Multitask machine learning for financial forecasting

LUCA DI PERSIO * OLEKSANDR HONCHAR †

***Abstract*—**
In this paper we propose a new neural networks based regularization method requiring only 4 additional hyperparameter and that can be easily injected in any machine learning architecture. It is based on the use of auxiliary loss functions designed to appropriately learn data momenta. Our approach can be used both for classification and regression problems. A comparative analysis with real time series will be provided concerning cryptocurrency data, showing improvements in accuracy of about 5% with respect to existing approaches, without requiring additional training data or further parameters. The presented approach constitutes an innovative, new step towards the *statistical moments oriented regularization* scheme for statistical forecasting.

***Keywords*—** Machine learning, deep learning, neural networks, forecasting, multitask learning, cryptocurrency.

I. INTRODUCTION

Introduced several years ago, machine learning algorithms have learned an impressive boost during recent years mainly with respect to recognition tasks, spanning from images to speech, as well as concerning forecasting issues. Such an increasing interest is essentially the by-product of a couple of factors: first the so called *big data* scenarios, second the possibility to effectively treat huge amount of data and algorithms processing them within reasonable execution times and with relatively small costs.

This is, in particular, true when we have to analyze time series related to, e.g., meteorological events, energy production, financial data, human beings behavior over the net, etc.

It is worth to mention that more often than we could think, such aspects are strictly related, as in the case of financial products based on renewable energy production, where we have to take into account data coming from, e.g., solar cells and/or windmills output related to meteo characteristics over specified time periods, people beliefs about the use of such type of energies versus oil based ones, and financial time series linked to costs productions and management of energy derivatives over hourly to daily markets.

Therefore, each type of data source has to be considered as related to the other ones. This is why, traditionally, a multivariate statistical type analysis has been considered, see, e.g., [4, 5, 17], and references therein.

The main goal being the one to explicitly treat correlations between different sources.

More precisely, these approaches are typically based on the serial correlation in the data, often assuming that possible random sources could be modeled as white noise components on the basis of past observations. This the starting point for are autoregressive models family (AR), moving average models (MA), often combined into autoregressive integrated moving average models (ARIMA), where the variance in time series is modelled with autoregressive conditional heteroskedasticity models (GARCH).

Usually, ARIMA and GARCH are combined to make better forecasts and to model situation when statistics is changing over time state space models, as, e.g., in the case of the Kalman-type filters, see, e.g. [2, 3, 21], and references therein.

Despite these approaches are often applied, they suffer of some relevant drawbacks. First, they are based on *serial correlation* assumption, while the real underlying process can be highly non-linear.

Second, these models usually are not able to efficiently treat memory effects and are not so accurate when dealing with o long time series. Moreover, they are difficult to be tuned with respect to local as well as to global geometric characteristics of the data.

Furthermore, such type of methods are not concretely applicable when large amount of different type of data have to be taken into account simultaneously aiming at producing accurate forecast in almost real time.

This is the main reason why automated analysis/selection procedures as the ones offered within the machine learning scenario have been proposed several years by now, being then successfully applied during recent periods, mainly exploiting the exponential growth in computational speed and the lowering of requested hardware.

In particular, neural networks and their related *deep architectures* features, have shown their power from both the classification and regression point of view, allowing to efficiently treat a number of forecasting problems based on heterogeneous and interconnected temporal data.

A negative side of such an approach is related to the tendency of ML methods to overfit. This means that it is not rare the case when multilevel NNs techniques tend to do not well model themselves according with new input, trying, instead, to give too much weight to old data. The latter phenomenon often give rise to undesirable memory effects.

To overcome this issue, alternatives have been proposed, such, e.g., L1, L2 regularizations, *dropout* smoothing, noise regularization, data augmentation, etc.

Nevertheless, these methods are mainly based on a reduction of the number/relevance of the model parameters, hence oversimplifying the model.

While, data augmentation techniques require a lot of efforts and makes the model training much more time consuming. Moreover it is worth to mention that such remedies to the *memory effect* need their own particular hyperparameters, which implies a further number of hyperparameters to be trained. Last but not least, these methods are definitively not the best in terms of forecasting accuracy, especially when dealing with time series characterized by medium to high level of variance.

