# Deep State-Space Model for Predicting Cryptocurrency Price

Shalini Sharma, Angshul Majumdar, Emilie Chouzenoux, Víctor Elvira

**Deep State-Space Model for Predicting Cryptocurrency Price**

Shalini Sharma[a] (shalinis@iiitd.ac.in), Angshul Majumdar[b] (angshul@iiitd.ac.in), Emilie Chouzenoux[c] (emilie.chouzenoux@inria.fr), Víctor Elvira[d] (victor.elvira@ed.ac.uk)

[a] PhD Scholar, Indraprastha Institute of Information Technology-Delhi, India

[b] Associate Professor, Indraprastha Institute of Information Technology-Delhi, India

[c] Research Director, Inria Saclay, University Paris Saclay, France

[d] Professor, School of Mathematics, University of Edinburgh, UK

**Corresponding Author:**

Shalini Sharma

PhD Scholar, Indraprastha Institute of Information Technology-Delhi, India

Email: shalinis@iiitd.ac.in.

# A Deep State-Space Model for Predicting Cryptocurrency Price

Shalini Sharma[a], Angshul Majumdar[b], Emilie Chouzenoux[c], Victor Elvira[d]

[a] *PhD Scholar, Indraprastha Institute of Information Technology-Delhi, India*
[b] *Associate Professor, Indraprastha Institute of Information Technology-Delhi, India*
[c] *Research Director, Inria Saclay, University Paris Saclay, France*
[d] *Professor, School of Mathematics, University of Edinburgh, UK*

**Abstract**

Our work presents two fundamental contributions. On the application side, we tackle the challenging problem of predicting day-ahead crypto-currency prices. On the methodological side, a new dynamical modeling approach is proposed. Our approach keeps the probabilistic formulation of the state-space model, which provides uncertainty quantification on the estimates, and the function approximation ability of deep neural networks. We call the proposed approach the deep state-space model. The experiments are carried out on established cryptocurrencies (obtained from Yahoo Finance). The goal of the work has been to predict the price for the next day. Benchmarking has been done with both state-of-the-art and classical dynamical modeling techniques. Results show that the proposed approach yields the best overall results in terms of accuracy.

*Keywords:* Time series analysis; deep state-space models; deep matrix factorization; Kalman filtering; Bayesian smoothing; EM algorithm;, cryptocurrency forecasting, dynamic recurrent network.

*Corresponding author.
*Email addresses:* shalinis@iiitd.ac.in (Shalini Sharma), angshul@iiitd.ac.in (Angshul Majumdar), emilie.chouzenoux@inria.fr (Emilie Chouzenoux), victor.elvira@ed.ac.uk (Victor Elvira)

## 1. Introduction

Investopedia defines crypto-currency as "a digital or virtual currency that is secured by cryptography, which makes it nearly impossible to counterfeit or double-spend" and is built on "decentralized networks based on block-chain technology—a distributed ledger enforced by a disparate network of computers". A defining feature of crypto-currencies is that they are usually not issued by central banking agencies like the Federal Reserve System in US, Bank of Canada, European Central Bank, or the People's Bank of China; this makes crypto-currencies (theoretically) immune to government interventions.

The introduction of Bitcoin around 2009 and its meteoric rise led to investors infuse their funds in crypto-currencies. One major reason behind the shift in investment largely owes to the 2008 financial crisis, which subsequently led to the waning of trust in the banking system. The market capitalization of crypto-currencies rose from less than 10 billion in 2014 to more than 2 trillion in 2021.

However, crypto-currencies are extremely volatile. To give an example, the volatility index of the most stable crypto-currency USD Tether (USDT) has been between 95 and 100 in August 2021, while that of a blue chip corporation like Microsoft (MSFT) has been around 16 in the same period; a highly volatile smallcap stock like Genworth Financial (GNW) in the same period had a volatility index less than 40. Such large volatility makes predicting crypto-currency prices a more challenging problem than stock forecasting. The reason crypto-currencies are volatile is because they do not have any intrinsic value. Their prices are mainly dependent on the emotion of investors, and in such a scenario, tweets from influencers can play a major role in swaying their prices; one example of how tweets from a major influencer can drive prices high or low can be seen from (R. Molla, 2021; S. Soni, 2021).

This work addresses the most challenging problem in personal finance today – forecasting - prices. There are a few studies on this subject. Recent study (Livieris et al., 2021),(Ye & Dai, 2022) use off-the-shelf deep learning tools for predicting crypto-currency prices. Competitive survey analysis of forecasting

crypto-currency using machine learning method can be found (Derbentsev et al., 2020). A different branch of study (Yasir et al., 2020) follows cues from social media for predicting the crypto-currency and gives more insights for estimating future prices (Kraaijeveld & De Smedt, 2020). One of the study uses ARCH-MIDAS framework to identify drivers of Crypto-currency volatility (Walther et al., 2019). The recent study uses the predictive power of social signals, specifically user behavior and communication patterns for forecasting prices for cypto-currencies (Glenski et al., 2019). The main shortcoming of all the said studies is that they yield point predictions; given the volatility of the crypto-currency market, the investor needs to know price value and the prediction uncertainty. None of the prior studies can provide that. This is the reason predicting the volatility of cryptocurrencies is a separate branch of research (Ma et al., 2020a; Catania et al., 2018; Kristjanpoller & Minutolo, 2018; Köchling et al., 2020).

Current research in cryptocurrency forecasting offers a piecemeal solution – one approach for predicting price and a separate one for predicting volatility. This is as good as comparing apples to oranges since the fundamental models and assumptions of the two approaches will be different. The paper propose a single model that yields both the point estimate of cryptocurrency prices as well as the uncertainty around the estimate.

The work is based on the classical state-space model (SSM) for time-series analysis. SSM is defined by two functions: the Markovian state model and the observation model. In the traditional approach, the models are assumed to be known, but in realistic financial forecasting applications this is never the case. This is the reason prior studies (Sharma et al., 2021; Sharma & Majumdar, 2021) proposed to learn the models instead; the two aforementioned papers are similar in principle and only vary in the introduction of an exogenous input in the model. However, these works were based on linear state-evolution and observation models and hence could only model piece-wise linear functions. In this work, the above said restriction is removed by proposing to model the underlying functions by deep learning. This results in the so-called deep SSM. The

underlying SSM structure ensures that we can both predict a point estimate as well quantify the uncertainty about it, making it a perfect fit for cryptocurrency forecasting.

The proposed work introduces deep non-negative matrix factorization (deep NMF) models for learning the approximations on operators. Deep NMF (De Handschutter et al., 2021) is equivalent to Deep Rectified Linear Unit (ReLU) networks. There is existing literature (Tariyal et al., 2016; Mahdizadehaghdam et al., 2019) which establishes its connection with deep dictionary learning; the first paper is a more generalised non-linear version of the later. The work have utilised the potential of deep NMF / ReLU in the proposed work by embedding it into a Gaussian SSM. This allows for modelling non-linearity of the underlying dynamical process. The main difference between the prior shallow model and the proposed deep one is that the proposed work is more generalized version of the former. The shallow model can only approximate piece-wise linear functions; the proposed one, being formulated on ReLU network can approximate arbitrary non-linear and non-smooth functions. The price we pay for the generalisation ability of our approach is the difficulty in training. The shallow model allowed for closed form solutions; the proposed deeper extension does not. Hence, the work resort to alternating majorization-minimization (AMM) approach to solve this. The proposed approach is named as Deep state-space model for Predicting Cryptocurrency Price (DeCrypt).

The rest of the paper is organized as follows. Related work in literature is reviewed in Section 2. The proposed model and inference algorithm are explained in Section 3. The application relevance is discussed in Section 4. The experimental results are presented and discussed in Section 5. The acknowledgement is discussed in Section 6. Conclusions and future directions of research are finally given in Section 7.

## 2. Related Work

The objective of this work is to predict cryptocurrency prices. Although the problem is new, it is akin to the problem of stock forecasting in particular and financial forecasting in general. There are two approaches to address such time-series modelling problems. The first approach comprises of state-space-model (SSM) and auto-regressive moving average (ARMA). These methods are usually employed in signal processing and statistical applications (e.g., in statistical ecology Newman et al. (2023)). The second approach is based on machine learning methods, more specifically on recurrent neural network (RNN).

Signal processing techniques are interpretable and yield uncertainty estimates. However, the problem with SSM and ARMA is that their model parameters need to be known. The pros and cons of such assumptions have been studied for the linear (Rankin, 1986) and non-linear (Andersen et al., 2009) cases. Specification of the underlying models lead to simplistic (and restrictive) models that fail to capture the movement of stock prices. ARMA (Rounaghi & Zadeh, 2016) and it variants like Autoregressive Integrated Moving Average (ARIMA) (Jarrett & Kyper, 2011) (Box Jenkins models in general (Dritsaki, 2015)) have been widely used in financial forecasting. These too require specification of underlying parameters defining the price movement; a higher order model fits the training data but fails to generalize and a lower order model yields poor results both on training and testing data.

RNN on the other hand can learn the underlying function from the training data; thanks to their function approximation ability (Hammer, 2000; Garzon & Botelho, 1999). They do not need specifying any function parameter, given enough data they can learn the underlying dynamical model. This is the reason they have been more successful in recent years for financial forecasting (Baek & Kim, 2018; Kim & Kim, 2019). The shortcoming of RNN is that in their vanilla form, they do not yield uncertainty estimates – we have already discussed its importance in the introduction. This is the reason, researchers are concentrating on building RNN models on probabilistic frameworks (Rangapuram et al., 2018;

Ma et al., 2020b).

One must note the fundamental difference between our proposal and previous approaches like (Rangapuram et al., 2018; Ma et al., 2020b). In (Rangapuram et al., 2018) the state-evolution of equation of SSM is modelled as a recurrent neural network, but the observation is assumed to be known with further restrictions of linearity and incoherence. Our model embeds deep neural networks in both the state-evolution and observation models, without any restrictions. One can assume that ours is a more generalized version of (Rangapuram et al., 2018). The work (Ma et al., 2020b) is in some sense complimentary to (Rangapuram et al., 2018); they embed a particle filter in an RNN thereby adding a probabilistic flavour to the otherwise deterministic latent states (see more on particle filters in (Särkkä, 2013; Elvira et al., 2017) for more details).

Several prior studies such as (Digalakis et al., 1993; Sharma et al., 2020) have proposed solutions for the linear SSM when both the state-evolution and observation matrices are unknown. These techniques were able to learn piece-wise linear functions from the data; however they were not able to model arbitrary functions. The work overcomes this limitation by modelling the state-evolution and observation matrices as deep neural networks. Recent papers such as (Liang & Srikant, 2016; Elbrächter et al., 2021) are showing how deep neural networks excel over shallow networks in terms of function approximation; the work is rooted on the same.

## 3. Proposed Method

This section will discuss the proposed approach (Deep State-Space Model for Predicting Cryptocurrency Price (DeCrypt)) in detail. For convenience the proposed method will be referred by name **DeCrypt**.

### *3.1. Model Details*

The proposed work is based on standard state-space model (SSM). It can be expressed as :

For every $k \in \{1, \ldots, K\}$:

$$\begin{cases} \mathbf{z}_k &= f(\mathbf{z}_{k-1}) + g(\mathbf{u}_k) + \mathbf{v}_{1,k}, \\ \mathbf{x}_k &= h(\mathbf{z}_k) + \mathbf{v}_{2,k}. \end{cases} \tag{1}$$

The goal is to infer $(\mathbf{z}_k)_{1 \leq k \leq K}$, a sequence of unknown latent space vector of size $N_z \geq 1$ given the input $(\mathbf{u}_k)_{1 \leq k \leq K}$ vector of size $N_y \geq 1$ and observed sequence $(\mathbf{x}_k)_{1 \leq k \leq K}$ of vector of size $N_x \geq 1$. The work assumes process noises $(\mathbf{v}_{1,k})_{1 \leq k \leq K}$, $(\mathbf{v}_{2,k})_{1 \leq k \leq K}$ to have a Gaussian distribution with zero-mean and covariance matrix $\mathbf{Q}$ and $\mathbf{R}$, respectively. The covariance matrices are symmetric definite positive. Here $K$ is the total number of data to be processed (window size in our case).

