Neural networks for cryptocurrency evaluation and price fluctuation forecasting

Emmanouil Christoforou[1,2], Ioannis Z. Emiris[1,3], and Apostolos Florakis[1]

[1] Department of Informatics & Telecommunications
National & Kapodistrian University of Athens, Greece
{echristo,emiris,sdi1400217}@di.uoa.gr
[2] Pythagoras Systems, Athens, Greece
[3] ATHENA Research Center, Maroussi, Greece

Abstract. Today, there is a growing number of digital assets, often built on questionable technical foundations. We design and implement supervized learning models in order to explore different aspects of a cryptocurrency affecting its performance, its stability as well as its daily price fluctuation. One characteristic feature of our approach is that we aim at a holistic view that would integrate all available information: First, financial information, including market capitalization and historical daily prices. Second, features related to the underlying blockchain from blockchain explorers like network activity: blockchains handle the supply and demand of a cryptocurrency. Lastly, we integrate software development metrics based on GitHub activity by the supporting team. We set two goals. First, to classify a given cryptocurrency by its performance, where stability and price increase are the positive features. Second, to forecast daily price tendency through regression; this is of course a well-studied problem. A related third goal is to determine the most relevant features for such analysis. We compare various neural networks using most of the widely traded digital currencies (e.g. Bitcoin, Ethereum and Litecoin) in both classification and regression settings. Simple Feedforward neural networks are considered, as well as Recurrent neural networks (RNN) along with their improvements, namely Long Short-Term Memory and Gated Recurrent Units. The results of our comparative analysis indicate that RNNs provide the most promising results.

Keywords: Cryptocurrency · Deep learning · Neural network · Blockchain · Price variation prediction · Coin features · Feature importance

1 Introduction

Digital cipher currencies based on blockchains as introduced by Nakamoto's specification in 2008 [13] are exponentially growing in number, during the last few years [7], thus affecting significantly the global financial and trading scene. Today, there is a large number of different cryptocurrencies (more than 1,600). Blockchain is primarily responsible for most of the advantages of cryptocurrencies over fiat currencies, such as decentralization and anonymity. The increasing

interest around blockchain-based currencies underlines the importance of methods to evaluate them and forecast their price tendencies. Our paper focuses on artificial neural networks, but first surveys related previous work.

There are several existing works for stock price forecasting with time series prediction methods, see e.g. [2], and for stock-market crisis forecasting using either deep and statistical machine learning, e.g. [5], or computing and manipulating copulas [4]. Moreover, neural networks have already been used to forecast stock and cryptocurrency price fluctuations in several works: in [12] they examined the accuracy of neural networks for the prediction of the direction of Bitcoin price (in USD) using a Bayesian optimized Recurrent neural network and a Long Short-Term Memory (LSTM) network, against the Autoregressive integrated moving average (ARIMA) model, with the LSTM having the highest classification accuracy (52% accuracy and 8% root mean squared error). Neural networks have been designed to examine whether chaoticity is inherently improving short-term predictability of cryptocurrencies [11]. Several technical indicators were included in deep learning architectures in order to predict the future return trend of Bitcoin [14], and in hybrid neural networks with generalized auto-regressive conditional heteroskedasticity (ANN-GARCH) to forecast Bitcoin's price volatility [10]. Blockchain features are explored as input to neural networks that may explain Bitcoin's price hikes using several machine learning methods [15], and as input to a Bayesian neural network to model and predict the Bitcoin price [9].

The main contribution of this work is to explore cryptocurrencies with neural networks in order to rank them as positive or negative, based on stability or price increase, as well as to capture their daily price tendency through regression. A related goal is to determine the most relevant features for such analysis. Blockchain-based currencies, are more accurately investigated by considering all of their relevant aspects namely financial, including market capitalization and historical daily prices, blockchain-related features, such as network activity, as well as software development metrics based on GitHub activity. Various neural networks are designed, trained, validated and experimentally compared looking at their accuracy and using most of the widely traded digital currencies. Simple Feedforward neural networks are considered, as well as Recurrent neural networks (RNN) along with their improvements, namely Long short-term memory (LSTM) and Gated recurrent units (GRU). The results of our comparative analysis are briefly reported in Tables 3, 4 and indicate that RNNs provide the most promising results.

The rest of the paper is organized as follows. Sect. 2 outlines our data, their processing and the neural networks that are compared. Sect. 3 presents the results of the neural networks in classifying cryptocurrencies and forecasting price fluctuations. Sect. 4 offers a discussion of results, and future work.

2 Methods

In this section, we outline the data and their characteristics. We then outline the high-level theme of our analysis, the generation of the labels used for training, and the neural networks used.

2.1 Cryptocurrency features

To describe efficiently daily price variations we consider several aspects.

Most of the existing approaches process time-series of prices and technical indicators; similarly this work also relies on historical daily prices. These prices consist of the open, high, low, and close (OHLC) prices, volume and market capitalization (Market Cap), all at a daily level.