*luca.dipersio@univr.it, Dept. of Computer Science, University of Verona, Strada le Grazie 15 37134 Verona

†oleksandr.honchar@studenti.univr.it, Dept. of Computer Science, University of Verona, Strada le Grazie 15 37134 Verona

Taking into account previously cited issues and limits, in this paper we propose a new regularization method, namely a *multitasking learning* solution, that does not require additional memory, being developed with just 4 additional hyperparameters and that can be easily injected in any machine learning model.

The main idea is to use auxiliary loss functions that are designed to learn data's statistics. Moreover, our approach can be used both for classification and regression problems.

To show the validity of our proposal, we benchmark its output on time series from different domains, showing significant improvements in accuracy of financial forecasts on cryptocurrency data, up to 5%, and requiring nor additional training data, neither overparametrizing the model.

We believe that statistical moments oriented regularization should become new pipeline standard for statistical forecasting. In fact, our method, contrary to the ones already proposed, instead of optimizing a single loss, as, e.g., the mean squared error between forecast datum and its real counterpart, allow to optimize over several *tasks* simultaneously.

The price we have to pay for such an optimization approach is due to the fact that different *tasks* come from different sources of data, hence requiring designing of rather complex machine learning architectures.

This latter issue is mainly concerned with computational efforts and related hardware demands, hence its relevance is rapidly decreasing.

The work is organized as follows: we will first show that we can augment simple feedforward neural network models with auxiliary tasks of forecasting not just the main objective (future price change of financial asset as our use case), but also rolling statistics as mean, variance, skewness and kurtosis.

Calculation of these statistics is cheap, efficient and does not increase the computational times required by training and/or evaluation process.

Our empirical results prove that our approach allows to increase directional accuracy of the forecasts up to 5% comparing to the baseline.

Then, we show how these auxiliary tasks reduce variance of the weights as other regularizers do, a feature that seems to do not hold, at least empirically testing it.

Moreover, we provide several performance metrics results both for regression and directional movement classification tasks that are useful for evaluation of the performance.

II. Background

In what follows we will review general machine learning approaches previously applied to financial time series forecasting. This will be helpful later to better underline the advantages due to our new proposal applied to real time series.

Before the *deep learning era* advent, a lot of statistical classifiers have been applied as, e.g., logistic regression [22] , Supported Vector Machine (SVM) techniques [16], decision trees and random forests techniques, [18], with respect to a rather large class of data and features.

Because of the tumultuous developments reached within the NNs scenario during recent years, it even more interesting to focus our attention in reviewing last developments related to the interplay between NNS and their deep learning realizations, particularly from the financial point of view, see, e.g., [7, 8].

In fact, in such framework both genetic algorithms and reinforcement learning approaches have been applied to attack nested portfolio optimization problems whose intrinsic nature does not suit traditional statistical approaches.

In this direction several examples can be made, see, e.g., [1] where the long-term (one month ahead) forecasting of prices in Japanese stock market have been analyzed by deep learning techniques, [13], where deep neural networks were compared to decision trees on different data sets including prices, volume of trades and dummy variables to encode date and time information, or in [19], where NNs approaches have been compared versus classical time series analysis, outlining significant improvements; moreover NNs neworks have been also applied to macroeconomic indicator forecasting, as in [**?**], and high-frequency trading applications, see [14], again showing their potentials.

III. Data preparation

In our analysis, as a benchmark data, we have used prices of well known cryptocurrency Ethereum (ETH), indexed as high, open, low and close prices, for every day and considering trading volume, market cap and the amount of tweets on this currency from the 7th of August 2015, and until the 21st January of 2018 (figure 1).

This implies that we have 7 variables for each trading day, and we have considered a window over 7 days to predict the next one.