Traditional solutions to SSM required the functions $f(.), g(.)$ and $h(.)$ to be known. When the functions are linear, prior studies (Digalakis et al., 1993; Sharma et al., 2020) proposed a solution called blind Kalman filtering; blind since the functions/matrices were assumed to be unknown. Recent extensions introduced a graphical perspective for $f$ (still assumed to be linear), along with suitable sparse priors Elvira & Chouzenoux (2022); Cox & Elvira (2023); Chouzenoux & Elvira (2023).

This work removes the linearity restriction, by embedding ReLU deep neural networks (DNNs) in place of the functions. Our model thus takes the form:
For every $k \in \{1, \ldots, K\}$:

$$\begin{cases} \mathbf{z}_k &= \mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12}\mathbf{z}_{k-1} + \mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22}\mathbf{u}_k + \mathbf{v}_{1,k}, \\ \mathbf{x}_k &= \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{z}_k + \mathbf{v}_{2,k}. \end{cases} \tag{2}$$

This is a multi-linear Gaussian model; everything except the input $(\mathbf{u}_k)_{1 \leq k \leq K}$ and observed sequence $(\mathbf{x}_k)_{1 \leq k \leq K}$ are unknown. The primary objective is to jointly learn the latent factor matrices $\mathbf{T}_{10} \in \mathbb{R}^{N_z \times N_z}$, $\mathbf{T}_{11} \in \mathbb{R}^{N_z \times N_z}$, $\mathbf{T}_{12} \in \mathbb{R}^{N_z \times N_z}$, three positive-valued linear factors leading to a multi-linear state operator $\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12}$, the control input transition matrices $\mathbf{T}_{20} \in \mathbb{R}^{N_z \times N_z}$, $\mathbf{T}_{21} \in \mathbb{R}^{N_z \times N_z}$, $\mathbf{T}_{22} \in \mathbb{R}^{N_z \times N_y}$, three positive-valued linear factors leading to a multi-linear control operator $\mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22}$, and the observation matrices $\mathbf{D}_0 \in \mathbb{R}^{N_x \times N_z}$, $\mathbf{D}_1 \in \mathbb{R}^{N_z \times N_z}$, $\mathbf{D}_2 \in \mathbb{R}^{N_z \times N_z}$, three positive-valued linear

factors yielding the multi-linear observation model $\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2$ and the sequence $(\mathbf{z}_k)_{1\leq k\leq K}$, from observed sequence $(\mathbf{x}_k)_{1\leq k\leq K}$ and $(\mathbf{u}_k)_{1\leq k\leq K}$. [1] The inference problem can be categorised as a blind filtering problem, where the objective is to infer the time series predictions and unknown model parameters from the given input data and observed sequences. As stated earlier, the classical SSM techniques need prior assumptions and information on model parameters. In the proposed model described here, this would imply explicitly setting some prior values to the positive latent factor matrices matrices $\{\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2\}$ involved in both state, control and observation models. In real-world applications, especially in areas as complex as financial modelling this is never known; this is largely owing to the non-stationarity and volatility of the process. The main objective of the proposed work is to provide a point-wise estimate of the positive latent factor matrices $\{\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2\}$ and obtain a probabilistic estimate of sequence $(\mathbf{z}_k)_{1\leq k\leq K}$, given the observed sequence $(\mathbf{x}_k)_{1\leq k\leq K}$ and control input $(\mathbf{u}_k)_{1\leq k\leq K}$. Therefore the work propose to jointly solve for (i) the three deep NMF problems, and (ii) the filtering/smoothing problem.

### *3.2. Model Analysis*

This section will describe the fundamental characteristics of the proposed DeCrypt approach. We have used the discrete time invariant state-space model with exogenous input. The mathematical fundamentals of the model in Eq. (2) and schematic diagram in Fig 1. The former part of the equation works on the evolution of the hidden state parameters, where the model assumes Markovianity between two consecutive hidden states. The later part defines the relationship between the hidden and observed states. We depart from all prior works in SSM based dynamical modelling in the way we define the functions. Usually a

[1] Throughout the paper, three-terms factorizations is considered , for the sake of readability. The 3-layers modeling and inference methodology has the great advantage of being generic enough to be straightforwardly extended to any number, greater or equals to one, of factors.

matrix is used when the functions are assumed to be non-linear and an explicit non-linear function otherwise (Andrieu et al., 2010; Chopin et al., 2013; Crisan & Miguez, 2018). Here we model non-linearity by a deep ReLU network; alternately this can be also seen as a deep NMF (De Handschutter et al., 2021). The reason for using a deep ReLU network is its universal function approximation capbility (Liu & Liang, 2021; Daubechies et al., 2022; Chen et al., 2019). It is essential to note here that classical state-space models uses Monte Carlo simulation or Variable Bayes type techniques, doe non-linear SSM. These techniques are complex and usually do not scale well. In contrast, in the proposed method, each layer is modeled and learnt in the form of matrix that can be estimated using an alternating majorization-minimization (AMM) procedure.

When compared to existing literature in machine learning approaches, DNN mostly utilizes in backpropagation for its training (Chen et al., 2021; Flenner & Hunter, 2017). Consequently while modelling dynamical systems, backpropagagtion through time (BPTT) needs to be employed. We are all aware of the pitfalls of BPTT. This is the reason we resort to AMM instead. Unlike BPTT, AMM (under certain conditions) at least guarantees convergence to a local minimum Chouzenoux et al. (2016); Jacobson & Fessler (2007).

### *3.3. Model Inference Algorithm*

The inference problem can be viewed as smoothing/filtering problem where we aim to infer probabilistic estimate of the hidden state $(\mathbf{z}_k)_{1\leq k\leq K}$. In this work, we have also introduced deep NMF factors (described earlier) which are unknown. We aim to jointly infer both probabilistic distribution on hidden state and deep NMF factors estimation from the data. To estimate the state matrices, control input transition matrices and observation transition matrices, we use Expectation-maximization strategy (for more details (Särkkä, 2013, chap.12) and (Shumway & Stoffer, 1982). The EM strategy operates in two steps namely E-step where we assume the positive latent factor matrices $\{\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2\}$ to be fixed and estimate probabilistic inference for the state representation $(\mathbf{z}_k)_{1\leq k\leq K}$. M-step involves updat-

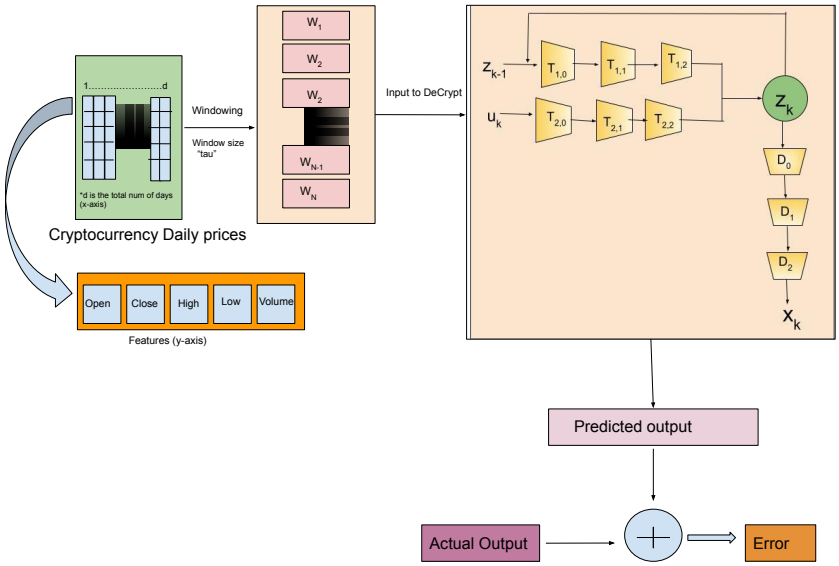



Figure 1: Schematic Diagram for Proposed model DeCrypt.

ing these matrices assuming fixed state (learnt from E-step). The E-step is akin to that of a Kalman filter / smoother. The M-step updates the matrices $\{\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2\}$ by maximizing the upper bound :

$$\varphi_K(\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2) = \log p(\mathbf{x}_{1:K}|\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2). \quad (3)$$

It is important to note that the inference of the $i+1$-th EM update is obtained from the estimates from the previous iteration $i$. We explain the EM algorithm in more detail.

*3.3.1. E-step: Kalman/RTS inference*

We consider the latent factors $\mathbf{T}_{10}^{[i]}$, $\mathbf{T}_{11}^{[i]}$, $\mathbf{T}_{12}^{[i]}$, $\mathbf{T}_{20}^{[i]}$, $\mathbf{T}_{21}^{[i]}$, $\mathbf{T}_{22}^{[i]}$, $\mathbf{D}_0^{[i]}$, $\mathbf{D}_1^{[i]}$, $\mathbf{D}_2^{[i]}$ to be fixed. The objective of this step is to infer the probabilistic estimation of the state. The initial state takes the form $\mathbf{z}_0 \sim \mathcal{N}(\bar{\mathbf{z}}_0, \mathbf{P}_0)$ with $\bar{\mathbf{z}}_0 \in \mathbb{R}$ and $\mathbf{P}_0$ defined as definite symmetric positive matrix $\in \mathbb{R}^{N_z \times N_z}$. The probabilistic estimation is provided by the Kalman filter through predictive distribution :

$$p(\mathbf{z}_k|\mathbf{x}_{1:k}, \mathbf{u}_{1:k}) = \mathcal{N}(\mathbf{z}_k; \bar{\mathbf{z}}_k, \mathbf{P}_k). \quad (4)$$

For every $k$, the mean $\bar{\mathbf{z}}_k$ and the covariance $\mathbf{P}_k$ are given by the Kalman iterations:

For $k = 1, \ldots, K$:

*Predict state:*

$$\begin{cases} \mathbf{z}_k^- &= \mathbf{T}_{10}^{[i]}\mathbf{T}_{11}^{[i]}\mathbf{T}_{12}^{[i]}\bar{\mathbf{z}}_{k-1} + \mathbf{T}_{20}^{[i]}\mathbf{T}_{21}^{[i]}\mathbf{T}_{22}^{[i]}\mathbf{u}_k, \\ \mathbf{P}_k^- &= \mathbf{T}_{10}^{[i]}\mathbf{T}_{11}^{[i]}\mathbf{T}_{12}^{[i]}\mathbf{P}_{k-1}(\mathbf{T}_{10}^{[i]}\mathbf{T}_{11}^{[i]}\mathbf{T}_{12}^{[i]})^\top + \mathbf{Q}. \end{cases} \tag{5}$$

*Update state:*

$$\begin{cases} \mathbf{y}_k &= \mathbf{x}_k - \mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]}\mathbf{z}_k^-, \\ \mathbf{S}_k &= \mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]}\mathbf{P}_k^-(\mathbf{D}_0^{i}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]})^\top + \mathbf{R}, \\ \mathbf{K}_k &= \mathbf{P}_k^-(\mathbf{D}_0^{[i]}\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]})^\top\mathbf{S}_k^{-1}, \\ \mathbf{z}_k &= \mathbf{z}_k^- + \mathbf{K}_k\mathbf{y}_k, \\ \mathbf{P}_k &= \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top. \end{cases} \tag{6}$$

Hereabove, $\mathbf{y}_k$ represents the measurement pre-fit residual, $\mathbf{S}_k$ represents the pre-fit covariance, $\mathbf{K}_k$ represents Kalman gain, $\bar{\mathbf{z}}_k$ represents the updated (a posteriori) state estimate, $\mathbf{P}_k$ represents the updated (a posteriori) covariance estimate. The backward recursion from the RTS smoother allow to build the smoothing distribution $p(\mathbf{z}_k|\mathbf{x}_{1:K}, \mathbf{u}_{1:K})$. For $k = K, \ldots, 1$

