.992	0.992	0.992	0.992	0.989
	test	0.94	0.976	0.974	0.895	0.952	0.971	0.896
	MASE	1.46	0.906	0.95	1.567	1.156	0.988	1.973

TABLE 10
COMPARING RESULTS FOR PARAMETER SETS OF DROPOUT_RATE OF TCN

Dropout_rate		0	0.2	0.5	0.7
Gold	train	0.981	0.988	0.99	0.991
	test	0.958	0.966	0.967	0.946
	MASE	0.857	0.753	0.729	0.933
Soybean	train	0.945	0.975	0.929	0.641
	test	0.834	0.94	0.639	-0.224
	MASE	1.04	0.526	1.562	3.124
Crude Oil	train	0.986	0.992	0.991	0.987
	test	0.796	0.974	0.893	0.773
	MASE	3.065	0.95	1.492	3.247

Analyzing the Profitability of Predicted Prices and Error Measurement

From the previous results, R^2 and MASE demonstrates the same trend on almost all parameter sets. Thus, for simplicity, we only use R^2 as the error measurement when studying the relationship between profitability and traditional error measurement.

To measure the profitability of deep learning models, we use two years' out-sample data with the trading program in Fig. 4 for the initial capital of 100,000.0 US dollars to calculate the profitability of all parameter sets from LSTM and TCN models as shown in Tables 4 and 5.

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

The average and maximum profitability for three merchandize in all models and parameter sets are shown on the left side of Table 11.

Comparing to LSTM, TCN provides higher profitability for gold. The standard deviation for the profitability and the averaged maximum drawdown (MDD) for TCN are both lower (and better) than those from LSTM. The relationship between profitability and error measurement of the learning models of gold is provided in Fig. 7. The (orange-colored) circle represents LSTM, and the (blue) triangle is for TCN. More dots at upper right corners indicate their models reveal a higher positive correlation between profitability and error measurement (i.e., lower error leads to higher profits). For gold, this correlation coefficient is -0.55 for LSTM and 0.25 for TCN. Going down to specific parameter settings, When (hidden layers, # of neuron) =(1, 80), (2, 50), (2, 80), the LSTM model gets the highest profit. The highest one for TCN is (SeLu, 256, [1,2,4,8,16,32], 0.2).

Soybean's TCN models also provide higher profitability than LSTM's. The standard deviation for the profitability from TCN is also lower than that of LSTM's. The relationship between profitability and error measurement of corresponding models of soybean is shown in Fig. 8. For soybean, the correlation coefficient is -0.24 for LSTM and 0.2 for TCN. Going down to specific parameter settings. When (hidden layers, # of neuron) = (2, 80), the LSTM model gets the highest profit. The highest one for TCN is also (SeLu, 256, [1,2,4,8,16,32], 0.2).

The profitability patterns of crude oil from LSTM models are quite unstable. TCN models of crude oil provide higher averaged profitability with lower standard deviation for better stability and lower MDD for low exposed risk. The relationship between profitability and error measurement of corresponding models of crude oil is shown in Fig 9. For crude oil, the correlation coefficient is -0.82 for LSTM and 0.01 for TCN. Going down to specific parameter settings. When (hidden layers, # of neuron) = (2, 3) (2,6), the LSTM model gets the highest profit. The highest one for TCN is also (SeLu, 256, [1,2,4,8,16,32], 0.2).

From these analysis, we have the following findings: (1) TCN models provide positive correlation between error measurement (R^2) and profitability, (2) parameter sets from TCN provide much stable profitability than LSTM, especially the parameter set (SeLu, 256, [1,2,4,8,16,32], 0.2) of TCN get highest profitability in all three merchandise, (3) profitability fluctuates greatly in crude oil as well as its error measurement from both LSTM and TCN models. The correlation between profitability and error measurement for crude oil is also the most unstable among the three tested merchandise.