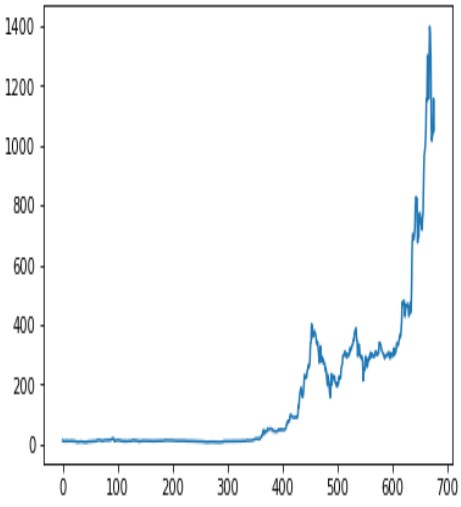

Figure 1: *ETH close prices*

To normalize data we use first order differences which gives us weakly stationary time series centered around zero, but with varying variance (figure 2).

After several experiments, we decided to use the first 80% of data as training set, while the last 20% has been adopted as

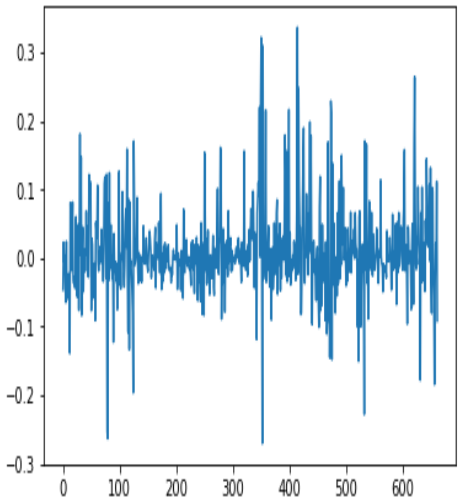

Figure 2: *ETH returns*

testing data. For multitask auxiliary targets we computed rolling mean, volatility, skewness and kurtosis,

$$\mu = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{1}$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i-\mu)^2} \tag{2}$$

$$\gamma_1 = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^3\right] = \frac{\mu_3}{\sigma^3} \tag{3}$$

$$\mathrm{Kurt}[X] = \mathrm{E}\left[\left(\frac{X-\mu}{\sigma}\right)^4\right] = \frac{\mu_4}{\sigma^4} \tag{4}$$

considering a *rolling window size* equal to 7. Hence, before training the neural networks we got a data set with 528 samples of train samples, each size of 49 variables, because we have to consider 7 days of 7 variables, and target vectors with varying length equal to 1, 2 or 5 depending on the experiment. Moreover we did not assume auxiliary task (baseline), while we have taken a single auxiliary task or all tasks together.

It is worth to mention that we have trained our network to forecast real values, but, for the trading applications, it is often needed to have a binary indicator, as, e.g., call for action.

This has implied our choice to concentrate on evaluation of metrics for binary classification, which also imples that our continuous output have been appropriately transformed into binary ones, hence allowing to consider the *sign* of variation as the variable to be forecasted.

IV. Neural network architecture

For the sake of clarity in performing our computations, we decided to use a very simple neural network architecture. In particular, we considered a single hidden layer perceptron, composed by 16 neurons in the hidden layer, while the activation function has been chosen to be the Rectified Linear Unit (ReLU) one:

$$f(x) = x^+ = \max(0,x) \tag{5}$$

The equations of a feedforward neural network are straightforward: superposition of functions (in our case - layers):

$$y(x_1,\ldots,x_n) = f(w_1x_1 + w_2x_2 + \ldots + w_nx_n) \tag{6}$$

And as the last activation for the prediction we use no non-linearity (as we want to foreacst real value).

This choice could seem as an over-semplification of the problem, since more sophisticated approaches can be used as well, as, e.g., deep learning architectures, that exploit local geometry, hence using convolutional neural networks, or long-term memory methodologies, hence exploiting recurrent neural networks.