*Backward Recursion (Bayesian Smoothing):*

$$\begin{cases} \mathbf{z}_{k+1}^- &= \mathbf{T}_{10}^{[i]}\mathbf{T}_{11}^{[i]}\mathbf{T}_{12}^{[i]}\bar{\mathbf{z}}_k + \mathbf{T}_{20}^{[i]}\mathbf{T}_{21}^{[i]}\mathbf{T}_{22}^{[i]}\mathbf{u}_k, \\ \mathbf{P}_{k+1}^- &= \mathbf{T}_{10}^{[i]}\mathbf{T}_{11}^{[i]}\mathbf{T}_{12}^{[i]}\mathbf{P}_k(\mathbf{T}_{10}^{[i]}\mathbf{T}_{11}^{[i]}\mathbf{T}_{12}^{[i]})^\top + \mathbf{Q}, \\ \mathbf{G}_k &= \mathbf{P}_k(\mathbf{T}_{10}^{[i]}\mathbf{T}_{11}^{[i]}\mathbf{T}_{12}^{[i]})^\top[\mathbf{P}_{k+1}^-]^{-1}, \\ \mathbf{z}_k^s &= \mathbf{z}_k + \mathbf{G}_k[\mathbf{z}_{k+1}^s - \mathbf{z}_{k+1}^-], \\ \mathbf{P}_k^s &= \mathbf{P}_k + \mathbf{G}_k[\mathbf{P}_{k+1}^s - \mathbf{P}_{k+1}^-]\mathbf{G}_k^\top. \end{cases} \tag{7}$$

Consequently, for every time step $k \in \{1, \ldots, K\}$, the RTS smoother provides:

$$p(\mathbf{z}_k|\mathbf{x}_{1:K}, \mathbf{u}_{1:K}) = \mathcal{N}(\mathbf{z}_k; \mathbf{z}_k^s, \mathbf{P}_k^s). \tag{8}$$

*3.3.2. M-step: Operator update*

This step utilizes the estimated state ($\mathbf{z}_k$) following an optimization step to increase the likelihood of the matrix parameters $\mathbf{T}_{10}$, $\mathbf{T}_{11}$, $\mathbf{T}_{12}$, $\mathbf{T}_{20}$, $\mathbf{T}_{21}$, $\mathbf{T}_{22}$, $\mathbf{D}_0$, $\mathbf{D}_1$, $\mathbf{D}_2$, using the smoothed predictive distribution obtained in the E-step.

$$\varphi_K(\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2) \geq \mathcal{Q}(\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2; \boldsymbol{\Theta}^{[i]}). \quad (9)$$

Here above, $\boldsymbol{\Theta}^{[i]} = \{\boldsymbol{\Sigma}^{[i]}, \boldsymbol{\Phi}^{[i]}, \mathbf{B}^{[i]}, \mathbf{C}^{[i]}, \boldsymbol{\Delta}^{[i]}, \mathbf{A}^{[i]}, \mathbf{F}^{[i]}, \mathbf{I}^{[i]}\}$ gathers eight quantities (variables) defined from the outputs of the E-step described in Sec. 3.3.1):

$$\begin{aligned}
&\mathbf{Q}(\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2; \boldsymbol{\Theta}^{[i]}) = \\
&+\frac{K}{2}\mathrm{tr}\Big(\mathbf{Q}^{-1}(\boldsymbol{\Sigma}^{[i]} - (\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12})^\top \mathbf{C}^{[i]} - \mathbf{A}^{[i]}(\mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22})^\top - (\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12})(\mathbf{C}^{[i]})^\top \\
&+(\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12})\boldsymbol{\Phi}^{[i]}(\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12})^\top + (\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12})\mathbf{F}^{[i]}(\mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22})^\top - (\mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22})(\mathbf{A}^{[i]})^\top \\
&\quad +(\mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22})(\mathbf{F}^{[i]})^\top(\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12})^\top + (\mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22})\mathbf{I}^{[i]}(\mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22})^\top)\Big) \\
&+\frac{K}{2}\mathrm{tr}\Big(\mathbf{R}^{-1}\boldsymbol{\Delta}^{[i]} - \mathbf{B}^{[i]}(\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2)^\top - \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2(\mathbf{B}^{[i]})^\top + \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\boldsymbol{\Sigma}^{[i]}(\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2)^\top\Big),
\end{aligned} \quad (10)$$

with:

$$\begin{cases}
\boldsymbol{\Sigma}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_k^s + \mathbf{z}_k^s(\mathbf{z}_k^s)^\top, \\
\boldsymbol{\Phi}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{P}_{k-1}^s + \mathbf{z}_{k-1}^s(\mathbf{z}_{k-1}^s)^\top, \\
\mathbf{B}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k(\mathbf{z}_k^s)^\top, \\
\mathbf{C}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} (\mathbf{P}_k^s\mathbf{G}_{k-1}^\top + \mathbf{z}_k^s(\mathbf{z}_{k-1}^s)^\top). \\
\mathbf{A}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{z}_k^s\mathbf{u}_k^\top \\
\mathbf{F}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{z}_{k-1}^s\mathbf{u}_k^\top \\
\mathbf{I}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{u}_k\mathbf{u}_k^\top \\
\boldsymbol{\Delta}^{[i]} &= \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_k\mathbf{x}_k^\top
\end{cases} \quad (11)$$

In this step lies the main bottleneck of the deeper extension (compared to the shallow state-space model of (Sharma & Majumdar, 2021)). For the shallow model each of the operators was a single matrix; therefore there update step resulted in a linear inverse problem. Such is not the current case; here the variables are multi-linear in nature. Therefore we do not have the simple (analytic) updates - as was the case for the shallow model. We have to resort to the paradigm of alternating direction method of multipliers (ADMM)(Wang et al., 2019; Nishihara et al., 2015; Lin et al., 2015) for solving updating the variables

from the multi-linear form. In ADMM, the idea is that, one can update one variable assuming the others to be constant and as long as each of the variables have a closed form update, the overall optimization will reach a local minimum. Based on the ADMM approach the computations each variable under the positivity constraints on the factors $\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}^{[i+1]}, \mathbf{T}_{20}^{[i+1]}, \mathbf{T}_{21}^{[i+1]}, \mathbf{T}_{22}^{[i+1]}$ and $\mathbf{D}_0^{[i+1]}, \mathbf{D}_1^{[i+1]}, \mathbf{D}_2^{[i+1]}$ leads to:

$$
\begin{aligned}
(\mathbf{T}_{10})^{[i+1]} &= \underset{\mathbf{T}_{10}\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}, \mathbf{T}_{11}^{[i]}, \mathbf{T}_{12}^{[i]}, \mathbf{T}_{20}^{[i]}, \mathbf{T}_{21}^{[i]}, \mathbf{T}_{22}^{[i]}\mathbf{D}_0^{[i]}, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\
(\mathbf{T}_{11})^{[i+1]} &= \underset{\mathbf{T}_{11}\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}, \mathbf{T}_{12}^{[i]}, \mathbf{T}_{20}^{[i]}, \mathbf{T}_{21}^{[i]}, \mathbf{T}_{22}^{[i]}D_0^{[i]}, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\
(\mathbf{T}_{12})^{[i+1]} &= \underset{\mathbf{T}_{12}\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}, \mathbf{T}_{20}^{[i]}, \mathbf{T}_{21}^{[i]}, \mathbf{T}_{22}^{[i]}\mathbf{D}_0^{[i]}, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\
(\mathbf{T}_{20})^{[i+1]} &= \underset{\mathbf{T}_{20}\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}^{[i+1]}, \mathbf{T}_{20}, \mathbf{T}_{21}^{[i]}, \mathbf{T}_{22}^{[i]}\mathbf{D}_0^{[i]}, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\
(\mathbf{T}_{21})^{[i+1]} &= \underset{\mathbf{T}_{21}\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}^{[i+1]}, \mathbf{T}_{20}^{[i+1]}, \mathbf{T}_{21}, \mathbf{T}_{22}^{[i]}\mathbf{D}_0^{[i]}, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\
(\mathbf{T}_{22})^{[i+1]} &= \underset{\mathbf{T}_{22}\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}^{[i+1]}, \mathbf{T}_{20}^{[i+1]}, \mathbf{T}_{21}^{[i+1]}, \mathbf{T}_{22}\mathbf{D}_0^{[i]}, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\
(\mathbf{D}_0)^{[i+1]} &= \underset{\mathbf{D}_0\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}^{[i+1]}, \mathbf{T}_{20}, \mathbf{T}_{21}^{[i+1]}, \mathbf{T}_{22}^{[i+1]}\mathbf{D}_0, \mathbf{D}_1^{[i]}, \mathbf{D}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\
(\mathbf{D}_1)^{[i+1]} &= \underset{\mathbf{D}_1\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}^{[i+1]}, \mathbf{T}_{20}^{[i+1]}, \mathbf{T}_{21}^{[i+1]}, \mathbf{T}_{22}^{[i]}, \mathbf{D}_0^{[i]}, \mathbf{D}_1, \mathbf{D}_2^{[i]}; \mathbf{\Theta}^{[i]}) \\
(\mathbf{D}_2)^{[i+1]} &= \underset{\mathbf{D}_2\geq 0}{\operatorname{argmax}} \mathbf{Q}(\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}^{[i+1]}, \mathbf{T}_{20}^{[i+1]}, \mathbf{T}_{21}^{[i+1]}, \mathbf{T}_{22}^{[i+1]}\mathbf{D}_0^{[i+1]}, \mathbf{D}_1^{[i+1]}, \mathbf{D}_2; \mathbf{\Theta}^{[i]})
\end{aligned}
$$

The above sub-problems can easily be rewritten as the minimization of convex quadratic functions which can be solved through several solvers. We stick to use simple projected least-squares updates, which is also reminiscent from the literature of deep nonnegative matrix factorization (Chen et al., 2021), and the deep ReLu neural networks models (Daubechies et al., 2022). The deep neural network (DNN) based operators are regularized by imposing a positivity constraint on the entries of the estimated matrices, by simply projecting them onto the positive orthant after each update of the M-step (similar to ReLU activation function). This is akin to deep non-negative matrix factorization (Trigeorgis et al., 2016; Mei et al., 2019), while keeping the convergence behaviour of the EM algorithm. DNN with ReLU activation is known for its function approximation ability (Chen et al., 2019; Yarotsky, 2018). This yields the following

analytic updates:

$$
\begin{aligned}
\mathbf{T}_{10}^{[i+1]} &= \mathrm{ReLu}\Big(\big(\mathbf{C}^{[i]}(\mathbf{T}_{12}^{[i]})^\top(\mathbf{T}_{11}^{[i]})^\top\big) - \big(\mathbf{T}_{20}^{[i]}\mathbf{T}_{21}^{[i]}\mathbf{T}_{22}^{[i]}(\mathbf{F}^{[i]})^\top(\mathbf{T}_{12}^{[i]})^\top(\mathbf{T}_{11}^{[i]})^\top\big) \\
&\quad \times(\mathbf{T}_{11}^{[i]}\mathbf{T}_{12}^{[i]}\mathbf{\Phi}^{[i]}(\mathbf{T}_{12}^{[i]})^\top(\mathbf{T}_{11}^{[i]})^\top))^\dagger\Big) \\
\mathbf{T}_{11}^{[i+1]} &= \mathrm{ReLu}\Big(((\mathbf{T}_{10}^{[i+1]})^\top\mathbf{Q}^{-1}(\mathbf{T}_{10}^{[i+1]})^{-1})\big(\big((\mathbf{T}_{10}^{i+1})^\top\mathbf{Q}^{-1}\mathbf{C}^{[i]}\mathbf{T}_{12}^{[i]}\big) \\
&\quad -\big(\mathbf{T}_{10}^{[i+1]}\mathbf{Q}^{-1}\mathbf{T}_{20}^{[i]}\mathbf{T}_{21}^{[i]}\mathbf{T}_{22}^{[i]}(\mathbf{F}^{[i]})^\top\mathbf{T}_{12}^{[i]}\big)\big) \times (\mathbf{T}_{12}^{[i]}\mathbf{\Phi}^{[i]}(\mathbf{T}_{12}^{[i]})^\top)^\dagger\Big) \\
\mathbf{T}_{12}^{[i+1]} &= \mathrm{ReLu}\Big(((\mathbf{T}_{11}^{[i+1]})^\top(\mathbf{T}_{10}^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{T}_{10}^{[i+1]}\mathbf{T}_{11}^{[i+1]})^\dagger \times \big((\mathbf{T}_{11}^{[i+1]})^\top(\mathbf{T}_{10}^{[i+1]})^\top\mathbf{C}^{[i]}\mathbf{Q}^{-1}\big) \\
&\quad -\big((\mathbf{T}_{11}^{[i+1]})^\top(\mathbf{T}_{10}^{[i+1]})^\top(\mathbf{Q}^{-1}\mathbf{T}_{20}^{[i]}\mathbf{T}_{21}^{[i]}\mathbf{T}_{22}^{[i]}(\mathbf{F}^{[i]})^\top)\mathbf{\Phi}^{-1}\big)\Big) \\
\mathbf{T}_{20}^{[i+1]} &= \mathrm{ReLu}\Big(\big((\mathbf{T}_{22}^{[i]})^\top(\mathbf{T}_{21}^{[i]})^\top - \mathbf{T}_{10}^{[i+1]}\mathbf{T}_{11}^{[i+1]}\mathbf{T}_{12}^{[i+1]}\mathbf{F}^{[i]}(\mathbf{T}_{22}^{[i]})^\top(\mathbf{T}_{21}^{[i]})^\top\big) \\
&\quad \times(\mathbf{T}_{21}^{[i]}\mathbf{T}_{22}^{[i]}\mathbf{I}^{[i]}(\mathbf{T}_{21}^{[i]})^\top(\mathbf{T}_{22}^{[i]})^\top))^\dagger\Big) \\
\mathbf{T}_{21}^{[i+1]} &= \mathrm{ReLu}\Big(\big(\mathbf{T}_{20}^{[i+1]}\mathbf{Q}^{-1}(\mathbf{T}_{20}^{[i+1]})^\top\big)^\dagger\big(\mathbf{T}_{20}^{[i+1]}\mathbf{A}^{[i+1]}\mathbf{Q}^{-1}(\mathbf{T}_{22}^{[i]})^\top \\
&\quad -\big((\mathbf{T}_{20}^{[i+1]})^\top\mathbf{Q}^{-1}(\mathbf{T}_{10}^{[i+1]})(\mathbf{T}_{11}^{[i+1]})(\mathbf{T}_{12}^{[i+1]})\mathbf{F}^{[i]}(\mathbf{T}_{22}^{[i]})^\top)\big) \\
&\quad \times(\mathbf{T}_{22}^{[i]}\mathbf{I}^{[i]}(\mathbf{T}_{22}^{[i]})^\top)^\dagger\Big) \\
\mathbf{T}_{22}^{[i+1]} &= \mathrm{ReLu}\Big(\big((\mathbf{T}_{21}^{[i+1]})^\top(\mathbf{T}_{20}^{[i+1]})^\top\mathbf{Q}^{-1}(\mathbf{T}_{20}^{[i+1]})(\mathbf{T}_{21}^{[i+1]})\big) \times \big(\big((\mathbf{T}_{21}^{[i+1]})^\top(\mathbf{T}_{20}^{[i+1]})^\top\mathbf{Q}^{-1}\mathbf{A}^{[i]}\big) \\
&\quad -\big((\mathbf{T}_{21}^{[i+1]})^\top(\mathbf{T}_{20}^{[i+1]})^\top\mathbf{Q}^{-1} \times (\mathbf{T}_{10}^{[i+1]})^\top(\mathbf{T}_{11}^{[i+1]})^\top(\mathbf{T}_{12}^{[i+1]})^\top\mathbf{F}^{[i]}\big)\big)(\mathbf{I}^{[i]})^{-1}\Big) \\
\mathbf{D}_0^{[i+1]} &= \mathrm{ReLu}\Big(\mathbf{B}^{[i]}(\mathbf{D}_2^{[i]})^\top(\mathbf{D}_1^{[i]})^\top(\mathbf{D}_1^{[i]}\mathbf{D}_2^{[i]}\mathbf{\Sigma}^{[i]}(\mathbf{D}_2^{[i]})^\top(\mathbf{D}_1^{[i]})^\top)^\dagger\Big) \\
\mathbf{D}_1^{[i+1]} &= \mathrm{ReLu}\Big(((\mathbf{D}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{D}_0^{[i+1]})^\dagger((\mathbf{D}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{B}^{[i]}(\mathbf{D}_2^{[i]})^\top \\
&\quad \times(\mathbf{D}_2^{[i]}\mathbf{\Sigma}^{[i]}(\mathbf{D}_2^{[i]})^\top)^\dagger\Big) \\
\mathbf{D}_2^{[i+1]} &= \mathrm{ReLu}\Big(((\mathbf{D}_1^{[i+1]})^\top(\mathbf{D}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{D}_0^{[i+1]}\mathbf{D}_1^{[i+1]})^\dagger(\mathbf{D}_1^{[i+1]})^\top \\
&\quad \times(\mathbf{D}_0^{[i+1]})^\top\mathbf{R}^{-1}\mathbf{B}^{[i]}(\mathbf{\Sigma}^{[i]]})^{-1}\Big)\,. \qquad (12)
\end{aligned}
$$

Hereabove, we use pseudo-inverse operator denoted by $(\cdot)^\dagger$. Each operator is passed over activation function ReLu which stands for, $\mathrm{ReLu}\,(\cdot)$ the rectified linear unit function, that projects each entry of its input to the positive orthant.

### *3.4. Model Summary*

The Proposed algorithm is summarized in Alg.1. The algorithm infers the probabilistic estimation of the hidden state $(\mathbf{z}_k)_{1\leq k\leq K}$ jointly with the estima-

tion of latent spaces $\{\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2\}$ by following the eq. (2). The DeCrypt model is ran for $i_{\max}$ number of iteration to achieve the stabilisation of the latent spaces.

***Algorithm 1. Proposed model inference algorithm.***

***Inputs.*** *Prior parameters* $(\overline{\mathbf{z}}_0, \mathbf{P}_0)$ *; model noise covariance matrices* $\mathbf{Q}$, $\mathbf{R}$ *; set of observations* $\{\mathbf{x}_k\}_{1\leq k\leq K}$ *and control input* $(\mathbf{u}_k)_{1\leq k\leq K}$ *.*

***Initialization.*** *Set positive latent factors*

$\{\mathbf{T}_{10}, \mathbf{T}_{11}, \mathbf{T}_{12}, \mathbf{T}_{20}, \mathbf{T}_{21}, \mathbf{T}_{22}, \mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2\}$.

***Recursive step.*** *For* $i = 0, 1, \ldots, i_{\max}$*:*

*(E step) Run the Kalman filter* (5)-(6) *and RTS smoother* (7) *using latent factors* $\mathbf{T}_{10}^{[i]}$, $\mathbf{T}_{11}^{[i]}$, $\mathbf{T}_{12}^{[i]}$, $\mathbf{T}_{20}^{[i]}$, $\mathbf{T}_{21}^{[i]}$, $\mathbf{T}_{22}^{[i]}$, $\mathbf{D}_0^{[i]}$, $\mathbf{D}_1^{[i]}$, $\mathbf{D}_2^{[i]}$.

*Calculate* $\mathbf{\Sigma}^{[i]}, \mathbf{\Phi}^{[i]}, \mathbf{B}^{[i]}, \mathbf{C}^{[i]}, \mathbf{\Delta}^{[i]}, \mathbf{A}^{[i]}, \mathbf{F}^{[i]}, \mathbf{I}^{[i]}$ *using* (11).

*(M step) Compute* $\{\mathbf{T}_{10}^{[i+1]}, \mathbf{T}_{11}^{[i+1]}, \mathbf{T}_{12}^{[i+1]}, \mathbf{T}_{20}^{[i+1]}, \mathbf{T}_{21}^{[i+1]}, \mathbf{T}_{22}^{[i+1]}, \mathbf{D}_0^{[i+1]}, \mathbf{D}_1^{[i+1]}, \mathbf{D}_2^{[i+1]}\}$ *using* (12).

***Output.*** *State filtering/smoothing pdfs* (4) *and* (8) *along with pointwise estimates of the latent factor from* (12).

*3.4.1. Time complexity*

▷ **Training.** We describe the time complexity for training DeCrypt in a given window of length $\tau$. The complexity can be understood by delving into Kalman-based approaches (Montella, 2011), which imply complexity analysis to $\mathcal{O}(\tau N_z^{2.376})$.

▷ **Testing.** In testing phase we just perform evaluation of multi-linear equation

(Equation 14). This concludes the complexity of $\mathcal{O}(N_x N_z^2)$ for each window. This can be further optimized to $\mathcal{O}(N_z^2)$ if performing forecast for just one feature (which looks very similar to the proposed case, ie. forecasting close price.)

## 4. Application to cryptocurrency price forecasting

This section discuss in detail how the proposed approach is applied on the very challenging application of predicting next day prices for crypto-currency.

### *4.1. Training*

The major drawback of the Expected-Maximization (EM) strategy used in the proposed approach is that it requires reprocessing on *the entire sequence* to estimate the state, control input, and observation transition operators / matrices. The approach requires imposing explicit prior static assumptions on these parameters for the entire duration of the sequence. Such an assumption may not be an appropriate in practice owing to the volatility of the data; furthermore, processing the entire sequence will be computationally expensive. Owing to the volatility, the parameters should be given the freedom to learn and evolve with time. We thus propose an online implementation based on a simple windowing strategy. Thus we are able to relax the non-volatility assumption on the entire sequence and reduce processing times.

In the said strategy, a window of size $\tau$ is slid on the entire dataset. For every time stamp $k$, the matrices in the multi-linear operators are estimated using the last $\tau$ observations contained in the set $\mathcal{X}_k = \{\mathbf{x}_j\}_{j=k-\tau+1}^{k}$ and $\mathcal{U}_k = \{\mathbf{u}_j\}_{j=k-\tau+1}^{k}$. The proposed EM algorithm is iteratively applied on the window to update the state and operators till convergence. Such a strategy reduces the operational cost significantly. Note that the non-volatility assumption is still there, but only on a small window - this is a reasonable assumption. The major challenge is to estimate a reasonable size of $\tau$. A smaller size would be a better approximation for the non-volatility assumption but would lead to overfitting on the multi-linear model. On the other hand a larger size would be

less prone to over-fitting but would be computationally costly. We initialize the matrices of the multi-linear model using the warm start strategy. The matrices for the current window are initialized with the final values of the prior window. Similarly, the mean and covariance are initialized for $k - \tau + 1$ with the past information on the operators from the smoothing process.

