USD/KZT exchange rate forecasting using Deep Learning

Final Project on Deep Learning

Angelina Gerbolt
School of Engineering and Digital Sciences
Nazarbayeve University
Department of Computer Science
Nur-Sultan, Kazakhstan
agelbort@nu.edu.kz

Toreniyaz Shomenov
School of Sciences and Humanities
Nazarbayev University
Department of Physics
Nur-Sultan, Kazakhstan
toreniyaz.shomenov@nu.edu.kz

Yerassyl Balkybek
School of Sciences and Humanities
Nazarbayev University
Department of Physics
Nur-Sultan, Kazakhstan
yerassyl.balkybek@nu.edu.kz

***Abstract*—Financial time series prediction using deep neural networks can be regarded as one of the most challenging applications of modern time series forecasting. Popular statistical model ARIMA can be considered as a benchmark when checking the performance of deep learning models in most of the cases. Deep learning models such as MLP, CNN, BiLSTM, CNN+LSTM are applied for forecasting the USD/KZT exchange pair and discovered that CNN+LSTM works best for time series forecasting compared to other deep neural networks. Crude oil price was implemented on selected deep learning models, to illustrate the effect of oil price on the USD/KZT exchange pair.**

***Index Terms*—CNN, LSTM, ARIMA, MLP, CNN+LSTM, technical indicators, Multivariate LSTM, cruid oil price.**

I. INTRODUCTION

Foreign exchange market, also known as Forex, is the largest trading market in the world. According to the Bank for International Settlements [1], the daily exchange in foreign exchange markets averaged \$5.09 trillion in April 2016. This market is traditionally used by central banks, commercial banks, and hedge funds for currency trading. However, by the advent of the internet and its development, the market became available for small retailers. Therefore, predicting the trend of the market and performing automated trading are equally important for both large-scale and small-scale investors.

In Forex market, trading is done by selling and buying currency pairs, i.e. KZT/USD. The main strategy in this market is to buy low and sell high. For example, a trader figures out that Kazakhstani tenge will increase in price against the US dollar, so he/she will buy KZT/USD pair at lower price and when the price appreciates, sell the currency pair to gain profit.

Forex market is open 24 hours a day and 5 days of a week. Due to high volatility of the market, it is important to monitor the market constantly. It is impossible for a human to monitor the market 24 hours a day and perform the manual trading. Therefore, traders are forced to use Expert Advisors, a computer programs that perform automatic trading with no human emotions involved, and which predictions are based on logic and discipline. Expert Advisors can learn from previous market data using the latest advancements of AI and machine learning.

In our project we would like to utilize neural network based approaches (MLP, CNN, RNN/LSTM, CNN+LSTM) to predict the exchange value for KZT/USD currency pair and compare the results with popular statistical model ARIMA for stochastic time series data. The choice of this particular currency pair is justified by the fact that starting from 2014 the USD to KZT rate is very unstable in Kazakhstan because of the national currency devaluation, fluctuating oil prices and some political events. Therefore, we would like to attempt to predict th value of this pair in order to test our results on the real Forex market in Kazakhstan. Moreover, some amount of research has been already done to predict EUR/USD, AU/USD, CAD/USD and many other currency pair. However, to our knowledge, nodody tried to predict the values of KZT/USD pair before so we see this area as a promising ground for research.

II. LITERATURE REVIEW

According to Philip et al. [2], financial time series forecasting is regarded as one of the most challenging applications of modern time series forecasting. The data associated with financial time series is noisy, unstable and fluctuating. However, the recent improvements in deep learning techniques allow to use neural networks for successful modeling of financial time series. They claim that neural networks are universal function approximators that can map any non-linear function without a-priori assumptions about the properties of the data. Neural networks are also more noise tolerant, having the ability to learn complex systems with incomplete and corrupted data. In addition, they are more flexible, having the capability to learn dynamic systems through a retraining process using new data patterns.

However, Gu et al. [3] argues that when applying neural networks, shallow learning outperforms deeper learning. At the same time Gu et al. studied a range of neural networks from very shallow (a single hidden layer) to deeper networks (with up to five hidden layers) and concluded that neural network performance peaks at three hidden layers then declines as more layers are added. This may be due to the fact that unlike computer vision with its enormous datasets and strong signals, financial forecasting problems often have relatively small amount of data and tiny signal-to-noise ratio.

Several approaches of only using Multilayer perceptrons (MLP) to perform time series forecasts have been proposed and they mostly argue that MLP outperforms usual linear models. Zhang et al claim that [4] MLP gives higher accuracy than usual linear models and higher number of input nodes is more important than the number of hidden layers, consequently this reduces the error in outputs in time series forecast. This assertion was also justified in the model designed by Adewole et al [5].

The combination of CNN with RNN models for forecasting problems showing up in many research fields, giving better accuracy over some other existing models. For example, Rehman et al. [6] designed a model based on CNN with Recurrent Cartesian Genetic Programming (CGP) to predict five currency rates against Australian dollar with 98.872% accuracy. At the same time Ni et al.[7] claims that CNN can effectively exploit the spatial characteristics of data, however, it cannot mine the temporal characteristics of data. Therefore, combining the advantages of CNN (convolution) with the advantages of RNN (cycling), it is possible to better exploit the spatio-temporal characteristics of Forex time series data. The prediction algorithm C-RNN constructed by Ni et al. [7] was compared with the prediction algorithm based on Long Short-Term Memory (LSTM) neural network and CNN deep neural network. As a result, C-RNN produced the smallest root-mean-square error for all nine volatile currency pairs being studied. For example, for USD/CAD currency pair C-RNN had RMSE of 520, while LSTM had RMSE of 555 and CNN had RMSE of 590 for the same currency pair correspondingly.