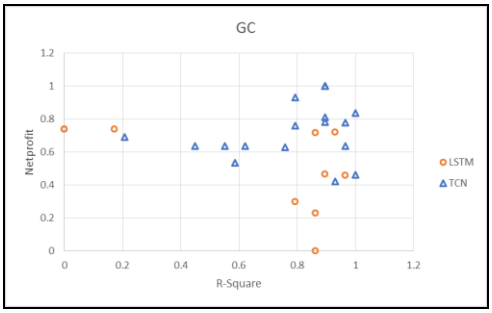

Figure 7: Relationship of profitability and error measurement on gold

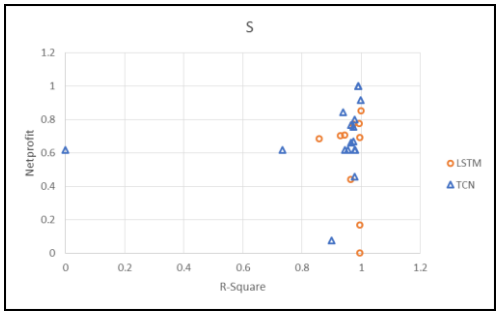

Figure 8: Relationship of profitability and error measurement on soybean

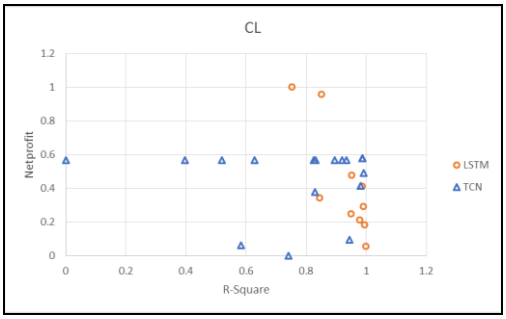

Figure 9: Relationship of profitability and error measurement on crude oil

Adding Customized Loss Functions for Re-Evaluation and Comparison

Granger (1999) suggested that the learning model and loss function determine the performance of machine learning. We have tested all reasonable parameter sets of LSTM and TCN models, and we now proceed to the experiments of adjusting loss function as described in Section III.

After creating the loss function (with directional error considered) and combining it into the similar Keras programs as in Section III, we repeat the same experiments and analysis from Section IV for similar models with the new loss function inserted. Due to the size limit for this paper, results similar to Table 6 from the re-evaluations are omitted, but their resulting diagrams (also for test sets only)

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

are shown in Fig. 10. For test sets, the error measurement for crude oil degrades much further after applying the loss function, and those for soybean and gold improve slightly. Though not shown here, the training sets for three merchandise still all have similarly high R^2.

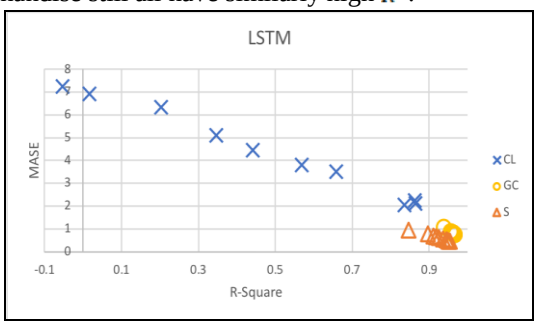

Figure 10: The relationship among two traditional error measurements, MASE and R^2, for LSTM models with the customized loss function

In TCN with the customized loss function, parameter sets for activation functions, filters, dilations, and dropout rates are all tested separately, and data similar to Tables 7 to 10 are collected. Due to space limitations, these detailed data are omitted with only their summary reported here. For activation functions, training sets for three merchandises are all with high R^2. In test sets, ReLu and SeLu both perform reasonably for gold and soybean, and SeLu performs better than ReLu in crude oil. Both R^2 and MASE reveal the same trend. Here the setting for other parameters is: Filters: 256, Dilations:[32], and Dropour_rate:0.2.

For TCN's Filters, in test sets, gold has good results on all parameters, crude oil performs well around 128 to 512, and soybean performs well in most values at 32, 512, 1024 with little difference. The same trend is observed in MASE. Here the setting for other parameters is: SeLu, Dilations:[32], and Dropour_rate:0.2.

For TCN's Dilations, in test sets, gold and crude oil both reveal good results on all parameters, and soybean performs well in values from 8 to 128. The same trend is observed in MASE. Here the setting for other parameters is SeLu, Filters:256, and Dropour_rate:0.2.

For TCN's Dropout_rate, in test sets, gold and soybean both reveal good results at 0 and 0.2, and crude oil performs well at 0.5. The same trend is observed in MASE. Here the setting for other parameters is SeLu, Filters:256, and Dilations:[32].

Comparing the error measurement for TCN with and without customized loss function, only crude oil displays worse performance. Gold and soybean perform slightly better. This finding is the same as what found from LSTM and matches the initial observation that the error measurement for crude oil is worst among the three merchandises for both LSTM and TCN.

Comparing Profitability for Those with Loss Functions and All Others

The average and maximum profitability for three merchandize in all models and parameter sets with the customized loss function are shown in the right side of Table 11.

Comparing to LSTM, TCN models for gold again provide higher profitability with lower standard deviation for better stability and lower MDD for low exposed risk. The relationship between profitability and error measurement for models of gold with the customized loss function is provided in Fig. 11. For gold, the correlation coefficient is -0.28 for LSTM and 0.06 for TCN. The profitability does increase after the customized functions are added for almost all parameter sets of TCN and LSTM. Furthermore, most points from both LSTM and TCN appear at upper right corners (i.e., high profitability with low error measurement), which indicates the customized loss function performs stably for both LSTM and TCN models on gold.