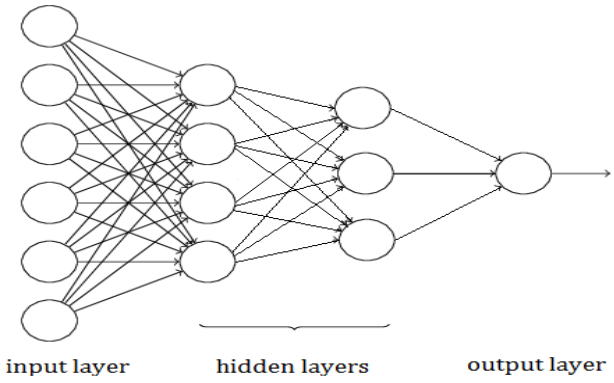

Figure 3: *Schematic neural network architecture*

Nevertheless, some considerations should be taken into account.

In particular, we aim at discovering how additional losses affect the training phases even when simple approaches are considered. Of course, we could choose logistic regression as an alternative baseline, but it tends to overfit without accurate regularization, and this is a characteristic that we do not to use.

Moreover, and this is a crucial point, when dealing with poor data-sets, more advanced NNs methodologies tend to produce high errors, even in case of short-time forecasting, see, e.g., [11, 23, 24] and references therein.

This is exactly our case, since we cannot count on a large amount of training data. It is worth to mention that well performing deep learning architectures require hundreds of thousands of samples.

Modern financial market theories as the Capital Asset Market Pricing (CAPM) or Arbitrage Pricing Theory (AP), typically assume explanation of price movements as a combination of linear factors, hence without using advanced non-linear based analysis instruments.

Last but not least, auxiliary losses are connected to the main one (with index 1) as following

$$L_1(y^1,t^1) + \sum_N \lambda_i \cdot L_i(y^i,t^i)\,; \tag{7}$$

which implies that, while *backpropagating*, the error produced by the process itself results from all the losses with pre-defined weights λ_i.

As the optimizer for the neural network, we have chosen the *Adam algorithm*, improved following the recent AMSGrad, [20], approach. In particular, our techniques improves the already obtained results allowing for a better convergence stability.

The learning rate has been chosen relatively small, i.e.: 0.0001, and we have adopted the *early stopping approach* as the only regularization method. Therefore, we stopped learning if performance (decrease of the loss) did not improve over *the last* 10 epochs.

Adaptive Moment Estimation (Adam) is a method that computes adaptive learning rates for each parameter.

In addition to storing an exponentially decaying average of past squared gradients, it also keeps an exponentially decaying average of past gradients mt, similar to momentum.

Whereas momentum can be seen as a ball running down a slope, Adam behaves like a heavy ball with friction, which thus prefers flat minima in the error surface. We compute the decaying averages of past and past squared gradients respectively as follows:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad (8)$$

As *mt* and *vt* are initialized as vectors of 0's, the authors of Adam observe that they are biased towards zero, especially during the initial time steps, and especially when the decay rates are small (i.e.β_1 and β_2 are close to 1). They counteract these biases by computing bias-corrected first and second moment estimates:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad (9)$$

They then use these to update the parameters just as we have seen in Adadelta and RMSprop, which yields the Adam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \quad (10)$$

Concerning the right weight for each auxiliary loss, we have used a *Bayesian optimization* type algorithm, namely the *Tree of Parzen Estimators* (TPE), see, e.g., [10, 6]. This method is particularly used to hyperparameters optimization purposes, acting as black-box optimizer.

The objective, for the hyperparameter optimization process, has been chosen to be R^2 score (4), where the numerator is the sum of squares of residuals, while the denominator equals the total sum of squares, proportional to the variance of the data, then using this metric as the main one to explain the variance values characterizing the obtained forecasts. In particular, using standard notations, we have

$$MAE = \sum |x_i - \widehat{x}_i| \quad (11)$$

$$MSE = \sum (x_i - \widehat{x}_i)^2 \quad (12)$$

$$R^2 = 1 - \frac{\sigma^2}{\sigma_x^2} \quad (13)$$

$$MCC = \frac{TP*TN - FP*FN}{(TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)} \quad (14)$$

It is worth to mention that even if we have conducted our computations in view of regression tasks, particularly aiming at predicting continuous time values, we also re-calculated forecasts as a *direction*.