Addition of some other dependent factors to the neural networks may sometimes improve the accuracy of the predictions. Bakir et al. [8] utilized both uni-variate and multivariate forecasting models to predict phone prices in European markets based on LSTM and SVM. They found that uni-variate models gave error of (RMSE) 33.43 euros, while multivariate models gave errors only up 23.640 euros for phone prices. Yin et al. [9] found that in case of multivariate time series forecasting, the periodicity of the patterns plays a significant role. If the data exhibits strong periodic patterns, then deep learning models perform best, otherwise statistical models such as ARIMA give higher accuracy.

According to Bergmeir et al. [10], cross validation on time series forecasting problems can be used to select the best models in different machine learning techniques. They have solved a particular problem using four deep learning models applying cross validation. Cross validation found the best models with the lowest errors. At the same time, Serkan et al. [11] proposed a model selection method for time series forecasting based on time series cross validation. Their method also helps to adjust the parameters and to make the selected neural networks to become more simpler. They could also conclude that neural network based models can give more accurate results than ARIMA models.

III. Time series data preparation

One can always transform the time series data into a supervised learning problem, where learning procedure is to use a function that maps an input to an output based on the examples of input-output pairs. Time series data have sequence of numbers and we can restructure the data to look like a supervised learning problem. It can be done by using previous time steps as input variables and use the next time step as the output variable.

For our project, the data was retrieved from Investing.com is in CSV format and contained historical exchange rates for USD/KZT currency pair from January 1, 2009, to August 31, 2019. It has five columns: 'Date', 'Price', 'Open', 'High', 'Low', and 'Change %'. Obtaining such data as open/close, high/low prices are important for calculation of technical indicators. Figure 1 shows the obtained data in the form of Python pandas dataframe. Later on, the crude oil price for a particular day was added as an additional column in order to test multivariate LSTM model.

	Date	Price	Open	High	Low	Change %
4	Sep 30, 2019	388.075	388.625	388.625	387.145	0.01%
5	Sep 27, 2019	388.025	387.625	388.125	387.095	0.17%
6	Sep 26, 2019	387.375	388.025	388.025	386.395	-0.17%
7	Sep 25, 2019	388.025	386.725	388.025	386.725	0.31%
8	Sep 24, 2019	386.825	386.425	386.925	385.945	0.10%
9	Sep 23, 2019	386.425	386.625	386.825	385.495	0.00%
10	Sep 20, 2019	386.425	387.025	387.045	385.495	-0.13%

Fig. 1. CNN+LSTM

For training the deep learning models a sliding window of 10 days was used as input variables and the next eleventh day was an output variable. In addition to this, we have split the data into 80% of training data and remaining 20% for for testing purposes. This percentage of data split is used in many related works for time series prediction by deep learning. In a cross validation part, 70% is used for training, 15% for validation, and 15% for testing.

IV. Time series prediction models

A. ARIMA

ARIMA, or Autoregressive Integrated Moving Average, is a statistical tool often fitted to time series data either to better understand the correlations of values in the data with each other or to predict future points in the series forecasting [12]. ARIMA model consists of three main parts: autoregressive part, moving average and the first derivative of the time series parts. The auto-regressive part (AR) based on the linear models, which describe the values of time series data based on the preceding observations. General formula of auto-regressive models can be written as follows:

$$x(t) = \sum_{i=1}^{p} \alpha_i x(t-i), \tag{1}$$

where p is the order of model. In the moving average part (MA), estimated forecasting errors, which depend on the individual values of time series, are used to estimate the next time series value. The difference between predicted and actual value commonly denoted by $\epsilon(t)$ and general formula of moving average models is:

$$x(t) = -\sum_{i=1}^{q} \beta_i \epsilon(t-i). \tag{2}$$

Combination of auto-regressive (AR) and moving average (MA) models are called ARMA$[p, q]$, which depend on both p and q:

$$x(t) = \sum_{i=1}^{p} \alpha_i x(t-i) - \sum_{i=1}^{q} \beta_i \epsilon(t-i). \tag{3}$$

Finally ARIMA$[p, q, d]$ (d is the number of differentiation steps) model can be considered when additional differentiation of the time series as well as integrating it after application of the model implemented.

This technique can be applied where data is not stable (stochastic behaviour), where an initial differencing step (corresponding to the "integrated" part of the model) can be applied several times to eliminate the non-stationarity. After that, ARMA model is fitted to the resulting values of time series and in the last step estimated forecasts have to be integrated d times. Since there is high stochastic behaviour for USD/KZT between 2016-2019 years, implementing the ARIMA model here is intuitive. Generally, ARIMA models have some dependent parameters ARIMA(p, d, q), where p, d, and q are positive integers, p is the autoregressive model order, d is the number of differentiation steps, and q is moving-average model order. Below, we provide results of ARIMA model for the testing period, which is between 2016-2019.

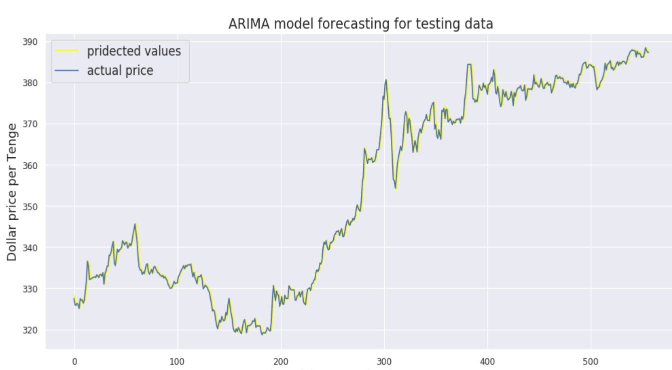

Fig. 2. ARIMA model performance, MSE is 2.283 Tenge

B. Multilayer Perceptron (MLP)

A multilayer perceptron (MLP) can be considered as a simple feedforward artificial neural network (ANN)[13]. Multilayer perceptron consists of at least three layers : an input layer, a hidden layer and an output layer.