*4.2. Forecasting*

As described in detail in Section *Model Details*, we follow the sliding window strategy. Training each observed window $\mathbf{x}_k$ along with control-input $\mathbf{u}_k$, extracts latent space features and helps in updating the parameters by following EM alternately. Once EM iteration stabilizes ( we have used 50 iterations for EM to converge used in Alg.1), we use the latent space features and learned matrices to estimate the close price for the next timestamp (i.e., the day indexed as $k + \tau + 1$).

$$\begin{cases} \mathbf{z}_k &= \mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12}\mathbf{z}_{k-1} + \mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22}\mathbf{u}_k + \mathbf{v}_{1,k}, \\ \mathbf{x}_k &= \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{z}_k + \mathbf{v}_{2,k}, \end{cases}$$

where input $\mathbf{u}_k$ is $(\mathbf{u}_j)_{k \le j \le k+\tau} \in \mathbb{R}^1$ which is computed using the technical indicator SMA(simple moving average)[2].Simple moving average (SMA) calculates the average of the a fixed range of prices, usually closing price by the number of period in that range.

$$SMA = \frac{c_1 + c_2 + ... + c_n}{n} \tag{13}$$

where $c_n$= closing price of an asset for period n and n= number of total periods The processing sequence(observed) $\mathbf{x}_k$ is $(\mathbf{x}_j)_{k \le j \le k+\tau} \in \mathbb{R}^5$ for every $k \in \{0, \ldots, K - \tau\}$, where $x_j[1]$ is the daily opening price, $x_j[2]$ is the daily adjusted close price, $x_j[3]$ is the daily high value, $x_j[4]$ is the daily low value, and $x_j[5]$ is the daily net asset volume. Running the DeCrypt model on the considered window yields the mean estimate of the five features for the immediate

[2] `https://www.investopedia.com/terms/s/sma.asp`

time stamp which can be indexed as $k + \tau + 1$:

$$\widehat{\mathbf{x}}_{k+\alpha+1} = \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{z}_{k+\alpha}^-, \tag{14}$$

The associated covariance matrix is defined as $\mathbf{S}_{k+\tau}$ for immediate next time stamp indexed as $k+\tau+1$. /cblueIn particular, *DeCrypt* is designed to forecast the whole five dimensional vector, but we focus primarily on prediction of single entry of the vector i.e., adjusted closing price of the sequence.

### *4.3. Uncertainty Quantification*

The proposed approach is based on a probabilistic framework and hence can provide confidence score/uncertainty quantification associated with each point-wise estimation. The quantification provides predictive distribution of the future observation which is conditioned on previously seen control-input ($\mathbf{u}_k$) and observed sequence ($\mathbf{x}_k$). The probabilistic validation provides informed decision about the (un)certainty associated with model estimation while predicting the future prices of the cryptocurrencies. For each index $k$, the distribution of the prediction conditioned on $\widehat{\mathbf{x}}_k$ past observations and control-input $\widehat{\mathbf{u}}_k$ :

$$p(\widehat{\mathbf{x}}_k|\mathbf{x}_{1:k-1}, \mathbf{u}_{1:k-1}) = \mathcal{N}\left(\widehat{\mathbf{x}}_k; \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{z}_k^-, \mathbf{S}_k\right), \tag{15}$$

where the covariance is defined as $\mathbf{S}_k = \mathbf{D}_0\mathbf{D}_1\mathbf{D}_2((\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12})\mathbf{P}_{k-1}(\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12})^\top + \mathbf{Q}) + \mathbf{R}$. It is important to note that $\mathbf{z}_k^-$ and $\mathbf{P}_k$ are achieved from Kalman filter, defined in Section E-step in DeCrypt model. The main objective of the proposed approach is to estimate the uncertainty score associated with the prediction given by model for price forecasting. In particular, the main aim is to focus on forecasting the sum of prediction i.e., estimating the first entry $\widehat{\mathbf{x}}_k$ denoted by $\widehat{\mathbf{x}}_k[0]$. The quantification about the prediction can be obtained from first row and column of $\mathbf{S}_k$, depicted by $\mathbf{S}_k[0,0]$. We define the (un)certainty score about an increase of the price forecasting value as :

$$\widehat{p}_k = \int_{\widehat{\mathbf{x}}_k[0]}^{+\infty} \mathcal{N}\left(y; \left[\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{z}_k^-\right][0], \mathbf{S}_k[0,0]\right) dy \tag{16}$$

$$= 1 - \text{CDF}(\widehat{\mathbf{x}}_k[0]|\left[\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2\mathbf{z}_k^-\right][0], \mathbf{S}_k[0,0]), \tag{17}$$

where CDF depicts the cumulative distribution function for the multivariate predictive distribution. The equation above quantifies and estimates the probability that forecasting price value will grow in the future time stamp. After estimating $\widehat{p}_k$ for every index $k$, we evaluate cross entropy loss defined as :

$$\text{log-loss} = \frac{1}{K}\sum_{k=1}^{K} -\left(L_k[i]\log(\widehat{p}_k)\right), \tag{18}$$

where ground-truth is depicted as $L_k \in \{0, 1\}$ at time $k$ to increase/decrease.

## 5. Experiments and Results

This section discuss the experimental results with proposed approach and other state-of-the art methods. The section presents qualitative and quantitative analysis, hence presenting comprehensive study for modeling time series signals.

### *5.1. Dataset description*

In this study, we consider a cryptocurrency dataset comprising of ten cryptocurrencies. The dataset is extracted from the Yahoo finance cryptocurrency repository using Yahoo finance API [3]. It consists of active cryptocurrencies ranging from some old and new cryptocurrencies. The data extracted is about eight years (between 01/01/2014 to 01/06/2021) for Litecoin, Namecoin, Dogecoin, Peercoin, Bitcoin, Ripple, NXT. For Gridcoin and Ethereum, we extracted seven years of data (between 01/01/2015 to 01/06/2021), the time of its first release year. The dataset created is divided into train and test datasets. Training data is from the year 2014 to 2018(2017 December). In contrast, testing data is from 2018 to 2021 is used.

### *5.2. Baseline methods*

- N-Beats: Nbeats is a deep neural network structure with forward and backward residual links. It consists of a deep stack of fully connected

[3] `https://finance.yahoo.com/cryptocurrencies`

layers. It does not perform any specific feature engineering or scaling (Oreshkin et al., 2019).

- Deep Auto regressive (DeepAR) : DeepAR functions by producing probabilistic forecast on long time series sequences (Salinas et al., 2020).

- Temporal Fusion Transformers (TFT): An attention mechanism based recurrent architecture which brings together multi-horizon forecasting without having prior information on how they interact with the target (Lim et al., 2019).

- Long short term memory(LSTM) : Stacked LSTM with 2-layer architecture is used to forecast the time series sequences. The LSTM used comprises 50 cells and ReLU activation to estimate the predictions.(Elsworth & Güttel, 2020).

- Convolutional neural network- Technical analysis(CNN-TA) :1-D Time series is converted to 2-D matrix with technical indicators as rows and time-units as columns. 2D CNN is used for classification and regression.(Sezer & Ozbayoglu, 2018)

- Recurrent dictionary learning (RDL): The RDL approach can be assumed to be a shallow (single layer) version of the proposed work. It can only model piece-wise linear functions.(Sharma et al., 2021)

- Multi filter neural network (MFNN) : An end-to-end deep neural network comprising of recurrent neural network and convolutional neural network.(Long et al., 2019)

- RAO-ANN: The Rao algorithms can be categorised as the metaphor-less optimization techniques which is utilized in optimizing the parameters of ANN in forecasting crypto-currency prices (Nayak et al., 2021).

- ARIMA: Autoregressive integrated moving average (ARIMA) methodology is used to forecast cryptocurrency prices. The paper identifies the pa-

rameter of ARIMA model using partial auto-correlation functions (PACF) and auto correlation function (ACF).(Abu Bakar & Rosbi, 2017)

We have compared our proposed approach with above mentioned models. The ARIMA parameters are set to $(p, d, q) = (2, 1, 2)$ as it was observed to lead to the best practical performance. We modified LSTM from its original version, by removing the softmax layer and instead included a fully-connected layer to obtain a one node output. The Adam optimizer has been used with learning rate of $10^{-4}$, 200 epochs and batch-size 16 is maintained to minimize the root mean square error. For methods like N-Beats, DeepAR, TFT, CNN-TA, MFNN, RDL, RAO-ANN we stick to their original implementation as described in respective papers mentioned above.

*5.3. Proposed model Parameter analysis*

The proposed approach is non-parametric. It only requires the specification of the window size; we found that $\tau = 50$ yields good results for all cryptocurrencies. The rest of the variables require initialization. These are given below:

$\mathbf{P}_0 = \sigma_P^2 \mathbf{I}$

$\mathbf{Q} = \sigma_Q^2 \mathbf{I}$

$\mathbf{R} = \sigma_R^2 \mathbf{I}$

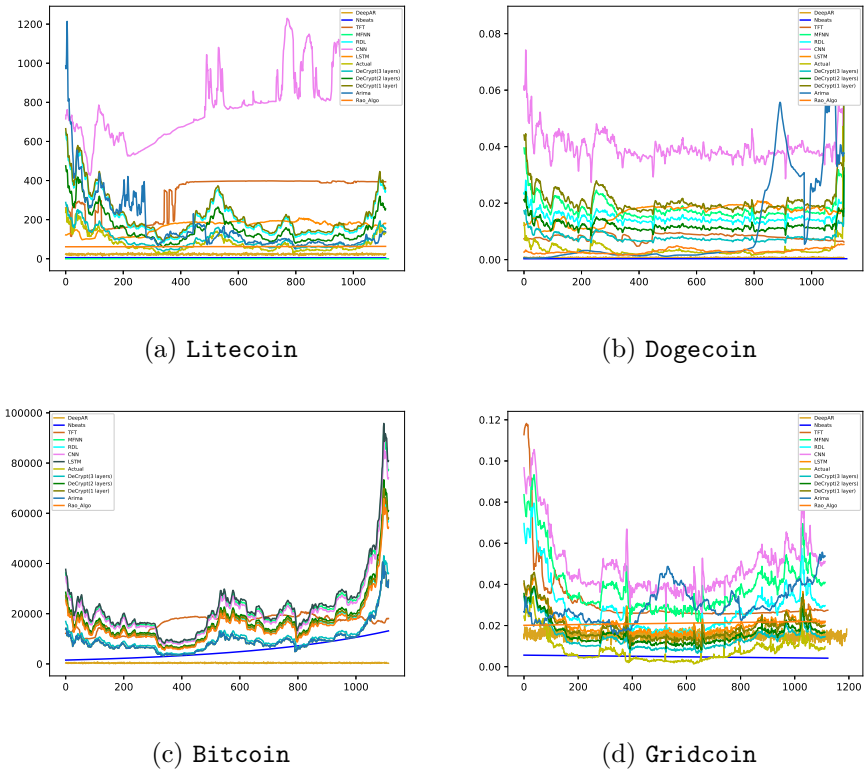
$(\sigma_Q, \sigma_R, \sigma_P) = (10^{-5}, 10^{-1}, 10^{-1})$, $\mathbf{I}$ : identity matrix

$\overline{z}_0 = \mathbf{0}$

$N_z$= 5,$N_y$= 1, $N_x$=5,

During training, the entries of the Deep neural network (DNN) matrices / operators are initialized at time 0 using a uniform distribution on $[0, 10^{-1}]$. During the test phase, these matrices are fixed, and only the Kalman/RTS inference is run. All presented scores are averaged over 10 trials, and computed only during the test phase. More specifically, we will distinguish in the experiments:

**DeCrypt (1 layer):** $\mathbf{T}_{11} = \mathbf{T}_{12} = \mathbf{I}$ is fixed and $\{\mathbf{T}_{10}\}$ is estimated; $\mathbf{T}_{21} = \mathbf{T}_{22} = \mathbf{I}$ is fixed and $\{\mathbf{T}_{20}\}$ is estimated; and $\mathbf{D}_1 = \mathbf{D}_2 = \mathbf{I}$ fixed and $\{\mathbf{D}_0\}$ is estimated;



(a) `Litecoin` (b) `Dogecoin`

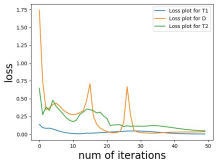
(c) `Bitcoin` (d) `Gridcoin`

Figure 2: Cryptocurrency price forecasting via different algorithms evaluating test data for (a) Litecoin, (b) Dogecoin, (c) Bitcoin, (d) Gridcoin

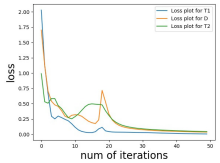
**DeCrypt (2 layers):** $\mathbf{T}_{12} = \mathbf{I}$ is fixed and $\{\mathbf{T}_{10}\,\mathbf{T}_{11}\}$ is estimated; $\mathbf{T}_{22} = \mathbf{I}$ is fixed and $\{\mathbf{T}_{20}, \mathbf{T}_{21}\}$ is estimated; and $\mathbf{D}_2 = \mathbf{I}$ fixed and $\{\mathbf{D}_0\,\mathbf{D}_1\}$ is estimated;

**DeCrypt (3 layers):** $\{\mathbf{T}_{10}\,\mathbf{T}_{11}\,\mathbf{T}_{12}\}$, $\{\mathbf{T}_{20}\,\mathbf{T}_{21}\,\mathbf{T}_{22}\}$ is estimated $\{\mathbf{D}_0, \mathbf{D}_1, \mathbf{D}_2\}$ are estimated.