Features representing technological aspects of a blockchain, at a daily level, are included to reflect price variations, considering that the blockchain records all transactions of a cryptocurrency. These data are collected from blockexplorers: platforms that allow search and navigation through the blocks of a blockchain, in order to produce several statistics. Some of these blockchain features reflect the daily number of blocks that were on the chain (block count), the number of bytes broadcast in final blocks (block size) and the difficulty level of the hash function to find a new block (average difficulty). Other features track the volume (in USD) that circulates on the blockchain per day (on-chain transaction volume), the number of transactions on the blockchain (transaction count), the USD value of the volume at cryptocurrency exchanges (exchange volume), and the number of new coins generated (generated coins). Finally, there are also features for the number of unique addresses used on the blockchain per day (active addresses), the total amount of fees (fees) and their median value (median fee), and the realized capitalization (realized cap), a metric that attempts to improve the market cap by counting the price when each coin lastly moved through the blockchain, instead of counting all of the mined coins at the last market price.

In addition, other features describing the development activity, such as the commit count in a four-week interval, and the popularity of a cryptocurrency (number of stars, forks, subscribers and contributors) are also considered. These features are collected from the git repositories (most importantly, GitHub) of each cryptocurrency.

These 28 features (Table 1) are selected in order to identify those that affect or describe the price variations and they are collected from several websites such as CoinMarketCap (OHLCV prices), Coin Metrics (blockexplorer features) and CoinGecko (Git features)[4]. All features are normalised feature-wise so that all inputs lie in $[0, 1]$.

For classifying the cryptocurrencies one needs to generate appropriate labels: Each daily feature vector is labeled as "positive" or "negative", based on the variation of the closing price (p_t) in comparison to the mean price of the previous

[4] https://coinmetrics.io, https://www.coingecko.com/en, https://coinmarketcap.com

Table 1. Features collected for cryptocurrencies (financial features are in USD).

Scope	**Features**
Financial	open, high, low, close, volume, market capitalization
Blockchain explorers	#active addresses, adjusted transaction volume, average difficulty, block count, block size, exchange volume, fees, #generated coins, median fee, median transaction value, payment count, realized cap, transaction count, on-chain transaction volume
Developer (Git)	#closed issues, #total issues, commit count (4 weeks frequency), #forks, #pull request contributors, #pull requests merged, #stars, #subscribers

30 days. When p_t is at least as large as 99% μ_{p_t}, namely

$$p_t \geq \mu_{p_t} - 1\% \, \mu_{p_t}, \quad \mu_{p_t} = \frac{1}{30} \sum_{i=-30}^{-1} p_{t-i},$$

we label the cryptocurrency as positive. Otherwise, it is negative, resulting to a dataset with 1035 positive and 760 negative instances for Bitcoin (BTC) in 2014-2018 (Fig. 1). We expect the neural network to identify those features that affect the price fluctuations, covering most aspects of a cryptocurrency (financial, blockchain, development). For the regression task, we aim to forecast price fluctuations, and set as target the closing price of the following day.

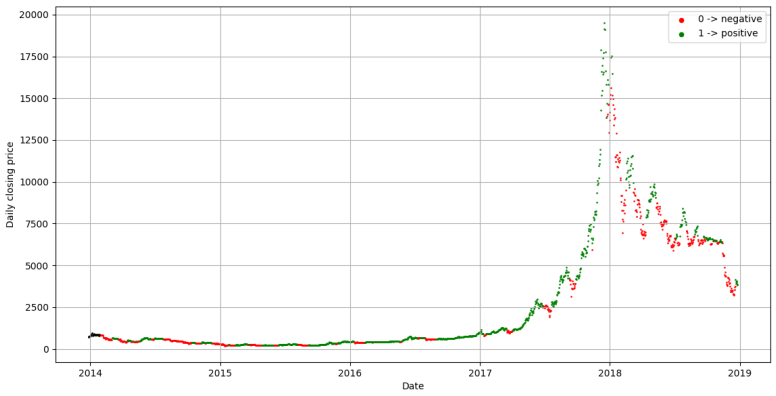

Fig. 1. BTC daily closing prices in 2014-18 labeled positive (green) or negative (red).

2.2 Neural Networks

Neural networks (NNs) are composed of an input layer, followed by an arbitrary number of hidden layers, and an output layer that makes the final decision or prediction. Trained on a set of input-output pairs, they model the correlation (or dependencies) between those inputs and outputs. A neural network is employed in two phases: The forward pass where the input signal flows from input through hidden layers, to the output layer. The latter is evaluated by the ground truth labels or values. Then, a backward pass follows, where the network parameters are back-propagated: the network weights and biases are adjusted in order to minimize error, using gradient-based optimization. The two passes are repeated until the loss does not reduce (convergence). Neural networks are distinguished in Feedforward neural networks (FNNs) and RNNs, based on whether they allow cyclic connections between nodes or not. Using cyclic connections, RNNs use internal state (memory) to process sequential data, such as time series.