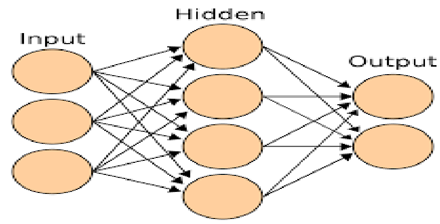

Fig. 3. MLP structure

In MLP architecture information flow is unidirectional: data is presented to $Input\ layer$, then it passes on to $Hidden\ Layer$, after that it goes to $Output\ layer$. That is how the information is distributed along the layers. Except for the input units, each unit is a neuron that uses a nonlinear activation function. Typically, units are grouped together into layers. MLP model performance for training data gives 9.80 Tenge mean squared error (MSE), while for testing data 10.43 Tenge. Generally, MLP architecture for time series data is trained much faster compared to another deep learning models such as CNN, LSTM and etc.

C. LSTM and bidirectional LSTM

LSTM stands for Long Short Term Memory and is a special kind of Recurrent neural networks. They are capable of learning long-term dependencies and specifically designed to avoid the long-term dependency problem. Hochreiter and Schmidhuber [14] were the first to introduce the LSTMs in 1997 in their original paper. All RNNs have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer, while repeating modules of LSTM have a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way, as in the Figure 4.

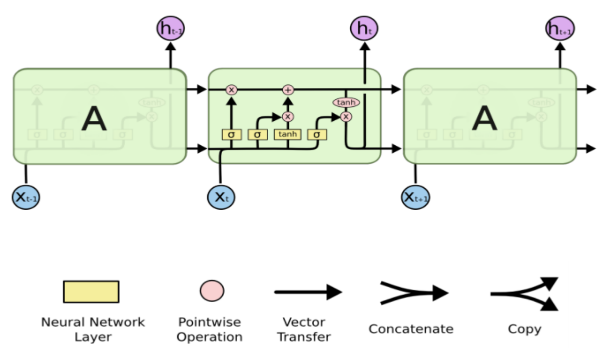

Fig. 4. LSTM structure [14]

Structure provided above can be considered as a vanilla LSTM, while in our project we used Bidirectional LSTM, whose general structure given in Figure 5.

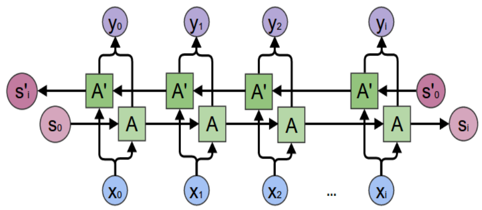

Fig. 5. Bidirectionl LSTM or BiLSTM structure [15]

Bidirectional LSTMs are neural networks where the input sequence is fed in normal time order for one network, and in reverse time order for another. The outputs of the two networks are usually concatenated at each time step, though there are other options, e.g. summation. As one can see from the structure of bidirectional LSTM, the main difference from simple or vanilla LSTM, it has a reverse time order hidden layer, which is specifically designed for time series data to extract features from future, not only from the past.

D. CNN+LSTM

In combined hybred models, such that CNN layers are used in order to extract some features from the input sequence of values continued with the LSTM layers to do time series prediction. In our project when implementing the hybrid CNN-LSTM model, we divided input sequence further subsequences (2 subsequences, 5 days per subsequence). The CNN model can be used to interpret each sub-sequence and the LSTM will piece together the interpretations from the subsequences. To our knowledge, CNN+LSTM is the best among deep learning models for stochastic time series forecasting as justified by our results of analysis in the table.

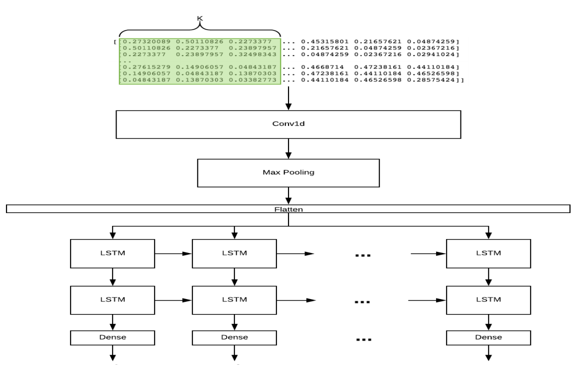

Fig. 6. CNN+LSTM [16]

E. Hyperprameters for LSTM

For all abovementioned architectures (Vanilla LSTM, Biderectional LSTM and CNN + LSTM) we use the following hyperparameters for the LSTM part, because they were used in several related works on LSTM time series prediction using deep learning models:

- Number of epochs: 40
- batchsize: 40
- Number of neurons: 36
- Number of layers: 2
- Time step: 10
- Output vector: 1
- Learning rate:0.001

V. Introduction of Brent Crude Oil price

In 2015, oil prices have plunged globally in the wake of rising oil production and concerns about global economic growth. Prices have fallen by roughly half since June 2014, rapidly falling to levels that markets have not seen since the near-total collapse of world trade during the Great Recession of 2009.

A currency that is significantly impacted by the rising and falling oil prices is commonly known as a *petrocurrency*. A petrocurrency is the currency of an oil-producing nation that has significant amounts of oil exports as a percentage of its entire export portfolio. Given such a large share of exports, the currency will rise and fall in correlation with the price of oil.