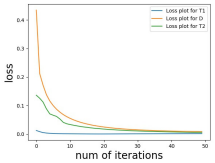
It is important to note that ignoring the positivity constraints on DeCrypt (1 layer) would identify with the previous work (Sharma & Majumdar, 2021). For benchmarking we have used the commons metrics for regression; they are **Root mean square error** (RMSE), **Mean Absolute Percentage Error** (MAPE) and **Symmetric Mean Absolute Percentage Error** (SMAPE).




(a) `1-Layer`

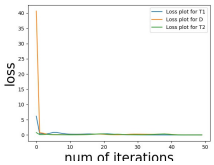



(b) `2 Layers`

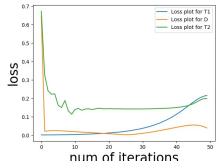



(c) `3 Layers`

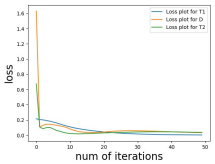
(A.) Convergence plot for Bitcoin




(a) `1 Layer`

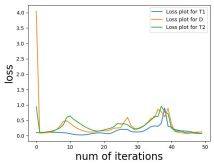



(b) `2 Layers`

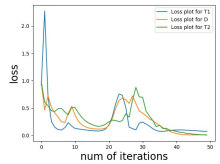



(c) `3 Layers`

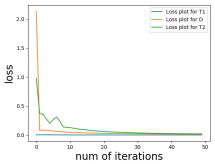
(B.) Convergence plot for Gridcoin




(a) `1 Layer`




(b) `2 Layers`




(c) `3 Layers`

(C.) Convergence plot for Litecoin

Figure 3: Convergence plots for different layer architecture((a.) 1 Layer, (b.) 2 Layers, (c.) 3 Layer) for (A.) Bitcoin, (B.) Gridcoin, (C.) Litecoin

All the said metrics are based on the error between the actual and predicted prices and a lower value implies better result. We have also computed the **Pearson correlation coefficient** (r) between the predicted and actual prices; for this metric a higher value implies better result.

We are showing these results for a given window size $\tau = 50$ and number of layers. If we continue to increase the window size, the results start to deteriorate, this is likely due to over-fitting. Increasing the window size does not help either;

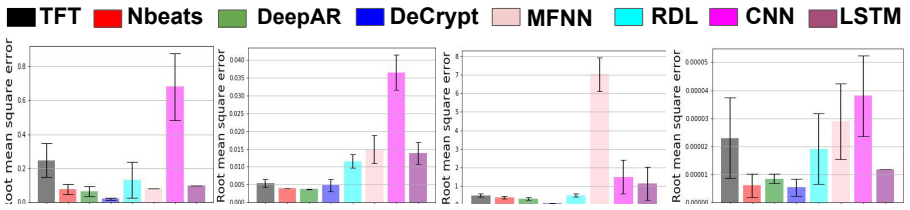



Figure 4: Error bar plot comparing RMSE for DeCrypt against other baseline methods for (a) Litecoin, (b) Dogecoin, (c) Ethereum, (d) Gridcoin.

larger window size fails to capture the volatility of the data and hence the performance falls.

### *5.4. Result Analysis Discussions*

In this section we focus on the performance analysis of the proposed approach. The proposed approach (DeCrypt) has been compared with various numerical methods like N-Beats, DeepAR, TFT, CNN-TA, MFNN, RDL, RAO-ANN, ARIMA, RAO-ANN.

#### *5.4.1. Influence of window size and depth*

The proposed solution is non-parametric. The only design parameters that need to be fixed are the window size and depth. Therefore, it is very important to choose the optimal window size which finds a good balance between the model complexity (depth) and accuracy. To better understand, we present Table 1, Table 2, Table 3 which represent the window size analysis in DeCrypt (1 layer), DeCrypt (2 layers), DeCrypt (3 layers) architecture respectively. The above mentioned tables provides comprehensive analysis on the performance of the model on varying window sizes. A comprehensive study has been compiled which offers analysis of various metrics like Pearson correlation (r), RMSE (Root Mean Square Error), MAE (Mean Absolute Error), and SMAPE (Symmetric mean absolute percentage error) for different window sizes $\tau$. After analysing the results we conclude that model performance keeps on improving as window

| Window size($\tau$) | $r$ | RMSE↓ | MAPE(%)↓ | SMAPE(%)↓ |
|---|---|---|---|---|
| 10 | 0.38 | 0.51 | 72.1 | 69.3 |
| 15 | 0.37 | 0.50 | 73.2 | 70.4 |
| 20 | 0.42 | 0.47 | 69.6 | 65.2 |
| 25 | 0.47 | 0.45 | 65.4 | 61.2 |
| 30 | 0.53 | 0.43 | 61.3 | 57.3 |
| 35 | 0.57 | 0.46 | 58.5 | 53.4 |
| 40 | 0.63 | 0.39 | 51.8 | 52.3 |
| 45 | 0.67 | 0.32 | 43.6 | 41.5 |
| 50 | 0.73 | 0.21 | 35.7 | 32.3 |
| 55 | 0.73 | 0.22 | 36.2 | 33.2 |
| 60 | 0.71 | 0.20 | 33.2 | 31.7 |

Table 1: Comparative performance of the proposed approach against different window size for 1-layer architecture. Lower value(↓) is considered the better.

| Window size($\tau$) | $r$ | RMSE↓ | MAPE(%)↓ | SMAPE(%)↓ |
|---|---|---|---|---|
| 10 | 0.42 | 0.59 | 74.1 | 71.2 |
| 15 | 0.45 | 0.54 | 73.5 | 70.7 |
| 20 | 0.49 | 0.43 | 71.8 | 70.2 |
| 25 | 0.53 | 0.41 | 68.5 | 65.6 |
| 30 | 0.57 | 0.39 | 65.1 | 64.3 |
| 35 | 0.59 | 0.33 | 58.6 | 57.8 |
| 40 | 0.65 | 0.27 | 51.5 | 50.8 |
| 45 | 0.69 | 0.21 | 47.5 | 39.7 |
| 50 | 0.76 | 0.17 | 31.2 | 28.7 |
| 55 | 0.77 | 0.17 | 32.3 | 29.6 |
| 60 | 0.75 | 0.15 | 31.6 | 27.8 |

Table 2: Comparative performance of the proposed approach against different window size for 2-layer architecture. Lower value(↓) is considered the better.

size increases till a stabilization point $\tau = 50$. Ideally one would expect that the results would improve as the window size increases; especially for deeper

| Window size($\tau$) | $r$ | RMSE↓ | MAPE(%)↓ | SMAPE(%)↓ |
|---|---|---|---|---|
| 10 | 0.43 | 0.59 | 0.76 | 0.72 |
| 15 | 0.47 | 0.56 | 0.71 | 0.71 |
| 20 | 0.48 | 0.51 | 0.67 | 0.65 |
| 25 | 0.53 | 0.47 | 0.61 | 0.62 |
| 30 | 0.59 | 0.42 | 0.59 | 0.57 |
| 35 | 0.64 | 0.35 | 0.54 | 0.51 |
| 40 | 0.69 | 0.27 | 0.43 | 0.42 |
| 45 | 0.73 | 0.21 | 0.39 | 0.33 |
| 50 | 0.81 | 0.12 | 29.1 | 21.2 |
| 55 | 0.82 | 0.14 | 29.3 | 20.3 |
| 60 | 0.81 | 0.13 | 28.4 | 20.4 |

Table 3: Comparative performance of the proposed approach against different window size for 3-layer architecture. Lower value(↓) is considered the better.

versions. This is because larger window size means more data and hence better generalisation ability. But note that it is a dynamical model. The tacit assumption here is that in the window the size the underlying dynamical function does not change. While this is true for shorter windows, this does not hold for larger ones as the non-stationarity comes into play. This is the reason we find a trade-off between window size and accuracy. We further set this value of window size in upcoming experiments.

*5.4.2. Comparison with state-of-the-art methods*

The overall performance of the model vis-a-vis the state-of-the-art is shown in Table4. The Table presents the average results for ten cryptocurrencies; owing to limitations in space we are not able to show individual results. One can see that the proposed method outperforms the baseline by a considerable margin. Fig 5.2 shows the forecast performance of DeCrypt with other baseline methods for visual evaluation and Fig 3 shows the convergence plots for model parameters for different layer of architecture for DeCrypt where $\mathbf{T}_1$ is the

| Model | $r$ | RMSE↓ | MAPE(%)↓ | SMAPE(%)↓ |
|---|---|---|---|---|
| LSTM | 0.33 | 0.71 | 83.2 | 64.3 |
| CNN-TA | 0.29 | 0.68 | 91.2 | 70.8 |
| ARIMA | 0.29 | 0.68 | 91.2 | 70.8 |
| Rao-ANN | 0.29 | 0.68 | 91.2 | 70.8 |
| MFNN | 0.31 | 0.21 | 70.28 | 68.2 |
| N-Beats | 0.38 | 0.27 | 41.2 | 38.72 |
| DeepAR | 0.30 | 0.39 | 46.8 | 41.47 |
| TFT | 0.52 | 0.24 | 48.95 | 40.62 |
| RDL | 0.69 | 0.20 | 48.7 | 35.68 |
| ARIMA | 0.58 | 0.49 | 56.4. | 46.24 |
| Rao-ANN | 0.42 | 0.59 | 68.7 | 64.38 |
| DeCrypt (1 layer) | 0.73 | 0.21 | 35.7 | 32.33 |
| DeCrypt (2 layers) | 0.76 | 0.17 | 31.2 | 28.7 |
| DeCrypt (3 layers) | **0.81** | **0.12** | **29.14** | **21.2** |

Table 4: Comparative performance of the proposed approach against baseline methods. Lower value(↓) is considered the better.

multi-linear state operator achieved from product of three positive-valued linear factors $\mathbf{T}_{10}\mathbf{T}_{11}\mathbf{T}_{12}$, $\mathbf{T}_2$ is the is the multi-linear control operator achieved from product of three positive-valued linear factors $\mathbf{T}_{20}\mathbf{T}_{21}\mathbf{T}_{22}$ and $\mathbf{D}$ is the multi-linear observation operator achieved from product of three positive-valued linear factors $\mathbf{D}_0\mathbf{D}_1\mathbf{D}_2$. It can be clearly seen that the *DeCrypt* with its three layers architecture outperforms the other baseline approaches significantly. This is mainly because of the deeper network's capacity to better model non-linearity compared to the shallower ones. The model achieves a 0.17 points drop in RMSE, 19.56% drop in MAPE, and 14.48% drop in SMAPE; it gains 0.12 in Pearson's correlation $r$ compared to the best performing benchmarks. We have plotted the error-bars for four different cryptocurrencies (Litecoin, Dogecoin, Ethereum, Gridcoin)in Fig.5.2. From these plots the reader can verify that not

only is the proposed method more accurate (least mean error) but is also the most robust (least deviation). In Table 6 we present the comparison of performance of proposed approach with state-of-the-art method through statistical test (T-test) with confidence interval of 0.95. We can observe that the T-test values for proposed approach DeCrypt (3 layers) is very small as compared to the other methods, hence we can conclude that more similarity exists between the actual closing prices and predicted closing prices when compared for different Crypto-currencies. Due to space constraints we have provided Avg. Score for t-test for all the ten cryptocurrency for each method in Table 6. From the results we conclude that average T-test score is very low for Decrypt (3 layers) method when compared with other state-of-the-art method, hence we conclude that Decrypt (3 layers) performance is very close to ground truth. In contrast when we see individual crypto-currency analysis from table we can see that TFT outperforms all the methods in Gridcoin and DeepAR outperforms Dogecoin.

| Method | Train Time cost (h.) | Test Time cost (min.) |
|---|---|---|
| DeCrypt (3 layers) | 2.21h | 22 min |
| DeCrypt (2 layers) | 2.32h | 22.4 min |
| DeCrypt (1 layer) | 1.48h | 18.8 min |
| ARIMA | 2.31h | 36 min |
| LSTM | 5 days | 41 min |
| DeepAR | 2.45h | 20 min |
| TFT | 2.25h | 27 min |
| Nbeats | 3.12h | 25 min |
| CNN-TA | 4.57h | 40 min |
| MFNN | 4.12h | 37 min |
| RDL | 1.69h | 35 min |
| Rao-ANN | 4.35h | 25 min |

Table 5: Averaged time over 50 random runs for processing the dataset (train(hrs) and test(min)), for the proposed approach and its competitors.