Data points indexed by time may be processed as time series. FNNs are able to handle data only in a unified way, thus ignoring any underlying time-dependencies between their time steps. As a result, it becomes difficult for the network to identify hidden patterns that are related to time-dependencies. On the other hand, RNNs learn conditional dependencies between sequence elements during learning. RNNs process separately each step of the time series keeping a memory mechanism about the whole processing of the series. The content of this memory unit is used while processing each timestep. Since our data in this work are daily features for each coin, they can be processed both as independent instances in FNNs, as well as time series in RNNs.

Feedforward neural networks (FNNs). Deep FNNs, also known as multi-layer perceptrons (MLPs), are neural networks with undirectional information flow, meaning that there are no cycles or feedback connections to feed back the output into the network. The basic idea behind an FNN is the perceptron (neuron), a linear classifier, that separates input into two categories by a straight line. An MLP is composed of more than one perceptrons placed in a single-direction (forward) graph. In this work we tested several such networks with an input layer, one to three hidden layers, and an output layer.

Recurrent neural networks (RNNs). For sequential data, where input is interdependent, FNNs appear to be inefficient. RNNs allow cyclic connections, fitting better to dynamic processes, such as time series. These models are able to represent the relation between previous input-output pairs, since every new output is a function of the previous one. RNNs process one example at a time, retaining memory with contextual information, to be reused at the next time step. This recurrent formulation allows the RNN to share the same weights across several time steps. In theory, classic ("vanilla") RNNs can keep track of arbitrarily long-term dependencies in the input, but suffer from computational issues. During training, the back-propagated gradients may "vanish" (tend to

zero) or "explode" (tend to infinity), because of the computations using finite-precision arithmetic. Therefore, a very deep computational graph of an RNN is unlikely to understand long-term dependencies. In order to cope with long-term dependency difficulties, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) architectures have been proposed. The idea behind LSTM and GRU units is to create connections through time with a constant error flow, thus the gradient neither explodes nor vanishes.

Long Short-Term Memory (LSTM). LSTM [8] is an RNN architecture. They were developed to deal with the vanishing gradient problems of traditional RNNs, providing a robust extension. LSTMs are explicitly designed to avoid long-term dependency problems. Their default behavior is to remember information for long time periods. This is why they are one of the most popular NNs for sequence learning, allowing gradients to flow unchanged, while preserving previous information. LSTM not only keep adjacent temporal information on a spontaneous manner, but also control long-term information introducing the notion of "controlling gate". A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the 3 gates regulate the flow of information into and out of the cell. Initially, the forget-gate extracts the amount of information that should be preserved from the prior state. Then, the input gate determines how much "current" information should be used as input in order to generate the current state. The output gate filters the information deemed significant. This procedure is repeated in every time step of sequential processing, allowing LSTM memory to remember or forget cell states.

Gated Recurrent Units (GRU). GRUs [6] are gating mechanisms in RNNs, similar to LSTMs with forget-gate, but have fewer parameters, as they lack an output gate. GRUs have been shown to exhibit better performance on certain smaller datasets. They can be trained to remember information from long ago, without washing it through time, and to remove information which is irrelevant to the prediction by deciding what should be passed to the output. A GRU is composed of a cell, an update gate and a reset gate. The update-gate determines the previous time steps that need to be passed along to the future, while the reset-gate decides the past information to forget.

In our work we tested both of the above cell types. Sequences are fed as input to the RNNs in order to identify the underlying patterns. The predicted price or class is regarded as the output of our network.

3 Results

In this section, we outline the metrics used to evaluate our results both for classification and regression, while we outline the employed architectures. A summary of results is in Tables 3, 4 with training and testing times, and the coins that were

used. Our NNs were developed using the open-source software library TensorFlow [1]. Training was accomplished within minutes using the GPU accelerated environment of Google's Colaboratory.

3.1 Classification

True positives (TP) and true negatives (TN) are the outcomes where the model predicted the correct class (positive or negative correspondingly). Incorrect outcomes are defined as false positives (FP) and false negatives (FN). To evaluate the results of the classifiers we use the following metrics:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \quad \text{Prec} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

where Accuracy (Acc) indicates the total proportion of correct predictions among the total number of cases examined; Precision (Prec) is the fraction of true relevant instances predicted in a class; Recall (Rec) is the fraction of true relevant instances predicted over the total amount of relevant instances.

FNN. An FNN is trained with historical instances of our dataset for Bitcoin from 14/06/2014 to 24/08/2017 and is tested using more recent instances (25/08/2017–21/12/2018). Using these instances as our test set, we want to investigate whether a plain NN is able to correctly classify them, identifying the underlying patterns that characterize the price trend of a cryptocurrency. A NN with three hidden layers of 50 nodes has Acc: 64% (while Prec: 56%, Rec: 76%).