Since Kazakhstan is one of such oil-producing nation, after global plunge of oil prices in 2014, the Kazakhstani tenge (KZT) have fallen by more than two times. Therefore, it is reasonable to introduce the crude oil price into the prediction model as an additional input.

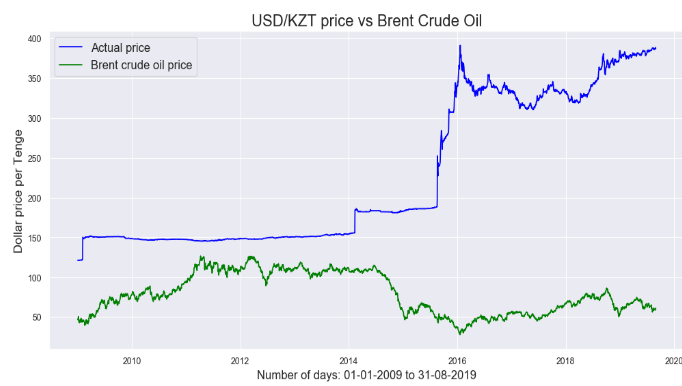

Fig. 7. USD/KZT historical price vs Brent Crude Oil price

VI. Introduction of technical indicators

In technical analysis, a technical indicator is a mathematical calculation based on historic price, volume, or (in the case of futures contracts) open interest information that aims to forecast financial market direction [17]. Technical indicators are a fundamental part of technical analysis and are typically plotted as a chart pattern to try to predict the market trend [18]. Indicators generally overlay on price chart data to indicate where the price is going, or whether the price is in an ”overbought” condition or an ”oversold” condition.

Technical indicators can generally be divided up into one of two categoriess: leading and lagging [19]. Leading Indicators give a preliminary signal. This means, that a leading indicator can anticipate future price moves in order to give the trader an edge in trading. Leading indicators provide early signal of entry or exit and allow more opportunities to trade. However, many times they will produce false signals along the way. Lagging indicators produce a signal that comes after the event and it acts like a confirmation, rather than a forecast. The biggest advantage of lagging indicators is that they typically give less false signals than the leading indicators. On the other hand, their disadvantage is that they put the trader in the trend later.

Indicators from both above-mentioned categories belong to one of the following types: trend, momentum, volatility or volume, or support and resistance indicators. Below is the list of technical indicators [17], which have been calculated and used as additional inputs to the biderectional LSTM model.

1) **Relative Strength Index (RSI) (momentum)** is a momentum oscillator that measures the speed and change of price movements. The RSI oscillates between zero and 100. Traditionally the RSI is considered overbought when above 70 and oversold when below 30.
2) **Simple Moving Average (SMA) (trend)** is one of the core indicators in technical analysis, and there are a variety of different versions. SMA is the easiest moving average to construct. It is simply the average price over the specified period. The average is called "moving" because it is plotted on the chart bar by bar, forming a line that moves along the chart as the average value changes.
3) **Exponential Moving Average (EMA) (trend)** is similar to Simple Moving Average (SMA), measuring trend direction over a period of time. However, whereas SMA simply calculates an average of price data, EMA applies more weight to data that is more current. Because of its unique calculation, EMA will follow prices more closely than a corresponding SMA.
4) **Commodity Channel Index (CCI) (momentum)** measures the current price level relative to an average price level over a given period of time. CCI is relatively high when prices are far above their average. CCI is relatively low when prices are far below their average.
5) **Moving Average Convergence and Divergence (MACD) (trend)** is a momentum oscillator primarily used to trade trends. Although it is an oscillator, it is not typically used to identify over bought or oversold conditions. It appears on the chart as two lines which oscillate without boundaries. The crossover of the two lines give trading signals similar to a two moving average system. MACD crossing above zero may indicate that an asset will rise in value, while crossing below zero indicates that the price of an asset will fall.
6) **Stochastic Oscillator (SR) (momentum)** is a momentum indicator that shows the location of the close price relative to the high-low range over a set number of periods. The indicator can range from 0 to 100.The closing price tends to close near the high in an uptrend and near the low in a downtrend. If the closing price then slips away from the high or the low, then momentum is slowing. Stochastics are most effective in broad trading ranges or slow moving trends.
7) **Bollinger Bands (volatility)** are envelopes plotted at a standard deviation level above and below a simple moving average of the price. Because the distance of the bands is based on standard deviation, they adjust to volatility swings in the underlying price.
8) **Ichimoku Cloud (support and resistance)** is a type of chart used in technical analysis to display support and resistance, momentum, and trend in one view.
9) **Average Directional Index (ADX) (trend)** can be used to help measure the overall strength of a trend. The ADX indicator is an average of expanding price range values.
10) **Daily Return (DR)** measures the dollar change in a stock's price as a percentage of the previous day's closing price. A positive return means the stock has grown in value, while a negative return means it has lost value. A stock with lower positive and negative daily returns is typically less risky than a stock with higher daily returns, which create larger swings in value.

In order to calculate these technical indicators, we have used a *ta (Technical Analysis)* package for Python.

The figure below shows the technical indicators overlaid on USD/KZT price data.