This is because we focus on the forecast of sign, evaluating its accuracy as a binary classification problem.

Let us also recall that the regression metrics used are MSE and MAE.

As regression metrics we use Mean Squared Error (MSE), Mean Absolute Error (MAE) and R^2 score, while, as binary classification metrics, we used simple accuracy, confusion matrix, precision, recall and Matthews correlation coefficient, see eq. (14) , calculated directly exploiting the confusion matrix using, respectively, false positive, false negative, true positive and true negative values, see, e.g., [12].

V. Experimental results

We have set up our experiments as following:

1. Learn a baseline network with a single loss and evaluate it
2. Learn a baseline network and auxiliary loss for rolling mean forecasting
3. Learn a baseline network and auxiliary loss for rolling volatility forecasting
4. Learn a baseline network and auxiliary loss for rolling skewness forecasting
5. Learn a baseline network and auxiliary loss for rolling kurtosis forecasting
6. Learn a baseline network with all above mentioned auxiliary losses together

In each experiment we have established appropriate weights for auxiliary losses with TPE algorithm.

The results of our experiments have been summarized in tables below, as well as by the graphs we provided to better show the price change forecasts.

i. Baseline

The MSE, MAE , R^2 and Matthews correlation coefficient for the aforementioned computations, have been defined as follows:MSE = 0.0069, MAE = 0.0611, R^2 = -0.4721 MCC = 0.0341, hence gaining something more with respect to the simple *random guessing* approach. Moreover, the neural networks weights have distribution with mean 0.0362 and variance 0.1737

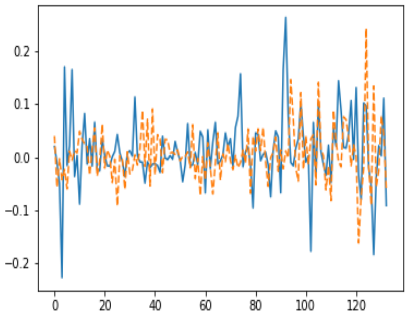

Figure 4: *Baseline network with a single loss forecasts*

	Precision	Recall	F1 score
DOWN	0.47	0.50	0.48
UP	0.57	0.53	0.55

Table 1: *Precision / recall / F1 score for a network with a single loss*

ii. Single auxuiliary loss

In what follows, we report the results obtained adding each of the auxiliary losses to the baseline network. The forecasting plots are on the figures 4, 5, 6, 7, while the main metrics comparisons have been summarized in table 6.

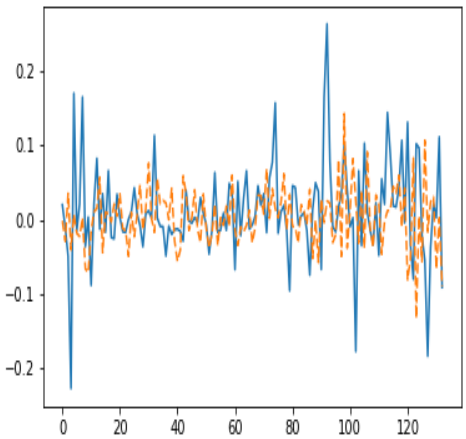

Figure 5: *Baseline network with a main loss + mean prediction loss*

iii. All auxiliary losses

The MSE, MAE , R^2 and Matthews correlation coefficient for this last experiment have been chosen to be equal to: MSE = 0.0079, MAE = 0.0684, R^2 = -0.9727, MCC = -0.04516. The weights λ_i for auxiliary losses are 0.6359, 0.5271, 0.2230 and 0.2812, respectively.