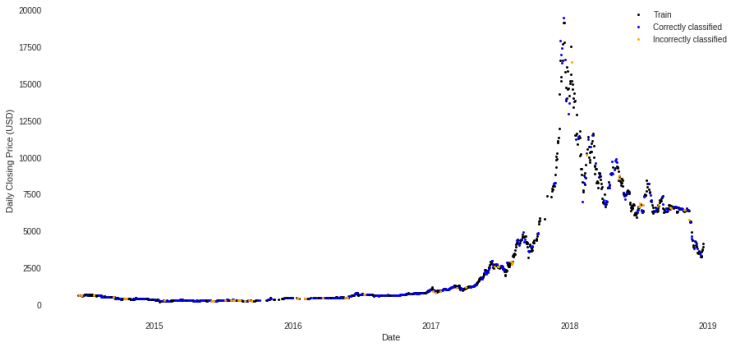

Fig. 2. Correctly (blue) and incorrectly (yellow) classified instances against closing price (BTC, 14/06/2014 to 21/12/2018), by an FNN with 3 hidden layers of 50 nodes (Acc: 84%, Prec: 86%, Rec: 89%). Black dots are training instances.

Selecting the test set randomly from the whole sample and training an FNN with three hidden layers of 50 nodes, results to Acc: 84% (Prec: 86%, Rec: 89%),

which is expected since the neural network is trained with samples through the whole history of the coin. Therefore, it is trained to understand most of its spectrum, classifying most of the instances correctly (Fig. 2).

Another approach is to divide every day's features by the corresponding value of the previous day, so as to represent the daily relevant change. Thus, we allow the NN to learn directly the rate at which features change, instead of their absolute values or differences between them. An FNN with one hidden layer of 100 nodes was trained with historical values for BTC (15/06/2014-09/12/2017) and tested using values on 10/12/2017-21/12/2018. It achieved Acc: 70%, Prec: 57%, Rec: 74% (Fig. 3). In Fig. 4 we observe the resulting labels on the test set, which are close to the original (Fig. 1).

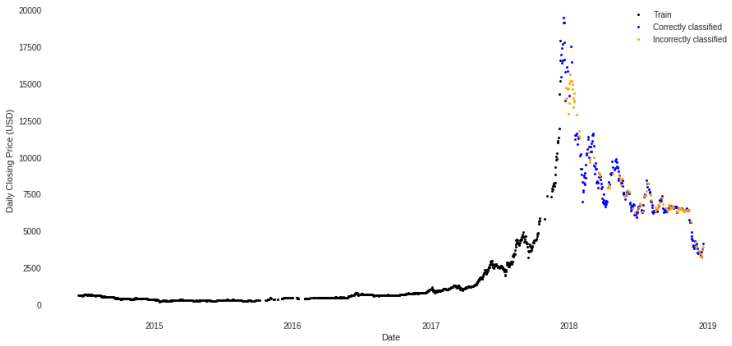

Fig. 3. Correctly (blue) and incorrectly (yellow) classified instances against closing price (BTC, 10/12/2017 to 21/12/2018), by an FNN with one hidden layer of 100 nodes (Acc: 70%, Prec: 57%, Rec: 74%). Black are training instances.

RNN. Using the same features, we train a single layer RNN with a GRU cell of 100 nodes, where the input is time series of a 4-day time window (for each instance the previous 4 days are used). The first 70% of the dataset is used for training and the rest 30% for testing, achieving Acc: 71%, Prec: 67%, Rec: 71%.Also, a single layer RNN with an LSTM cell of 128 nodes and 10-days time window has been trained with data from BTC, Ethereum (ETH) and Litecoin (LTC) and was tested with a different cryptocurrency, namely Dash (DASH) coin instances. The results were Acc: 64%, Prec: 49%, Rec: 82%. It is very encouraging to notice that the performance is comparable to models tested on the same coin as the one used at training.

The previous RNN approaches exploit only past information for every instance. In order to make use of past and future states for each instance in a 4-day time window, we apply Bidirectional RNN (BRNN) [16]. The BRNN here consists of two stacked GRUs of 80 nodes with opposite directions (positive and negative time direction) to the same output. This enables the use of informa-

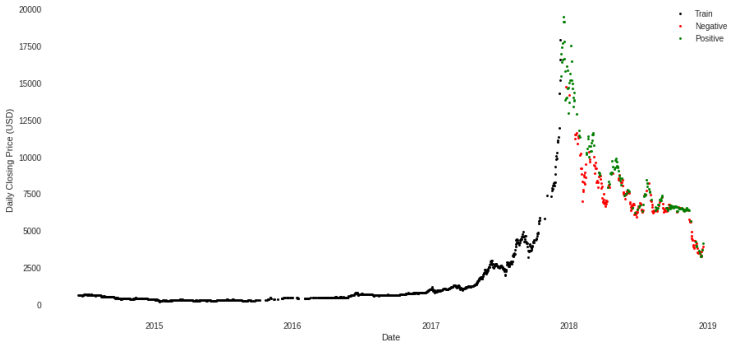

Fig. 4. Resulting labels (BTC, 10/12/2017 to 21/12/2018) by an FNN with one hidden layer of 100 nodes (Acc: 70%, Prec: 57%, Rec: 74%).

tion from past and future states simultaneously. Hence, the performance may be enhanced by the prices tendencies located before and after every price in the specified window. In Fig. 5 the correctly and incorrectly classified instances are plotted, with Acc: 72%, Prec: 70%, Rec: 65%.