To understand the comparison in performance between the proposed method

| Method | Bitcoin | Gridcoin | Dogecoin | Litecoin | Avg. Score |
|---|---|---|---|---|---|
| DeCrypt (3 layers) | 0.31 | 0.73 | 0.82 | 0.57 | 0.68 |
| DeCrypt (2 layers) | 0.56 | 0.85 | 0.91 | 0.69 | 0.72 |
| DeCrypt (1 layer) | 0.58 | 0.98 | 0.99 | 0.72 | 0.78 |
| ARIMA | 0.90 | 0.87 | 0.95 | 0.93 | 0.83 |
| LSTM | 0.59 | 1.75 | 0.97 | 0.82 | 0.94 |
| DeepAR | 0.54 | 0.79 | 0.81 | 0.55 | 0.71 |
| TFT | 0.40 | 0.70 | 1.12 | 0.71 | 0.73 |
| Nbeats | 0.56 | 0.68 | 0.99 | 0.61 | 0.84 |
| CNN-TA | 0.87 | 1.53 | 2.11 | 1.19 | 1.13 |
| MFNN | 0.82 | 1.30 | 1.21 | 0.74 | 1.37 |
| RDL | 0.58 | 0.95 | 0.94 | 0.68 | 0.84 |
| Rao-ANN | 0.54 | 0.70 | 1.02 | 1.13 | 0.94 |

Table 6: Comparison of T-test score for the proposed approach with state-of-the-art method for (a) Bitcoin, (b) Gridcoin, (c) Dogecoin, (d) Litecoin, (e)Avg. Score for all the ten crypto-currencies.

and state-of-the-art methods, we present Table 5 which depicts the computational time for forecasting the next day closing price of ten cryptocurrencies. We provide a comprehensive analysis by distinguishing the time required to train and test the methods (on their training and testing time frame as described in sec 5.1) using the walk-forward method described in (Sharma et al., 2021, Section 4.2.1). We conclude that the highest computational time was consumed by LSTM approach. Among the other methods, DeCrypt (1 layer) and RDL method is the fastest while the computation time of DeCrypt (3 layers) is very much comparable to DeepAR and TFT. However, note that the existing algorithms are optimized to take advantage of GPU, the proposed approach does not, it runs only on the CPU. It may be possible to improve the performance in the future through parallelization.

| Cryptocurrency | DeCrypt (1 layer) | DeCrypt (2 layer) | DeCrypt (3 layers) | CVI (DeCrypt*)) | CVI (Nbeats) |
|---|---|---|---|---|---|
| Bitcoin | 0.79 | **0.67** | 0.86 | **0.31** | 0.48 |
| Dogecoin | **4.32** | 4.89 | 4.75 | **2.62** | 2.71 |
| Namecoin | 4.34 | 3.79 | **3.65** | 1.61 | 1.81 |
| Litecoin | 1.21 | 1.10 | **0.92** | **0.35** | 0.49 |
| Gridcoin | 1.81 | 1.56 | **1.45** | 0.59 | 0.63 |
| Peercoin | 1.67 | 1.43 | **1.23** | 0.50 | 0.59 |
| Ripple | 1.24 | 1.11 | **0.97** | **0.42** | 0.53 |
| NXT | 1.32 | 1.10 | **0.91** | **0.32** | 0.42 |
| Ethereum | 1.56 | **1.41** | 1.43 | **0.38** | 0.63 |
| Binance coin | 1.34 | 1.21 | **0.84** | **0.33** | 0.72 |

Table 7: (Un)certainty quantification (log-loss) and Cryptocurrency Volatility Index (CVI) evaluated using DeCrypt (3 layers) and Nbeats

| Cryptocurrency | LSTM | CNN-TA | ARIMA | Rao-ANN | MFNN | DeepAR | TFT | RDL |
|---|---|---|---|---|---|---|---|---|
| Bitcoin | 0.57 | 0.53 | 0.46 | 0.59 | 0.72 | 0.52 | 0.49 | 0.38 |
| Dogecoin | 2.87 | 3.27 | 2.35 | 3.43 | 3.87 | 2.72 | 2.83 | 2.57 |
| Namecoin | 1.93 | 2.12 | 1.88 | 1.83 | 1.96 | **1.58** | 1.68 | 1.73 |
| Litecoin | 1.31 | 0.56 | 0.48 | 0.51 | 0.67 | 0.38 | 0.46 | 0.41 |
| Gridcoin | 0.99 | 0.87 | 0.67 | 0.74 | 0.96 | 0.68 | 0.75 | **0.53** |
| Peercoin | 0.81 | 0.78 | **0.48** | 0.69 | 0.93 | 0.64 | 0.71 | 0.58 |
| Ripple | 0.78 | 0.83 | 0.64 | 0.73 | 0.88 | 0.61 | 0.68 | 0.54 |
| NXT 0.57 | 0.63 | 0.46 | 0.53 | 0.64 | 0.58 | 0.54 | 0.59 | 0.47 |
| Ethereum | 0.98 | 0.78 | 0.43 | 0.58 | 0.49 | 0.52 | 0.61 | 0.45 |
| Binance coin | 0.53 | 0.64 | 0.41 | 0.49 | 0.51 | 0.45 | 0.54 | 0.39 |

Table 8: Cryptocurrency Volatility Index (CVI) evaluated using state-of-the-art method predictions

*5.4.3. Uncertainty quantification*

The advantage of DeCrypt over other baseline approaches is the estimation of (un)certainty quantification associated with each prediction. For cryptocurrencies this measure will be directly proportional to the volatility index. As discussed in section *Uncertainty Quantification*, it is easy to evaluate (un)certainty of prediction of an increase/decrease of price forecast by calculating the log-loss penalization as explained in eq. 18. To validate the proposed method results on (un)certainty quantification the work also evaluated cryptocurrency volatility index for each cryptocurrency. Cryptocurrecy Volatility Index (CVI) (Kim et al., 2021; Woebbeking, 2021)can be defined as a measure of market's expectation of volatility over the near trading terms for a particular asset. Volatility is often described as the "rate and magnitude of changes in prices" and in finance often referred to as risk. Volatility is sometimes associated with the uncertainty of risk related to the amount of changes in security's value. This can be further described as if the security's value can potentially be spread out over a larger range of values, it indicates that the price of the security can change dramatically over a short time period in either direction which is flagged as higher volatility. On the other hand, A lower volatility means that a security's value does not fluctuate dramatically, and tends to be more steady. The mathematical formula to calculate CVI (Woebbeking, 2021):

$$\text{CVI} = \sqrt{365} * \sqrt{\frac{1}{N}\sum_{N=1}^{N}((Closeprice - PriceatN)^2))}, \tag{19}$$

Table 7 depicts the calculated log-loss values for each cryptocurrency vis-a-vis their cryptocurrency volatility index (CVI)[4].Table 7 represents the log loss score for all the layer architecture for DeCrypt and CVI score for DeCrypt (3 layers) and Nbeats. Table 8 represents the CVI scores from state-of-the-art method. A smaller value of the loss should be associated with lower volatility and vice versa. It can be clearly seen that log-loss associated with the Bitcoin,

[4]`https://github.com/dc-aichara/PriceIndices`

Litecoin, Peercoin, Ripple, NXT, Binance coin is less than one, meaning prices associated with these cryptocurrencies are less volatile. In contrast, Dogecoin, which is highly volatile and has a history of spiked values after a tweet by a major influencer, is more difficult to assess and has a log-loss score of 4.75. Thus one can see how the proposed algorithm can quantify uncertainty and how this measure is proportionate to the oracle volatility. This is by far the most important result in the paper. This result shows how the proposed approach may be used for practical trading where both the point estimate as well as the uncertainty about the estimate is required for making decisions.

Owing to limitations in space, unfortunately not able to show the convergence of the proposed algorithm or the run-times of different techniques. Although the work have used 50 iterations and its empirically checked that the proposed algorithm converges in about 20-25 iterations. The convergence is monotonic. In terms of speed the proposed method is about 2-4 times faster than LSTM, CNN-TA and MFNN, and is about an order of magnitude faster than TFT, Nbeats and DeepAR. Of the existing methods, only RDL is comparable to the proposed method in terms of speed.

### *5.5. Discussion*

Consistently beating in modeling Time series signals has been challenging for a long time. The proposed three-layer architecture method performed better than the current state-of-the-art method. The proposed method is based on the deep state-space and feedback strategy. As can be seen from Fig. 5.2 the proposed model can predict the sudden spikes in the data compared to other state-of-the-art methods. This is mainly because the proposed method is based on SSM, which uses probabilistic predictive distribution to estimate the future state of the price trajectories. The technique also embeds deep non-negative factors to learn the model parameters. Deep factors helps in updating the model parameters and state-space of unseen signals continuously with time, in contrast to machine learning models, which use a huge amount of data to learn approximations. We observed that when the time series signals grow,

these models suffer from vanishing gradient and exploding gradient problems, which hampers learning model parameters. Due to this, model approximations are not learned efficiently and cannot capture sudden spikes (highs and lows in prices). These models also suffer from over-fitting. The proposed method avoids over-fitting as we move ahead in the sliding window protocol, and previously updated model parameters are used as initialized values for the next window parameters. We have also presented a comprehensive analysis of empirical and statistical performance. When Table 4 and Table 6 are analyzed, it is observed that the proposed method performance is outstanding in 3 out of 4 crypto-currency presented, and its average score for ten crypto-currency is smallest as compared to other state-of-the-art methods hence we can conclude that more similarity exists between the actual closing prices and predicted closing prices when compared for different Crypto-currencies. The proposed method can be applied to various other prediction applications where its challenging to model unseen volatile data, such as short-term load monitoring, sales and revenue prediction, and predicting hate intensity for social media content.

## 6. Acknowledgement

The authors are indebted for the support provided by the Infosys Center for Artificial Intelligence at Indraprastha Institute of Information Technology-Delhi,(IIIT Delhi).

## 7. Conclusion

The current study forecasts the prices of crypto-currencies; this is halfway to the goal. The final objective is to take trading positions (BUY / SELL / HOLD). But that is a very challenging problem where strategies come into play. Unfortunately successful investors do not reveal their strategy. Pedagogically it is a matured area in traditional financial markets (Molinero & Riquelme, 2021) where game theory is mainly used in arriving at the decision boundaries; however how effective they are in practice is not known. Currently, a combination of

game theory and social network analysis is used for arriving at such decision boundaries (Molinero & Riquelme, 2021). In future, the authors would like to see if such cues from stock trading that can be used for maximising returns in crypto-currency trading.