Similarly, TCN models for soybean provide higher profitability with lower standard deviation for better stability than those from LSTM. The relationship between profitability and error measurement for models of soybean is shown in Fig. 12. For soybean, the correlation coefficient is -0.003 for LSTM and -0.07 for TCN. With customized loss function added, the profit decreases for LSTM models but increases in the TCN model. Most points from TCN appear at upper right corners, but not the case for those of LSTM. This indicates the customized loss function performs stably for TCN but not for LSTM on soybean.

TCN models for crude oil also provide higher profitability with lower standard deviation for better stability than those from LSTM. The relationship between profitability and error measurement for models of crude oil is shown in Fig. 13. For crude oil, the correlation coefficient is -0.55 for LSTM and -0.33 for TCN. With the customized loss function added, the profit decreases for LSTM models, but increases in the TCN model. Almost all points from TCN appear at upper right corners, but not the case for those of LSTM. This indicates the customized loss function performs stably for TCN but not for LSTM in crude oil.

From results of all experiments in Section IV, we can summarize as follows: (1) TCN with customized functions perform best across all merchandise in terms of profitability and error measurement when either compared to the cases for LSTM or the cases for TCN without loss functions, (2) in LSTM, models with customized loss function only improve profitability on gold, but not that for soybean nor that for crude oil, (3) for a merchandise, if the error measurement for either LSTM or TCN model is low (gold and soybean), the profitability is usually good, and the loss function could improve profitability. On the

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

contrary, if the error measurement for either LSTM or TCN model is high (crude oil), the profitability usually fluctuate considerably, and the loss function cannot help, (4) the relationship between profitability and error measurement for models of LSTM and TCN with or without customized loss function is not, as commonly believed, highly positively correlated (i.e., the more precise the predicted value, the more trading profit) since the correlation coefficients are rarely higher than 0.5 in all our experiments, and (5) the customized loss functions perform better in TCN than in LSTM.

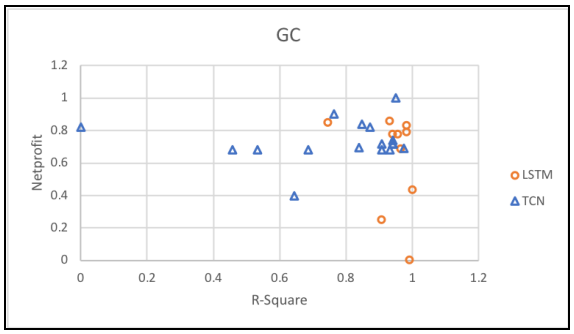

Figure 11: Profitability and error measurement for models with customized loss function from gold

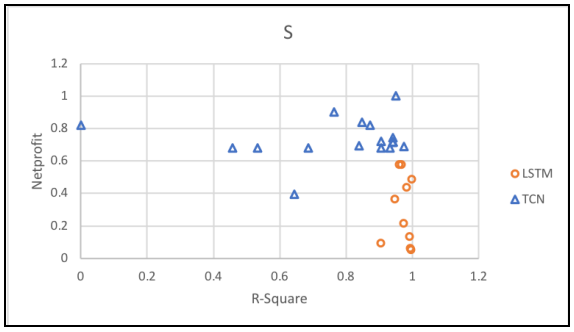

Figure 12: Profitability and error measurement for models with customized loss function from soybean

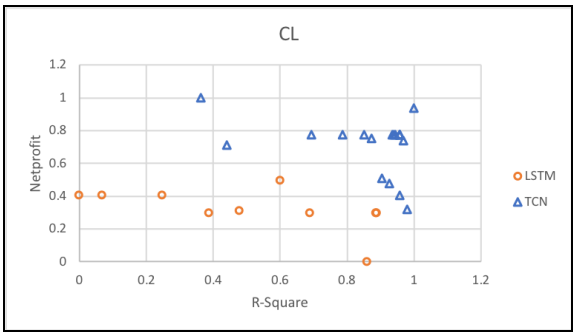

Figure 13: Profitability and error measurement for models with customized loss function from crude oil

TABLE 11
PERFORMANCE SUMMARY OF ALL MODELS FOR THREE MERCHANDISES

		LSTM	TCN	LSTM + customized loss fn.	TCN+ customized loss fn.
Gold	Max. profit	15535	**20990**	18412	**21308**
	Avg. profit	10749	**15672**	13714	**15876**
	Standard dev.	5550	3835	5989	2422
	Avg. MDD	12217	10540	10343	9475
Soybean	Max. profit	11925	**13775**	9900	**15763**
	Avg. profit	8485	**10255**	6039	**10948**
	Standard dev.	3578	2854	2959	3867
	Avg. MDD	5561	6446	5609	5668
Crude Oil	Max. profit	**34897**	20896	12308	**27149**
	Avg. profit	15596	**17195**	7109	**18781**
	Standard dev.	10568	6410	3903	5025
	Avg. MDD	21435	12716	8572	13124

V. CONCLUSION

In this work, a new validation approach is proposed for evaluating the quality of deep learning models when applied to merchandise price data (as time series). The proposed approach explores the relationships between profitability and error measurements of deep learning models. The profitability of learning models is measured by techniques from program trading with a trading program matching the deep learning prediction model (a next-day exit trading program in this paper). Besides, a new (customized) loss function considering both magnitude-based and direction-based error is introduced to improve the trading profitability of deep learning models.