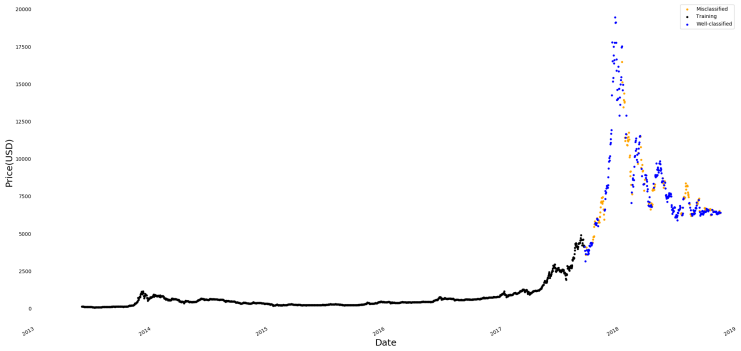

Fig. 5. Correctly (blue) and incorrectly (yellow) classified instances against the closing price of BTC, by BRNN with two GRU cell of 80 nodes (Acc: 72%, Prec: 70%, Rec: 65%). Black are training instances.

FNN and RNN with single coin. Combining some of the previously mentioned layers we are able to build more effective neural networks. Here, we design a network with four hidden layers: a fully-connected layer of 64 nodes, a BRNN with LSTM layers of 128 nodes, followed by another LSTM layer of 128 nodes and a fully-connected layer with 64 nodes. As input we use the daily relevant change of each feature, as described in previous experiment. The network was

trained using 10-day time-step window for BTC instances (15/06/2014 until 06/04/2018). The results on the test set (BTC 07/04/2018 - 06/04/2019) indicate that this network outperforms the previous experiments with Acc: 78% (Prec: 72%, Rec: 90%) (Fig. 6). Therefore, we use this model to explore the importance of each feature.

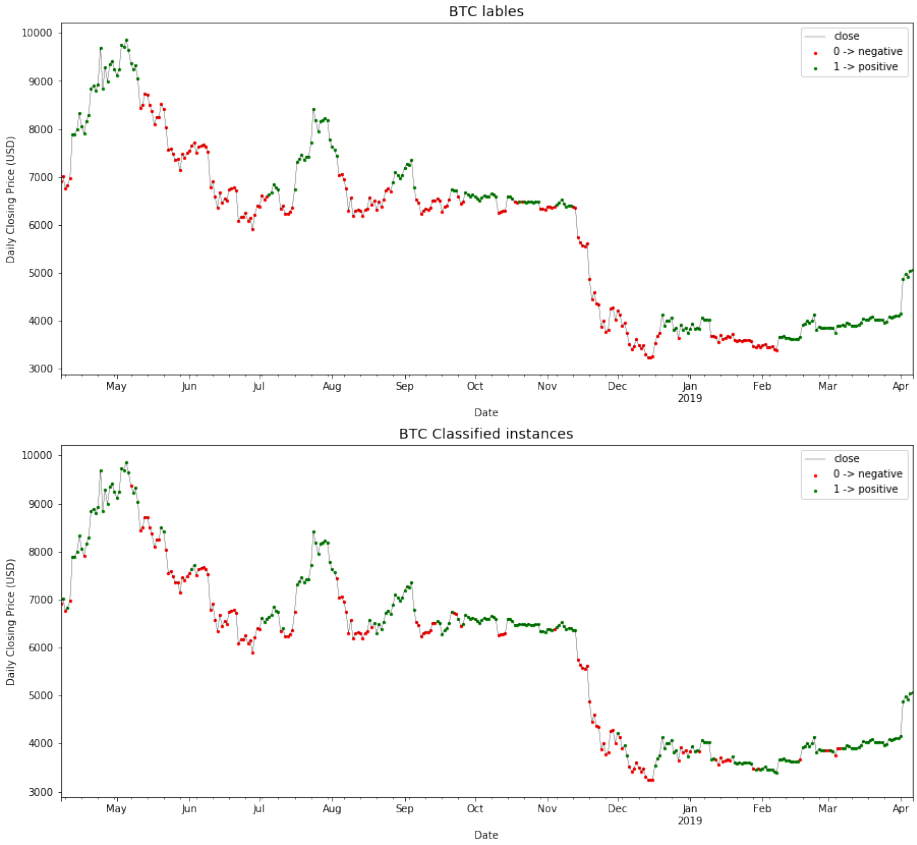

Fig. 6. Resulting labels for BTC, from 07/04/2018 to 06/04/2019, by an NN with four hidden layers (a fully-connected layer, a BRNN with LSTM layers, a LSTM layer and a fully-connected layer (Acc: 78%, Prec: 72%, Rec: 90%).