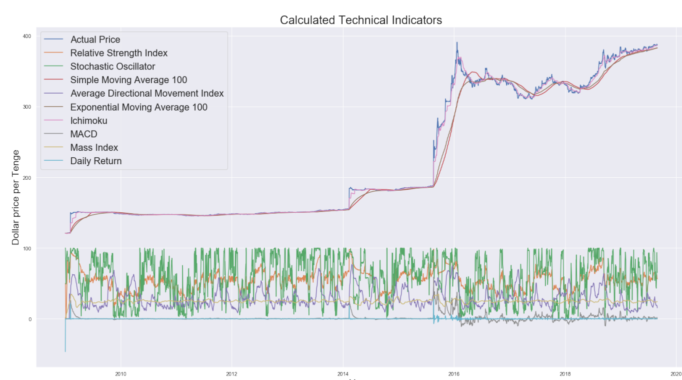

Fig. 8. Technical indicators overlaid on USD/KZT price data

VII. Results of Deep Learning Models

The figure below shows the results of testing prediction of MLP, Biderectional LSTM, and CNN+LSTM models against the actual USD/KZT price for the last 560 days.

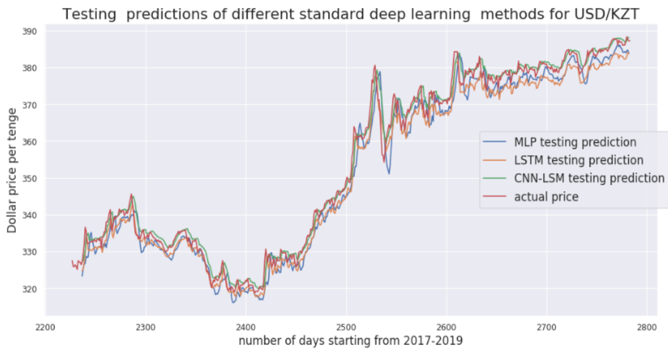

Fig. 9. Standard Deep Learning models for USD/KZT

It can be seen that the most accurate prediction is made by CNN + LSTM model. The training and testing performance of this model is further plotted alone versus the actual price.

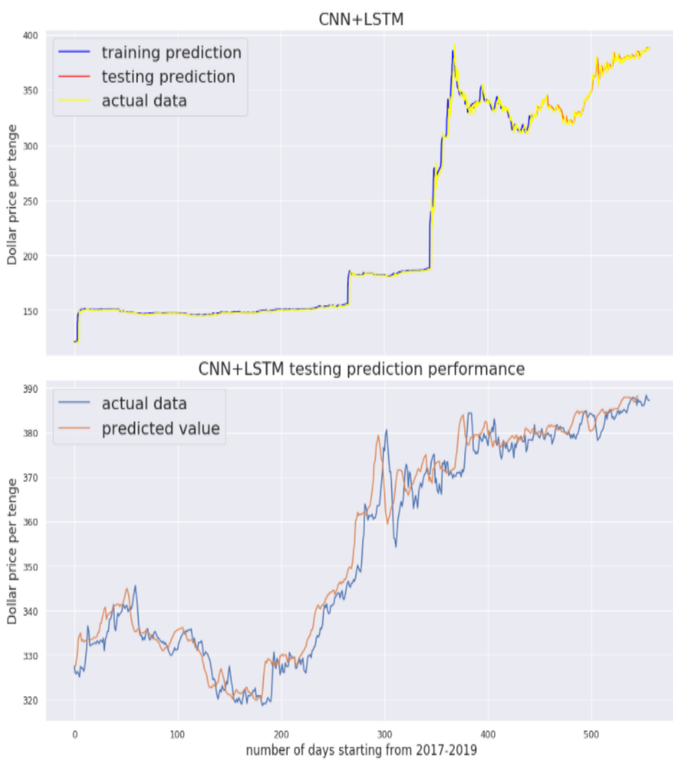

Fig. 10. Best results can be obtained using CNN+LSTM architecture

The figure above suggests that the CNN+LSTM model follows the trends of the actual price very closely, however, plotting the testing period alone reveals that even these predictions are not very accurate and lag the actual price by several days.

Figure 13 shows the results for multivariate LSTM with Brent Crude Oil Price as an additional input channel for the model. It can be seen that the introduction of the oil price improves the training phase, however, performes poorly on the test dataset. This is due to the fact that the USD/KZT price is influenced by many not captured factors and parameters, and the relationship between oil price and currency is not exactly linear.

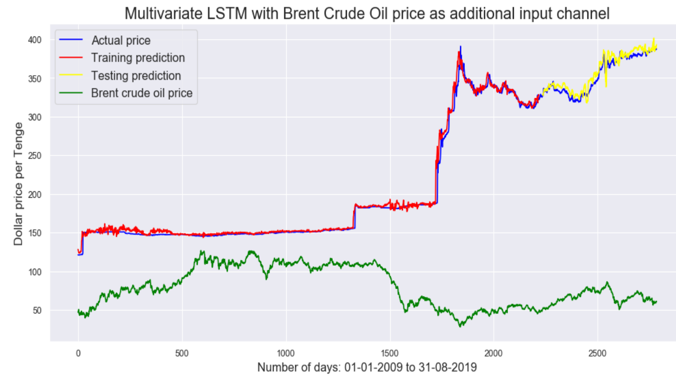

Fig. 11. Multivariate bidirectional LSTM + Brent Crude Oil price as an input

Next figure illustrates the effect of using multivariate LSTM with all ten technical indicators introduced as additional input channels for the model. This figure suggests that using all technical indicators as input channels is a bad idea, since the training and testing prediction worsened significantly due to the high variability and stochastic behavior of some technical indicators.