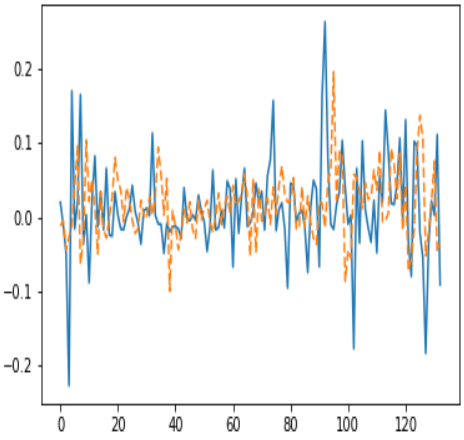

Figure 6: *Baseline network with a main loss + volatility prediction loss*

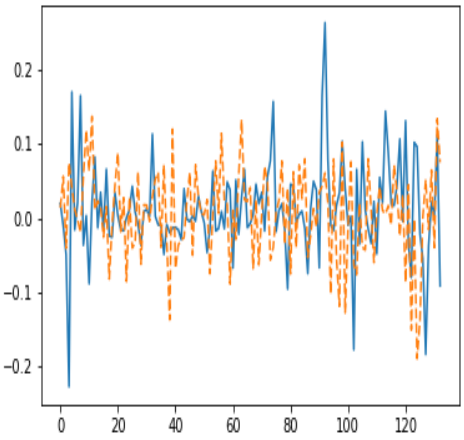

Figure 7: *Baseline network with a main loss + skewness prediction loss*

We underline that this network could not converge to an acceptable result, see table 7.

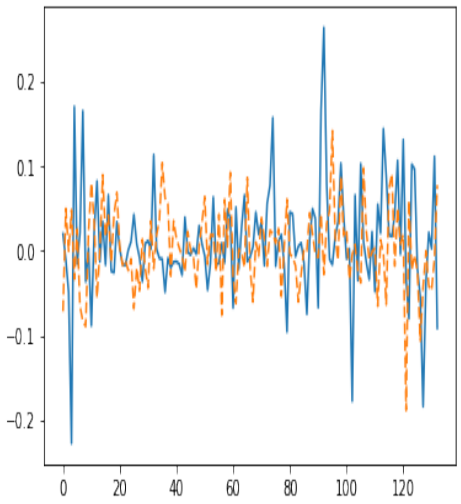

Figure 8: *Baseline network with a main loss + kurtosis prediction loss*

	Precision	Recall	F1 score
DOWN	0.52	0.53	0.52
UP	0.61	0.59	0.60

Table 2: *Precision / recall / F1 score for a network with a main loss + mean prediction loss*

VI. Experiments discussion

Our baseline approach after conversion results to the binary classification could get 0.52 of F1 score. This witnesses how such a model already can find some useful patterns in the data.

We should notice though that recall for the *down* price movement is just 0.5, while for the *up* movement, we have 0.53, which is most likely happening due to class imbalance.

It also clear, comparing obtained results, that the additional tasks that could boost performance at most, are represented by rolling mean and rolling kurtosis.

In fact, they could improve final F1-score to 0.55 and 0.56, respectively.

Success of mean auxiliary loss can be explained with the fact of describing momentum of the price changes. Skewness prediction success is even more interesting result. In particular, assuming that price changes according to a Gaussian distribution, the skewness parameter provides us useful info about which side, positive or negative, will be followed by price changes.

We also have tried to build a forecasting algorithm including all the four auxiliary losses to the architecture alongside with main objective, but it has shown worse performance even comparing to the baseline.

This is not an unexpected result, since we have to take into account the high impact of objectives showing bad performance, namely volatility and kurtosis.

Moreover, we also have to consider possible fails within the hyperparameter optimization processes.

Latter point is surely an interesting line for future research.

In particular, we aim at extending to an analytical approach the search for the *right* weight of a loss, which is, until now, a

	Precision	Recall	F1 score
DOWN	0.51	0.43	0.47
UP	0.59	0.66	0.62

Table 3: *Precision / recall / F1 score for a network with a main loss + volatility prediction loss*

	Precision	Recall	F1 score
DOWN	0.52	0.53	0.52
UP	0.61	0.59	0.60

Table 4: *Precision / recall / F1 score for a network with a main loss + skewness prediction loss*

hyperparameter chosen according to the results of a number of experiments, hence *empirically obtained*, or is tuned by hyperparameter optimization process, as in the tree-structured Parzen estimator (TPE) approach, or incorporated in the NN-architecture itself, being a part of a bigger meta-learning architecture.