To identify the most important features in this work we use techniques such as permutation importance. Specifically, after training, we distort one of the features and we evaluate the accuracy of the model on the resulting test set. If the accuracy diverges significantly from our baseline, which is the accuracy of the same model on the original test set (Acc: 78%, Fig. 6), we conclude the corresponding feature is important. Repeating the same procedure for every feature allows us to distinguish those that appear to be more important.

We set two distortion test cases: (i) replace all values of a feature with a single value, indicating the feature does not change through time (called "Repeating-value importance"), and (ii) randomly shuffle all values of a feature ("Permutation importance"). The results are in Table 5. We report the accuracy of the model per feature, and the MAPE and MSE from the baseline accuracy, sorted by the MSE of Permutation importance. For Permutation importance, every experiment was performed 5 times, for accuracy, and the mean values are reported. MSE is used in order to take into account any outliers, especially for Permutation importance. Higher MAPE and MSE indicate the most important features. As we move to the top we find mostly financial and blockchain features, as expected.

We apply the same methods for the different groups of features (Table 1). Again, as expected, results indicate that the most important scopes are the financial and blockchain ones (Table 6).

FNNs and RNNs with multiple coins. A similar NN to previous experiment, but with more nodes in each hidden layer, is trained for totally 73 coins. The NN consists of a fully-connected layer of 128 nodes, a BRNN with LSTM layers of 256 nodes, followed by another LSTM layer of 256 nodes and a final fully-connected layer with 128 nodes. We use the daily relevant change of each feature for the 73 coins as input. The features used are a subset of the financial and blockexplorer features. Namely, the features are: all the financial scope and the active addresses, exchange volume, fees, median fee, median transaction value, transaction count and transaction volume. The network was trained with time series of 10-days window and for test set we keep the last year for all the coins that is available. Thus, we get a dataset with 31546 train and 25667 test instances. The results were Acc: 63%, Prec: 58%, Rec: 59%. Even though the overall accuracy is relatively low, it is very encouraging to notice that the performance on some specific coins (bold in Table 2) appears to be comparable to previous experiments.

Table 2. Results of FNN and RNN with multiple coins. Each coin is reported with the accuracy of the model on its last year's instances.

Coin (Acc)
ADA (52.3%), AE (57.8%), AION (68.8%), ANT (68.5%), BAT (68.8%), BCH (67.7%), BNB (67.1%), **BTC (77.8%)**, BTG (60.3%), BTM (57.3%), **CENNZ (72.4%)**, CTXC (67.1%), CVC (56.7%), **DAI (83.6%)**, DASH (58.9%), DCR (61.4%), DGB (65.5%), DOGE (67.7%), DRGN (60.5%), ELF (63.0%), ENG (61.4%), EOS (64.7%), ETC (65.2%), ETH (59.5%), ETHOS (65.2%), FUN (59.2%), GAS (66.0%), GNO (61.1%), GNT (65.8%), GUSD (50.5%), ICN (57.3%), ICX (59.5%), KCS (45.8%), KNC (65.8%), LOOM (46.2%), LRC (54.8%), LSK (65.2%), LTC (67.7%), MAID (62.5%), **MANA (71.5%)**, MTL (62.7%), NAS (68.5%), NEO (54.5%), OMG (60.5%), **PAX (72.7%)**, PAY (55.6%), PIVX (57.5%), POLY (60.8%), POWR (58.1%), PPT (58.6%), QASH (59.7%), REP (68.2%), RHOC (66.6%), SALT (69.6%), SNT (61.6%), SRN (61.6%), TRX (59.5%), TUSD (54.5%), **USDC (84.6%)**, **USDT (89.0%)**, **VERI (72.1%)**, VET (56.2%), VTC (67.9%), WAVES (51.2%), WTC (61.9%), XEM (51.0%), XLM (69.0%), XMR (66.0%), **XRP (73.2%)**, XVG (44.1%), ZEC (55.1%), ZIL (63.6%), ZRX (65.2%)

3.2 Regression

In order to forecast price fluctuations we employ regression by NNs. The criteria are mean squared error (MSE) and mean absolute percentage error (MAPE) over n instances. MSE equals the average squared difference between predicted F_t and true A_t values for $t = 1, \ldots, n$, while MAPE measures the percentage (relative) difference between predicted and true values, as follows.