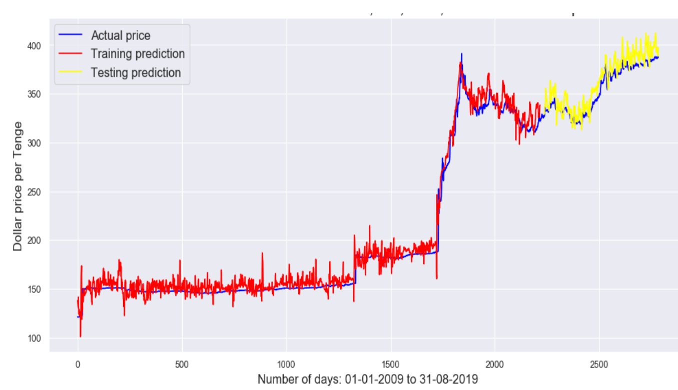

Fig. 12. Multivariate bidirectional LSTM + ALL technical indicators as additional inputs

After trial and error, it was deduced that the technical indicators that bring more accuracy to the model are those, which react only on significant price changes, i.e. trend technical indicators, such as MACD, ADX, DR. On the other hand, the momentum technical indicators, such as RSI, CCI, and SR bring more instability to the model performance. Therefore, the next figure represents the effect of using MACD and DR technical indicators as two additional input channels to the multivariate LSTM model.

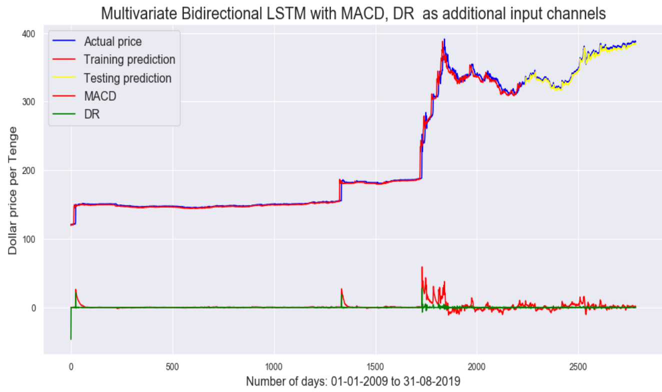

Fig. 13. Multivariate bidirectional LSTM + MACD, DR as additional inputs

After presenting the effects of oil price and technical indicators to USD/KZT price, results of the abovementioned deep learning models are provided in the table in a compact form to make some conclusions.

Model	MSE$_{train}$	MSE$_{test}$	MAPE$_{train}$	MAPE$_{test}$
ARIMA	3.43	2.83		
MLP	11.21	7.99	0.50	0.53
CNN	10.04	9.83	0.67	0.66
LSTM	8.11	10.75	0.66	0.71
CNN + LSTM	7.13	7.62	0.48	0.56
LSTM + Oil Price	7.18	19.92	0.73	0.90
LSTM + tech ind	16.74	28.58	1.44	1.52
LSTM + MACD, DR	5.29	4.17	0.36	0.40

VIII. Cross validation on Bidirectional LSTM

Usual cross validation technics such as k-fold splitting, leave p oup cannot be applied for time series data due to its ordered time dependence. In real world applications, time-ordered data are used in order to predict the future events based on the past observations. The cross validation structure for time series data can be described as follows (fig. 14). In the beginning some proportion of the data can be used only for training. The remaining of the data can be used in order to perform cross validation, splitting them into n parts. In the first iteration, the first part can be used in order to test the pre-trained model. In the next iteration, this test part will be joined to the overall training part and again pre-trained. This pre-trained part will be tested using the following part in the split.

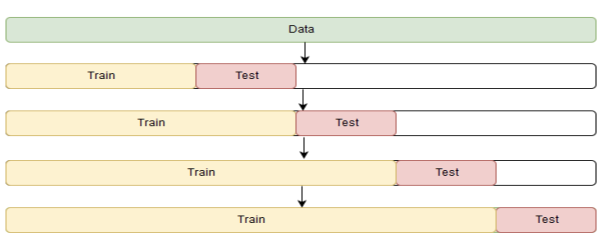

Fig. 14. Cross-validation structure for time series data [20]

This scheme will be continued until the end of the data. At each iteration, we can compute MSE error E_i obtained from the testing. The total error E_{valid} can be averaged from all n splits. The time splits can be performed easily using $TimeSeriesSplit$ from $sklearn.model\_selection$ library. The only requirement is to provide the number of splits n_splits into the $TimeSeriesSplit$ function, and split the data according to the returned indexes [21].

In our problem, in order to see the effect of this scheme, we have taken first 70% of the data only for training and next 15% for time series cross validation. The number of splits (n_splits) was set to 10. The last 15% is left for testing. The overall graph for training, validation and testing using time series cross validation can be found below (fig. 15). Running this scheme, our program returns the following result: the overall train score is $E_{train} = 9.09MSE$, the averaged validation error $E_{valid} = 5.48MSE$ and the overall test error is $E_{test} = 7.54MSE$. Comparing this result with the corresponding LSTM error from table above, we find that the total test error has been slightly reduced.

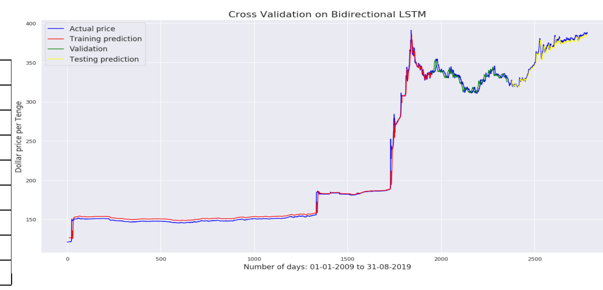

Fig. 15. Result from cross validation

In literature, this scheme sometimes called as walk-forward validation and it can also be used for financial evaluations in time series forecasting problems. In this case, the remaining data should be divided into the remaining data size. For large data, this scheme is very computationally expensive, since in each iterations the previous data should be saved and pre-trained again[22].