## References


Abu Bakar, N., & Rosbi, S. (2017). Autoregressive integrated moving average (arima) model for forecasting cryptocurrency exchange rate in high volatility environment: A new insight of bitcoin transaction. *International Journal of Advanced Engineering Research and Science*, *4*, 130–137.

Andersen, T. G., Davis, R. A., Kreiß, J.-P., & Mikosch, T. V. (2009). *Handbook of Financial Time Series*. Springer Science & Business Media.

Andrieu, C., Doucet, A., & Holenstein, R. (2010). Particle markov chain monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *72*, 269–342.

Baek, Y., & Kim, H. Y. (2018). Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention lstm module and a prediction lstm module. *Expert Systems with Applications*, *113*, 457–480.

Catania, L., Grassi, S., & Ravazzolo, F. (2018). Predicting the volatility of cryptocurrency time-series. *Mathematical and Statistical Methods for Actuarial Sciences and Finance*, (pp. 203–207).

Chen, M., Jiang, H., Liao, W., & Zhao, T. (2019). Efficient approximation of deep relu networks for functions on low dimensional manifolds. *Advances in Neural Information Processing Systems*, *32*.

Chen, Z., Jin, S., Liu, R., & Zhang, J. (2021). A deep non-negative matrix factorization model for big data representation learning. *Frontiers in Neurorobotics*, *15*. URL: `https://www.frontiersin.org/article/10.3389/fnbot.2021.701194`. doi:`10.3389/fnbot.2021.701194`.

Chopin, N., Jacob, P. E., & Papaspiliopoulos, O. (2013). SMC2: an efficient algorithm for sequential analysis of state space models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *75*, 397–426.

Chouzenoux, E., & Elvira, V. (2023). Graphit: Iterative reweighted l1 algorithm for sparse graph inference in state-space models. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1–5). IEEE.

Chouzenoux, E., Pesquet, J.-C., & Repetti, A. (2016). A block coordinate variable metric forward-backward algorithm. *Journal of Global Optimization*, *66*, 457–485.

Cox, B., & Elvira, V. (2023). Sparse Bayesian estimation of parameters in linear-gaussian state-space models. *IEEE Transactions on Signal Processing (to appear in)*, .

Crisan, D., & Miguez, J. (2018). Nested particle filters for online parameter estimation in discrete-time state-space markov models. *Bernoulli*, *24*, 3039–3086.

Daubechies, I., DeVore, R., Foucart, S., Hanin, B., & Petrova, G. (2022). Nonlinear approximation and (deep) relu networks. *Constructive Approximation*, *55*, 127–172.

De Handschutter, P., Gillis, N., & Siebert, X. (2021). A survey on deep matrix factorizations. *Computer Science Review*, *42*, 100423.

Derbentsev, V., Matviychuk, A., & Soloviev, V. N. (2020). Forecasting of cryptocurrency prices using machine learning. In *Advanced Studies of Financial Technologies and Cryptocurrency Markets* (pp. 211–231). Springer.

Digalakis, V., Rohlicek, J. R., & Ostendorf, M. (1993). ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition. *IEEE Transactions on Speech and Audio Processing*, *1*, 431–442.

Dritsaki, C. (2015). Box-Jenkins modeling of Greek stock prices data. *International Journal of Economics and Financial Issues*, *5*.

Elbrächter, D., Perekrestenko, D., Grohs, P., & Bölcskei, H. (2021). Deep neural network approximation theory. *IEEE Transactions on Information Theory*, *67*, 2581–2623.

Elsworth, S., & Güttel, S. (2020). Time series forecasting using LSTM networks: A symbolic approach. *https://arxiv.org/abs/2003.05672*, .

Elvira, V., & Chouzenoux, E. (2022). Graphical inference in linear-Gaussian state-space models. *IEEE Transactions on Signal Processing*, *70*, 4757–4771.

Elvira, V., Míguez, J., & Djurić, P. M. (2017). Adapting the number of particles in sequential monte carlo methods through an online scheme for convergence assessment. *IEEE Transactions on Signal Processing*, *65*, 1781–1794.

Flenner, J., & Hunter, B. (2017). A deep non-negative matrix factorization neural network. *Semantic Scholar*, .

Garzon, M., & Botelho, F. (1999). Dynamical approximation by recurrent neural networks. *Neurocomputing*, *29*, 25–46.

Glenski, M., Weninger, T., & Volkova, S. (2019). Improved forecasting of cryptocurrency price using social signals. *https://arxiv.org/abs/1907.05058*, .

Hammer, B. (2000). On the approximation capability of recurrent neural networks. *Neurocomputing*, *31*, 107–123.

Jacobson, M., & Fessler, J. (2007). An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms. *IEEE Transactions on Image Processing*, *16*, 2411–2422.

Jarrett, J. E., & Kyper, E. (2011). Arima modeling with intervention to forecast and analyze chinese stock prices. *International Journal of Engineering Business Management*, *3*, 53–58.

Kim, A., Trimborn, S., & Härdle, W. K. (2021). Vcrix—a volatility index for crypto-currencies. *International Review of Financial Analysis*, *78*, 101915.

Kim, T., & Kim, H. Y. (2019). Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PloS one*, *14*, e0212320.

Köchling, G., Schmidtke, P., & Posch, P. N. (2020). Volatility forecasting accuracy for bitcoin. *Economics Letters*, *191*, 108836.

Kraaijeveld, O., & De Smedt, J. (2020). The predictive power of public twitter sentiment for forecasting cryptocurrency prices. *Journal of International Financial Markets, Institutions and Money*, *65*, 101188.

Kristjanpoller, W., & Minutolo, M. C. (2018). A hybrid volatility forecasting framework integrating garch, artificial neural network, technical analysis and principal components analysis. *Expert Systems with Applications*, *109*, 1–11.

Liang, S., & Srikant, R. (2016). Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*, .

Lim, B., Arik, S. Ö., Loeff, N., & Pfister, T. (2019). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *CoRR*, *abs/1912.09363*. URL: `http://arxiv.org/abs/1912.09363`. `arXiv:1912.09363`.

Lin, T., Ma, S., & Zhang, S. (2015). On the global linear convergence of the admm with multiblock variables. *SIAM Journal on Optimization*, *25*, 1478–1497.

Liu, B., & Liang, Y. (2021). Optimal function approximation with relu neural networks. *Neurocomputing*, *435*, 216–227.

Livieris, I. E., Kiriakidou, N., Stavroyiannis, S., & Pintelas, P. (2021). An advanced cnn-lstm model for cryptocurrency forecasting. *Electronics*, *10*, 287.

Long, W., Lu, Z., & Cui, L. (2019). Deep learning-based feature engineering for stock price movement prediction. *Knowledge-Based Systems*, *164*, 163–173.

Ma, F., Liang, C., Ma, Y., & Wahab, M. (2020a). Cryptocurrency volatility forecasting: A markov regime-switching midas approach. *Journal of Forecasting*, *39*, 1277–1290.

Ma, X., Karkus, P., Hsu, D., & Lee, W. S. (2020b). Particle filter recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 5101–5108). volume 34.

Mahdizadehaghdam, S., Panahi, A., Krim, H., & Dai, L. (2019). Deep dictionary learning: A parametric network approach. *IEEE Transactions on Image Processing*, *28*, 4790–4802.

Mei, J., De Castro, Y., Goude, Y., Azaïs, J.-M., & Hébrail, G. (2019). Nonnegative matrix factorization with side information for time series recovery and prediction. *IEEE Transactions on Knowledge and Data Engineering*, *31*, 493–506. doi:10.1109/TKDE.2018.2839678.

Molinero, X., & Riquelme, F. (2021). Influence decision models: from cooperative game theory to social network analysis. *Computer Science Review*, *39*, 100343.

Montella, C. (2011). The kalman filter and related algorithms: A literature review. *Res. Gate*, (pp. 1–17).

Nayak, S. K., Nayak, S. C., & Das, S. (2021). Modeling and forecasting cryptocurrency closing prices with rao algorithm-based artificial neural networks: A machine learning approach. *FinTech*, *1*, 47–62.

Newman, K., King, R., Elvira, V., de Valpine, P., McCrea, R. S., & Morgan, B. J. (2023). State-space models for ecological time-series data: Practical model-fitting. *Methods in Ecology and Evolution*, *14*, 26–42.

Nishihara, R., Lessard, L., Recht, B., Packard, A., & Jordan, M. (2015). A general analysis of the convergence of admm. In *International Conference on Machine Learning* (pp. 343–352). PMLR.

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2019). N-beats: Neural basis expansion analysis for interpretable time series forecasting. *arXiv preprint arXiv:1905.10437*, .

R. Molla (2021). When Elon Musk tweets, crypto prices move. *https://www.vox.com/recode/2021/5/18/22441831/elon-musk-bitcoin-dogecoin-crypto-prices-tesla*, .

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., & Januschowski, T. (2018). Deep state space models for time series forecasting. *Advances in Neural Information Processing Systems*, *31*, 7785–7794.

Rankin, J. (1986). *Kalman filtering approach to market price forecasting* . Ph.D. diss. Iowa State University.

Rounaghi, M. M., & Zadeh, F. N. (2016). Investigation of market efficiency and financial stability between s&p 500 and london stock exchange: monthly and yearly forecasting of time series stock returns using arma model. *Physica A: Statistical Mechanics and its Applications*, *456*, 10–21.

S. Soni (2021). Crypto investors lost $748 billion in last seven days as bitcoin, ethereum, dogecoin, others declined. `https://www.financialexpress.com/market/crypto-investors-lost-748-billion-in-last/seven-days-as-bitcoin-ethereum-dogecoin-others-declined`. Accessed: 2021-05-23.

Salinas, D., Flunkert, V., Gasthaus, J., & Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, *36*, 1181–1191.

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*. (3rd ed.). Cambridge University Press.

Sezer, O. B., & Ozbayoglu, A. M. (2018). Algorithmic financial trading with deep convolutional neural networks: Time series to image conversion approach. *Applied Soft Computing*, *70*, 525–538.

Sharma, S., Elvira, V., Chouzenoux, E., & Majumdar, A. (2021). Recurrent dictionary learning for state-space models with an application in stock forecasting. *Neurocomputing*, *450*, 1–13.

Sharma, S., & Majumdar, A. (2021). Sequential transform learning. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *15*, 1–18.

Sharma, S., Majumdar, A., Elvira, V., & Chouzenoux, E. (2020). Blind kalman filtering for short-term load forecasting. *IEEE Transactions on Power Systems*, *35*, 4916–4919.

Shumway, R. H., & Stoffer, D. S. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, *3*, 253–264.

Tariyal, S., Majumdar, A., Singh, R., & Vatsa, M. (2016). Deep dictionary learning. *IEEE Access*, *4*, 10096–10109.

Trigeorgis, G., Bousmalis, K., Zafeiriou, S., & Schuller, B. W. (2016). A deep matrix factorization method for learning attribute representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*, 417–429.

Walther, T., Klein, T., & Bouri, E. (2019). Exogenous drivers of bitcoin and cryptocurrency volatility–a mixed data sampling approach to forecasting. *Journal of International Financial Markets, Institutions and Money*, *63*, 101133.

Wang, Y., Yin, W., & Zeng, J. (2019). Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing*, *78*, 29–63.

Woebbeking, F. (2021). Cryptocurrency volatility markets. *Digital Finance*, *3*, 273–298.

Yarotsky, D. (2018). Optimal approximation of continuous functions by very deep relu networks. In *Proceedings of the Conference on Learning Theory (COLT 2018)* (pp. 639–649). PMLR.

Yasir, M., Attique, M., Latif, K., Chaudhary, G. M., Afzal, S., Ahmed, K., & Shahzad, F. (2020). Deep-learning-assisted business intelligence model for cryptocurrency forecasting using social media sentiment. *Journal of Enterprise Information Management*, .

Ye, R., & Dai, Q. (2022). A relationship-aligned transfer learning algorithm for time series forecasting. *Information Sciences*, *593*, 17–34.