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(A_t - F_t)^2, \qquad \text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|.$$

RNN. We design an NN with two hidden layers of 128 nodes. The first layer is an LSTM with layer normalization [3] and recurrent dropout [17] followed by a simple LSTM layer. The former normalizes layer output and is quite effective at stabilizing the hidden state dynamics. The recurrent dropout is a technique used to avoid overfitting and improve results by dropping some neurons using a prescribed keep-probability. Here we train the NN, with 80% keep probability for the dropout layer, using BTC's financial scope (OHLCV and Market Cap) and the results of an Autoregressive integrated moving average (ARIMA) model (total of 7 features for each instance). ARIMA are commonly used models for time series forecasting and their results here are used to improve the fluctuation forecasting. The parameters of the ARIMA model were selected using an auto ARIMA implementation (lag order: 5, degree of differencing: 1, order of moving average model: 0).

Training uses a 3-day time-step window per instance, using BTC prices from 01/01/2015 to 04/01/2018. The test set contains BTC prices from 05/01/2018 to 27/01/2019, with total MSE: 0.00119, MAPE: 7.28%. In Fig. 7 we observe the similarity of forecast against the actual daily closing price fluctuation.

Table 3. Results summary for models with normal input.

NN	Train	Test	Train time (sec)	Test time (sec)	Accuracy
FNN	BTC	BTC	10.4	0.65	63.6%
GRU	BTC	BTC	8.51	0.75	71.0%
BRNN	BTC	BTC	9.85	0.81	72.0%
LSTM	BTC+LTC+ETH	DASH	80.0	0.81	64.0%

4 Discussion and Future work

We have presented a comparative analysis of NNs to classify cryptocurrencies and forecast their price fluctuations. We have included several features to cover most of their aspects and evaluated their relevance. Besides daily prices (open, high, low, close prices, volume and market cap) and blockchain features (number

Table 4. Results summary for models with the daily difference of each feature (from previous day) as input.

NN	Train	Test	Train time (sec)	Test time (sec)	Accuracy
FNN	BTC	BTC	9.3	0.63	70.3%
FNN + RNN	BTC	BTC	194	0.48	78%
FNN + RNN	73 coins	73 coins	1300	24	63%

Fig. 7. BTC price forecasting from 05/01/2018 to 27/01/2019 (red) vs actual closing price normalized (green).

of transactions, blocks, active addresses, fees, etc), we have also used features to describe the development and software code popularity and penetration into the community (stars, subscribers, forks, commit counts, etc).

The classification of each instance as positive or negative, based on the daily change of features relative to their previous value, seems to provide good results, since the accuracy on last year's instances is 78% by a NN with a fully-connected layer of 64 nodes, a BRNN with LSTM layers of 128 nodes, followed by a LSTM layer of 128 nodes and a fully-connected layer with 64 nodes. Positive and negative instances are those where daily closing prices are increasing or decreasing respectively, compared to the previous days. Therefore, its direct purpose is not to predict, but to evaluate the current snapshot of a coin.

Concerning prediction, the regression results of an RNN with two LSTM cells of 128 nodes, are quite promising in forecasting price fluctuations with total error of 7.28% from actual price, using only daily prices and an ARIMA prediction. Integrating the remaining features to a more complex architecture should provide more accurate forecasts.

Even though the NNs used were fed with raw features from different scopes, results already appear to be promising. New features shall be considered e.g. technical indicators like moving average convergence divergence (MACD), relative strength index (RSI) as well as sentiment analysis. The combination of features from various widely traded cryptocurrencies to generate a larger dataset, would be suitable for deep learning methods. More powerful, deep neural network architectures shall be evaluated. Using several layers, and state-of-the-art deep learning, should enhance forecasting power. For this, Autoencoders and Generative Adversarial Networks (GANs) shall be considered.

Acknowledgments. Ioannis Emiris is member of joint team AROMATH between INRIA Sophia-Antipolis (France) and NKUA. The first two authors are partially supported by the European Union's H2020 research and innovation programme under grant agreement No 734242 (LAMBDA).

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., et al.: Tensorflow: A system for large-scale machine learning. In: Proc. USENIX Symp. Operating Systems Design & Implement. pp. 265–283 (2016)
2. Ariyo, A., Adewumi, A., Ayo, C.: Stock price prediction using the ARIMA model. In: Proc. UKSim-AMSS Inter. Conf. Comp. Modeling & Simul. pp. 106–112 (2014)
3. Ba, J., Kiros, R., Hinton, G.E.: Layer normalization. Tech. Rep. 1607, arXiv (2016)
4. Calès, L., Chalkis, A., Emiris, I.Z., Fisikopoulos, V.: Practical volume computation of structured convex bodies, and an application to modeling portfolio dependencies and financial crises. In: Proc. Inter. Symp. Comput. Geom., Budapest. pp. 19:1–19:15 (2018)
5. Chatzis, S.P., Siakoulis, V., Petropoulos, A., Stavroulakis, E., Vlachogiannakis, N.: Forecasting stock market crisis events using deep and statistical machine learning techniques. Expert Systems with Appl. **112**, 353–371 (2018)

Table 5. Feature importance. Results are sorted by permutation importance MSE.