IX. Financial evaluation on Bidirectional LSTM

In real world, these models can be useful if they give some certain profits to the companies, financial sectors etc. Now let us consider Bidirectional LSTM in terms of financial benefits for those who wishes (let them call "agents") to gain profits continuously exchanging their money from USD to KZT and vise versa. The main idea is the following. Each day, the agent will predict the value ($pred\_value$) for KZT per USD for the next day and if the $pred\_value$ is higher than today's value ($today\_value$) more than by some certain threshold ($pred\_value$-$today\_value > thresh$), then he will exchange some portion of his KZT to USD. On the contrary, if $pred\_value$-$today\_value < -thresh$ happens, then he will exchange some portion of his USD to KZT. In this problem, we allocated 10000 USD and 3 million KZT, and compared this money at the end of the process. Using Bidirectional LSTM, we used two approaches. In the first case we only trained our model for 2678 days and predicted for the next day. Applying our financial strategy, we retrained our model for that single day. This continued by 84 days as in sliding window approach. Using $Keras$, retraining the model for each day is very time consuming. The computational cost increases exponentially as we wish to test as more days including. As a result we had to test only for the last 84 days, which took more than 2 hours. In the second approach, we trained our model for 2227 days and tested for the last 557 days. In this approach, on the each day of testing, we train our model from $sctrach$ (which means, training for the $train\ days$ + $all\ the\ days\ until\ testing\ day\ (i)$) and predict for the next day, $testing\ day\ (i+1)$. Overall it took 5 hours.

A. *Results from Financial evaluation. Model 1*

In the first model, starting with 10 thousand USD and 3 million KZT (in total 17876 USD) we end up with optimal threshold ($thresh$ = 0.0044) at 17766 USD, which means we lost 110 USD. This model failed. In the fig. 17, you can see that after some threshold, there is no changes at all.

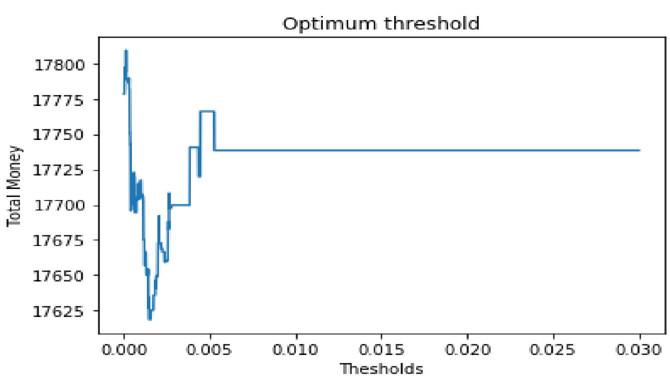

Fig. 16. Threshold optimum

B. *Results from Financial evaluation. Model 2*

In the second model, starting with 10 thousand USD and 3 million KZT (in total 19192 USD), at the optimum threshold ($thresh$ = 0.0032), we end up with only 18919 USD. As a result, we lost 273 USD in 546 days.

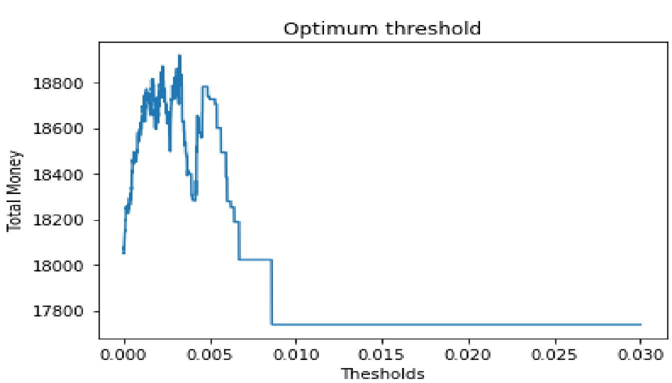

Fig. 17. Threshold optimum

These two models in bidirectional LSTM failed, which hints bidirectional LSTM fails in USD/KZT exchange forecasting.

X. Financial evaluation on CNN + bidirectional LSTM

In this part we will provide the financial evaluation results obtained from CNN+LSTM hybrid model. The strategy how we determined the profits using this model is the same as in single bidirectional LSTM, but we adhered on the second model which enables us to test for larger data due to its faster computational time.

A. *CNN+LSTM without oil prices*

In this model we started with 19192 USD and end up with 18228 USD at the optimum threshold 0.0152 which can be seen from fig. 18

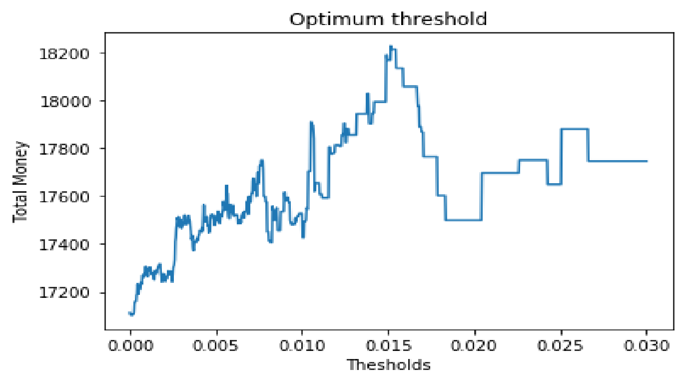

Fig. 18. CNN+LSTM financial outcomes at different thresholds

B. *CNN+LSTM with oil prices*

This model started with 19192 USD and at the optimum threshold 0.0157 we finished with 18223. The fig. 19 shows how financial outcome changes depending on different thresholds.