A three-stage process of validation, as proposed in Section III, is conducted on three commonly traded merchandise in this paper, with all experiment results displayed and analyzed in Section IV. Our findings can be summarized as follows: (1) TCN with customized functions perform best across all merchandise in terms of profitability and error measurement when either compared to the cases for LSTM or the cases of TCN without loss functions, (2) in LSTM, models with customized loss function only improves profitability on gold, but not that for soybean nor that for crude oil, (3) for a merchandise, if the error measurement for either LSTM or TCN model is low (gold and soybean), the profitability is usually excellent and stable, and the proposed loss function could improve profitability. On the contrary, if the error measurement for either LSTM or TCN model is high (crude oil), the profitability usually fluctuate considerably, and the loss function cannot help, (4) the relationship between

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-10, Issue-1 (February 2020)
https://doi.org/10.31033/ijemr.10.1.19

profitability and error measurement for models of LSTM and TCN with or without customized loss function is not, as commonly believed, highly positively correlated (i.e., the more precise the predicted value, the more trading profit) since the correlation coefficients are rarely higher than 0.5 in all our experiments, and (5) the customized loss functions perform better in TCN than in LSTM.

REFERENCES

[1] Bai, S., Kolter, J.Z., & Koltun, V. (2018). *An empirical evaluation of generic convolutional and recurrent networks for sequence modeling*. Available at: https://arxiv.org/pdf/1803.01271.pdf.

[2] Brownlee, J. (2019). *Better deep learning*. Australia: Machine learning Mastery.

[3] Chollet, F. (2017). *Deep learning with Python*. USA: Manning Publications.

[4] Davidson-Pilon, C. (2013). *Computes the MEAN-ABSOLUTE SCALED ERROR forecast error for univariate time series prediction*. Available at: https://github.com/CamDavidsonPilon/Python-Numerics/blob/master/TimeSeries/MASE.py. Accessed on 18 January 2019.

[5] Greff, K., Srivastava, R. K., Koutnik, J., Steunebring, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232.

[6] Hochreiter, S., Klambauer, G., Unterthiner, T., & Mayr, A. (2017). Self-normalizing neural networks. *Proceedings of the NIPS 2017, Advances in Neural Information Processing Systems 30*. Availabel at: https://papers.nips.cc/paper/6698-self-normalizing-neural-networks.pdf.

[7] Hyndman, R.J., Koehler, A.B. (2006). Another look at measures of forecast accuracy. *Proceedings of the International Journal of Forecasting*, *22*(4), 679-688. https://doi.org/10.1016/j.ijforecast.2006.03.001.

[8] Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). ImageNet classification with deep convolutional neural network. *NIPS2012: Proceedings of the 25th International Conference on Neural Information Processing Systems*, *1*, 1097-1105.

[9] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436-444.

[10] Leinweber, D.J. (2007). Stupid data miner tricks: Overfitting the S&P 500. *The Journal of Investing*, *16*(1), 15-22.

[11] Li, H., Shen, Y., & Zhu, Y. (2018). Stock price prediction using attention-based multi-input LSTM. *Proceedings of the 10th Asian Conference on Machine Learning (PMLR 95)*, pp. 454-469.

[12] Multicharts. (2019). *MultiCharts12*. Available at: https://www.multicharts.com/. Accessed on 18 June 2019.

[13] NVDIA. (2018). *cuDNN developer guide*. Available at: https://docs.nvidia.com/deeplearning/sdk/cudnn-developer-guide/index.html. Aaccessed on 2 October 2019.

[14] Pant, N. (2017). *A guide for time series prediction using recurrent neural networks(LSTMS)*. Available at: https://blog.statsbot.co/time-series-prediction-using-recurrent-neural-networks-lstms-807fa6ca7f. Accessed on 6 September 2018.

[15] Pardo, R. (2008). *The evaluation and optimization of trading strategies*. (2nd ed.) USA: John Wiley.

[16] Schoneburg, E. (1990). Stock price prediction using neural networks: A project report. *Proceedings of the Neurocomputing 2*, 17-27.

[17] Walter, J., Ritter, H., & Schulten, K. (1990). Non-linear Prediction with Self-organizing Maps. *Proceedings of the IJCNN International Joint Conference on Neural Networks*, pp. 17-21.