We also had a hypothesis that auxiliary losses will act like more known regularizers, like L1 or L2 regularization. This put less importance to the algorithm parameters that can be closer to zero.

That is why, after every experiment, we have also taken weights matrix to be between inputs and hidden layer, then checking for its mean and variance.

As it is easy to note looking at the associated numerical tables below, such choices do not produce a significant reduction of variance of the ways.

Therefore, we conclude that auxiliary loss functions in financial forecasting could not be taken as a classical regularizers, even if they have been injected to the training process in a similar way, but, instead, they should be intended as an additional term to the general optimization objective.

Having said that, it is also important to note that an improvement ranging from 3% to 6%, w.r.t. the baseline, constitutes a rather important result.

This is even more evident when taking into account that we are considering the financial scenario, where small *leverage opportunities* can raise to large monetary returns.

Let us also underline that a lot of other time series properties can be calculated in a rolling fashion and then used the same way as auxiliary losses.

For example, we can forecast rolling autocorrelation of time series or even parameters of ARIMA or GARCH models to make a connection with classical time series analysis theory.

We also can use this data as inputs to the neural network. Nevertheless, following this approach, we might want to use deeper models, more data and more time to train the models, hence contradicting our primary goal.

In future research we aim to better study the aforementioned path, which seems to be highly related to the two mathematical fields of representation theory and optimization theory.

In particular, from the representation theory point of view, we need to remember that we add auxiliary losses because we want our neural network to incorporate several properties that could not be learnt *automatically*, namely by a straight learning

	Precision	Recall	F1 score
DOWN	0.50	0.47	0.48
UP	0.58	0.62	0.60

Table 5: *Precision / recall / F1 score for a network with a main loss + kurtosis prediction loss*

	MSE	MAE	R^2	MCC	Loss λ_i
Mean aux.	0.0059	0.0564	-0.4199	0.1220	0.9474
Volat. aux.	0.0062	0.0587	-0.4208	0.0929	0.2590
Skew. aux.	0.0058	0.0551	-0.3711	0.1310	0.1482
Kurt. aux.	0.0092	0.0721	-1.2131	0.0837	0.027

Table 6: *Metrics for single auxiliary tasks experiments*

approach on raw data.

While, to what concerns optimization theory, we can use its techniques to consider previously mentioned additional tasks as, e.g., regularizers, penalty constraints, multicriteria optimization approaches, leading to totally different research paths.

VII. Conclusion

In this paper we developed and evaluated multitask approach to financial time series forecasting. We defined statistical moments of the first four orders as auxiliary tasks that predictive model has to optimize alongside the main one - price change prediction.

As the benchmark for evaluation we have chosen Ethereum cryptocurrency prices time series, solving price change in percentage regression and considering related regression and direction forecasting as well.

Our results show that rolling volatility and rolling skewness are the best auxiliary objectives to forecast, being able to obtain directional accuracy better up to 5%. We also used Bayesian optimization to find the best regularization parameters to these tasks, obtaining values equal to 0.25, and 0.15, respectively.

This shows that volatility impact is rather higher than skewness effect. We also analysed how additional tasks affect weights distribution, but without noticing any significant difference. Hence, we can conclude that multitask regularization acts differently from known L1/L2 penalty regularizers.

In our future works we plan to test the aforementioned approach considering heterogeneous datasets, hence without limiting the analysis to consider only financial time series.

Moreover, we aim at discover the impact due to different auxiliary tasks drawn from statistical models, as, e.g., in the case of the ARIMA, GARCH models, as well as w.r.t. other rolling properties, as, e.g., rolling autocorrelation, rolling slope, etc., by studying them analogously to what can be done with the Kalman filter.

	Precision	Recall	F1 score
DOWN	0.42	0.37	0.39
UP	0.53	0.55	0.55

Table 7: *Precision / recall / F1 score for a network with a all losses together*

Last but not least, we are going to research other paths defined in Chapter VI, even if we would like to underline that the presented method based on regularization with statistical moments auxiliary tasks, looks as one of the promising approaches in view of improving machine learning models without overparametrization and/or additional data.