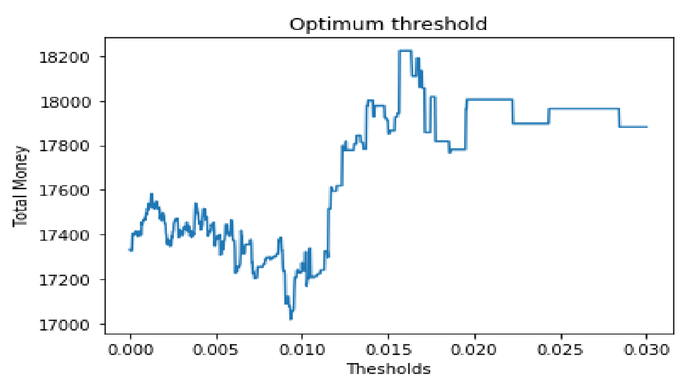

Fig. 19. CNN+LSTM with oil prices included financial outcomes at different thresholds

Overall, CNN+bidirectional LSTM works worse independent on whether to include the oil prices or not to include than bidirectional LSTM alone.

XI. Discussion

From Table 1 in previuos section, it can be seen that the best result for both training and testing is obtained with ARIMA, which is considered as a benchmark approach to the time series prediction problems.

Among deep learning models the best performance was obtained when applying a hybrid CNN+LSTM architecture with 7.13 and 7.62 train and test errors respectively.

With the introduction of crude oil price as an additional input channel for the multivariate bidirectional LSTM, the training error was reduced (7.18), thus outperforming CNN+LSTM architecture, however, the test error for this model is rather high (19.92). This can be explained by the fact that it is possible that the oil price does not influence the USD/KZT pair immediately and its effect accumulates over time. Therefore, a bigger sliding window (more than 10-days period) is suggested to use for training the model while using crude oil price as an additional input.

Regarding technical indicators, it can be said that the involvement of all major technical indicators worsens the performance of multivariate bidirectional LSTM significantly, affecting both training and testing errors. On the other hand, the use of selected technical indicators (mostly trend indicators) outperforms the results of CNN+LSTM architecture with USD/KZT closing price as one input.

However, besides obtaining RME and MAPE errors to evaluate the performance of the model, it is suggested to use financial assessment analysis. This part of the procedure is the subject of our future work.

XII. Conclusion

A statistical tool ARIMA, which is specialized for stochastic time series data, performed better compared to the CNN, LSTM and MLP models, while CNN+LSTM was the better among Deep Learning models used.

Introduction of oil price brings a lot of improvement on train data, however, performs poorly on test data. In general, it should be assumed that oil price affects the USD/KZT exchange pair by some orders, while there are many factors have to be taken into account. Introduction of selected technical indicators brings a little improvement in both train and test data. Performing financial evaluation on bidirectional LSTM, in is easy to see that there is no profit can be gained from the models, but some financial losses instead.

References

[1] Monetary Bank for International Settlements and Economic Department. Triennial central bank survey: Foreign exchange turnover in april 2016. September 2016.

[2] Adewole Adetunji Philip, Akinwale Adio Taofiki, and Akintomide Ayo Bidemi. Artificial neural network model for forecasting foreign exchange rate. *World of Computer Science and Information Technology Journal*, 1(3):110–118, 2011.

[3] S. Gu, B. Kelly, and D. Xiu. Empirical asset pricing via machine learning. 2018.

[4] Gioqinang Zhang and Michael Y Hu. Neural network forecasting of the british pound/us dollar exchange rate. *Omega*, 26(4):495–506, 1998.

[5] Adetunji Philip Adewole, Adio Taofiki Akinwale, and Ayo Bidemi Akintomide. Artificial neural network model for forecasting foreign exchange rate. 2011.

[6] Mehreen Rehman, Gul Muhammad Khan, and Sahibzada Ali Mahmud. Foreign currency exchange rates prediction using cgp and recurrent neural network. *IERI Procedia*, 10:239–244, 2014.

[7] Lina Ni, Yujie Li, Xiao Wang, Jinquan Zhang, Jiguo Yu, and Chengming Qi. Forecasting of forex time series data based on deep learning. *Procedia computer science*, 147:647–652, 2019.

[8] Houda Bakir, Ghassen Chniti, and Hédi Zaher. E-commerce price forecasting using lstm neural networks. *Int. J. Mach. Learn. Comput*, 8:169–174, 2018.

[9] Jiaming Yin, Weixiong Rao, Mingxuan Yuan, Jia Zeng, Kai Zhao, Chenxi Zhang, Jiangfeng Li, and Qinpei Zhao. Experimental study of multivariate time series forecasting models. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2833–2839. ACM, 2019.

[10] Christoph Bergmeir and José M Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

[11] Serkan Aras and İpek Deveci Kocakoç. A new model selection strategy in time series forecasting with artificial neural networks: Ihts. *Neurocomputing*, 174:974–987, 2016.

[12] Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci. *Introduction to time series analysis and forecasting*. John Wiley & Sons, 2015.

[13] Jason Brownlee. How to create an arima model for time series forecasting in python, Sep 2019.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[15] Ceshine Lee. Understanding bidirectional rnn in pytorch, Mar 2018.

[16] Yitong Ren. Get started with using cnn lstm for forecasting, Oct 2019.

[17] John J. Murphy. *Technical analysis of the financial markets : a comprehensive guide to trading methods and applications (2nd ed.)*. New York [u.a.]: New York Inst. of Finance., 1999.

[18] The Nippon Technical Analysis Association (NTAA). The first step guide to technical analysis.

[19] Sakshi Agarwal. Lagging vs. leading technical indicator – do you know the difference? 2019.

[20] Getting started with time series analysis.

[21] Jason Brownlee. How to backtest machine learning models for time series forecasting, Aug 2019.

[22] Samuel Monnier. Cross-validation tools for time series, Sep 2018.