	Permutation importance			Repeating value importance		
Feature	**Acc(%)**	**MAPE(%)**	**MSE**	**Acc(%)**	**MAPE(%)**	**MSE**
realized cap	70.8	8.9	0.005397	76.2	2.1	0.000270
close	74.3	4.5	0.001853	74.8	3.9	0.000908
low	79.3	3.3	0.001118	79.7	2.5	0.000368
exchange volume	77.2	3.7	0.001031	78.1	0.4	0.000008
market cap	78.2	3.4	0.000868	77.5	0.4	0.000008
average difficulty	80.2	3.1	0.000688	78.9	1.4	0.000120
transaction count	76.4	2.9	0.000659	77.8	0.0	0.000000
payment count	80.1	3.0	0.000561	80.0	2.8	0.000480
adjusted transaction volume	79.6	2.6	0.000542	79.2	1.8	0.000188
high	75.7	2.7	0.000525	75.3	3.2	0.000608
open	77.2	2.2	0.000524	78.4	0.7	0.000030
active addresses	76.7	2.4	0.000426	77.3	0.7	0.000030
fees	79.5	2.1	0.000315	80.5	3.5	0.000751
block size	76.4	1.8	0.000246	75.9	2.5	0.000368
pull requests merged	77.8	1.7	0.000228	79.5	2.1	0.000270
total issues	77.9	1.1	0.000195	77.8	0.0	0.000000
forks	79.1	1.6	0.000179	78.4	0.7	0.000030
pull request contributors	78.8	1.3	0.000167	78.4	0.7	0.000030
commit count (4 weeks)	78.5	1.3	0.000161	79.5	2.1	0.000270
block count	78.1	1.3	0.000161	78.4	0.7	0.000030
stars	78.7	1.2	0.000101	79.5	2.1	0.000270
subscribers	78.6	1.1	0.000096	78.6	1.1	0.000068
transaction volume	77.5	1.1	0.000092	76.4	1.8	0.000188
volume	77.5	0.8	0.000075	78.9	1.4	0.000120
generated coins	78.3	0.6	0.000050	77.0	1.1	0.000068
median fee	78.0	0.6	0.000029	78.4	0.7	0.000030
closed issues	78.0	0.2	0.000008	77.8	0.0	0.000000
median transaction value	77.8	0.1	0.000002	77.8	0.0	0.000000

Table 6. Features importance per scope.

	Permutation importance			Repeating value importance		
Scope	**Acc(%)**	**MAPE(%)**	**MSE**	**Acc(%)**	**MAPE(%)**	**MSE**
Financial	57.9	25.6	0.040344	54.0	30.6	0.056814
Blockchain explorers	72.2	7.2	0.003561	75.1	3.5	0.000751
Developer (Git)	78.5	2.0	0.000288	81.9	5.3	0.000751

6. Cho, K., van Merrinboer, B., Gulcehre, C., Bougares, F.e.a.: Learning phrase representations using rnn encoder-decoder for statistical machine translation (Jun 2014). https://doi.org/10.3115/v1/D14-1179
7. Hileman, G., Rauchs, M.: 2017 global cryptocurrency benchmarking study. SSRN Electronic J. (Jan 2017)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–1780 (Dec 1997). https://doi.org/10.1162/neco.1997.9.8.1735
9. Jang, H., Lee, J.: An empirical study on modeling and prediction of bitcoin prices with bayesian neural networks based on blockchain information. IEEE Access **6**, 5427–5437 (2018)
10. Kristjanpoller, W., Minutolo, M.C.: A hybrid volatility forecasting framework integrating GARCH, artificial neural network, technical analysis and principal components analysis. Expert Systems with Appl. **109**, 1–11 (2018)
11. Lahmiri, S., Bekiros, S.: Cryptocurrency forecasting with deep learning chaotic neural networks. Chaos, Solitons & Fractals **118**, 35–40 (Jan 2019)
12. McNally, S., Roche, J., Caton, S.: Predicting the price of bitcoin using machine learning. In: Euromicro Intern. Conf. Parallel, Distributed & Network-based Processing (PDP). pp. 339–343 (2018)
13. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Cryptography Mailing list at https://metzdowd.com (Mar 2009)
14. Nakano, M., Takahashi, A., Takahashi, S.: Bitcoin technical trading with artificial neural network. Physica A: Statistical Mechanics and Appl. **510**, 587–609 (2018)
15. Saad, M., Mohaisen, A.: Towards characterizing blockchain-based cryptocurrencies for highly-accurate predictions. In: Proc. IEEE Conf. Computer Communications (Infocom) Workshops. pp. 704–709 (2018)
16. Schuster, M., K. Paliwal, K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Processing **45**, 2673–2681 (1997). https://doi.org/10.1109/78.650093
17. Semeniuta, S., Severyn, A., Barth, E.: Recurrent dropout without memory loss. In: Proc. COLING Inter. Conf. Comp. Linguistics. pp. 1757–1766 (2016)