References

[1] Masaya Abe and Hideki Nakayama. *Deep Learning for Forecasting Stock Returns in the Cross-Section*

[2] Carter, C.K., Kohn, R., *On Gibbs sampling for state space models*, Biometrika, 81 (3), pp. 541-553, 1994.

[3] Chen, R., Liu, J.S., *Mixture Kalman filters*, Journal of the Royal Statistical Society, 2000. Series B: Statistical Methodology, 62 (3), pp. 493-508.

[4] Di Persio, L., Cecchin, A., Cordoni, F., *Novel approaches to the energy load unbalance forecasting in the Italian electricity market* , Journal of Mathematics in Industry, 7 (1), art. no. 5, 2017.

[5] Di Persio, L., Perin, I., *An ambit stochastic approach to pricing electricity forward contracts: The case of the German Energy Market* , Journal of Probability and Statistics, 2015, art. no. 626020, 2015.

[6] Désir, C., Bernard, S., Petitjean, C., Heutte, L., *One class random forests*, Pattern Recognition, 46 (12), pp. 3490-3506, 2013.

[7] Di Persio, L., Honchar, O., *Analysis of recurrent neural networks for short-term energy load forecasting* AIP Conference Proceedings, 1906, art. no. 190006, 2017.

[8] Di Persio, L., Honchar, O., *Artificial neural networks architectures for stock price prediction: Comparisons and applications* International Journal of Circuits, Systems and Signal Processing, 10, pp. 403-413, 2016.

[9] Thomas R. Cook and Aaron Smalter Hall. *Macroeconomic Indicator Forecasting with Deep Neural Networks*, Federal Reserve Bank of Kansas City Working Paper No. 17-11

[10] Ian Dewancker, Michael McCourt. *A Stratified Analysis of Bayesian Optimization Methods*

[11] Anders Krogh, Jesper Vedelsby. *Neural network ensembles, cross validation, and active learning.*, Advances in neural information processing systems, 1995.

[12] Ting, Kai Ming, *Encyclopedia of machine learning*, Springer, 2011.

[13] Liew, Jim Kyung-Soo and Mayster, Boris. *Forecasting ETFs with Machine Learning Algorithms*

[14] Matthew F Dixon, *A High Frequency Trade Execution Model for Supervised Learning*

[15] Xin-Yao Qian. *Financial Series Prediction: Comparison Between Precision of Time Series Models and Machine Learning Methods*

[16] SAAHIL MADGE. *Predicting Stock Price Direction using Support Vector Machines* Independent Work Report Spring 2015, Princeton University
[17] JIAO, Y., MA, C., SCOTTI, S., SGARRA, C., *A branching process approach to power markets*, Energy Economics 2018.
[18] SHWETA TIWARI , REKHA PANDIT, VINEET RICHHARIYA. *Predicting Stock Price Direction using Support Vector Machines* International Journal of Electronics and Computer Science Engineering 1578
[19] XIN-YAO QIAN. *Financial Series Prediction: Comparison Between Precision of Time Series Models and Machine Learning Methods*
[20] SASHANK J. REDDI, SATYEN KALE, SANJIV KUMAR. *On the Convergence of Adam and Beyond*, ICLR 2018 Conference Blind Submission
[21] SHUMWAY, R.H., STOFFER, D.S., *AN APPROACH TO TIME SERIES SMOOTHING AND FORECASTING USING THE EM ALGORITHM*, Journal of Time Series Analysis, 3 (4), pp. 253-264, 1982.
[22] MAKRAM ZAIDI. *Forecasting stock market trends with logistic regression and neural networks* International Journal of Economics, Commerce and Management United Kingdom Vol. IV, Issue 6, June 2016
[23] ZHANG, G. P. *Neural networks for classification: a survey*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 30(4), 451-462, 2000.
[24] ZHANG, Q., YANG, L.T., CHEN, Z., LI, P. *A survey on deep learning for big data* Information Fusion, 42, pp. 146-157, 2018.