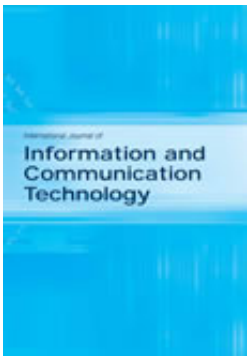


International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

Recurrent neural network optimisation based on linearly constrained numerical methods

Wenmin Song, Wei Han, Ping Gu, Min Li

DOI: 10.1504/IJICT.2025.10071630

Article History:

Received:	15 April 2025
Last revised:	29 April 2025
Accepted:	29 April 2025
Published online:	20 June 2025

Recurrent neural network optimisation based on linearly constrained numerical methods

Wenmin Song*, Wei Han, Ping Gu and Min Li

Department of Artificial Intelligence,
Laiwu Vocational and Technical College,
Laiwu 271199, China
Email: kelemi@163.com
Email: 21076018@qq.com
Email: guping0127@163.com
Email: xslxlm19870426@126.com
*Corresponding author

Abstract: Time-series data analysis has grown even more crucial in many sectors as information technology and big data expand rapidly. This work proposes a recurrent neural network (RNN) optimisation model based on the linear constraint numerical method, namely, LSTM-LP optimiser, which combines the powerful time-series modelling capability of long short-term memory (LSTM) and the optimisation characteristics of linear programming (LP) optimisation features, and so effectively improves the training efficiency and stability of the model in resource-constrained environments. This helps to efficiently capture the temporal dependencies in time-series data and solve the noise and missing problems in the data. On two datasets, experimental results show the LSTM-LP optimiser beats the conventional model in several performance criteria. Future studies will investigate more effective optimisation techniques, increase the generalisation capacity of the model, and simplify the hyperparameter tweaking process to thus further promote the model in practical uses.

Keywords: time-series data analysis; recurrent neural network; RNN; linear programming; LP; long short-term memory; LSTM; resource-constrained optimisation.

Reference to this paper should be made as follows: Song, W., Han, W., Gu, P. and Li, M. (2025) 'Recurrent neural network optimisation based on linearly constrained numerical methods', *Int. J. Information and Communication Technology*, Vol. 26, No. 21, pp.70–86.

Biographical notes: Wenmin Song received her Master's degree from Yanshan University in 2008. She is currently an Associate Professor at the Laiwu Vocational and Technical College. Her research interests include artificial intelligence, big data and computer networks.

Wei Han received her Master's degree from Qingdao University in 2009. She is currently a Lecturer at the Laiwu Vocational and Technical College. Her research interests include artificial intelligence and information security.

Ping Gu received her Master's degree at Tianjin University of Technology in 2014. She is currently a Lecturer at the Laiwu Vocational and Technical College. Her research interests include artificial intelligence and cloud computing.

Min Li received her Master's degree from Inner Mongolia University of Technology in 2013. She is currently a Teaching Assistant at the Laiwu Vocational and Technical College. Her research interests include artificial intelligence and big data.

1 Introduction

The fast growth of information technology and big data has resulted in a lot of time-series data accumulated in many sectors (Wen et al., 2020). Time-series data are widespread and indispensable in many sectors, from the prediction of stock market swings in the financial market to weather forecasting, energy consumption management, traffic flow analysis and other fields (Bhogade and Nithya, 2024). Time-series data shows the dynamic properties of objects over time, therefore how to extract useful information from these data cannot only better comprehend the evolutionary principles of the system, but also enhance resource optimisation and risk management and help decision making.

Analysing time-series data presents difficulties mostly related to its intricate time-dependent interactions. First of all, time-series data typically have long-term dependencies – that is, the current state has a complicated relationship with the states at several times in the past that cannot be properly managed by conventional machine learning techniques. Second, in time-series data, noise, missing values, and external disruptions often complicate data processing. Consequently, the main challenge in time-series data analysis is now how effectively to capture the temporal dependencies in time-series data and eliminate the noise and missing issues in the data.

Deep learning technology has evolved RNN into one of the primary models for time-series data processing (Han et al., 2019). By using the hidden state of the past instant as the input of the current moment, RNN can effectively grasp long-term dependencies in time-series data. But when processing extended sequences, typical RNN suffers from the issue of gradient vanishing or gradient explosion, therefore restricting their performance over large times spans. LSTM originated to help us solve these challenges. By including forgetting gate, input gate, and output gate, LSTM efficiently solves the gradient vanishing problem and can retain the critical information over a long time span (Landi et al., 2021). Wide-ranging applications for LSTM abound in natural language processing, speech recognition, machine translation, etc.; it has produced amazing results. LSTM models still suffer with high computational complexity and high resource consumption in the training process, as well as insufficient capacity to manage constraints in the model.

Apart from deep learning techniques, conventional machine learning systems find extensive use in time-series data analysis. Common classic techniques for classification and regression of time-series data are support vector machines (SVM) and decision trees (Fawaz et al., 2019). Suitable for smaller linear issues, support vector regression (SVR) finds an ideal hyperplane to match time-series data (Luo et al., 2019). Conversely, decision trees fit for handling noisy or high number of missing values since they partition the feature space recursively. Nevertheless, when handling long time dependencies and complicated constraints, the performance of these conventional methods is more constrained; so, it is usually difficult to handle the issues of big-scale time-series data.

More and more study has started to try to mix optimisation techniques with deep learning models in order to offset the drawbacks of deep learning and conventional machine learning algorithms when dealing with time-series data (Ahmed et al., 2023). By means of linear constraints to guarantee the computational efficiency of the model and to guarantee its performance under certain constraints, LP, as a traditional optimisation technique, is able to efficiently restrict the model training process. Linear programming can assist the model to better satisfy the needs in actual applications while handling time-series data analysis problems with specific constraints.

Aiming to tackle the performance bottleneck of current approaches when confronted with challenging restrictions, this work suggests the LSTM-LP optimiser model, which combines the time-series modelling capabilities of LSTM with the optimisation features of LP.

This paper's innovations consist in the following:

1 An RNN model (LSTM-LP optimiser) incorporating linear constrained optimisation is proposed: combining the time-series modelling capacity of LSTM with the optimisation characteristics of LP, this combination not only effectively captures the long-term dependencies in the time-series data, but also guarantees the computational efficiency and the rationality of resource usage through linear constraint optimisation during model training, so solving the problem of the limited performance of the current deep learning models under complex constraints.

2 A linear constraint optimisation mechanism is introduced in model training: this mechanism not only guarantees the stability and high efficiency of model training by dynamically adjusting the hidden layer state of LSTM to meet the preset resource constraints, but also guarantees the adaptability of the model under resource-constrained environments by means of new solution for resource management in time-series data analysis.

3 The performance advantages of the model on different datasets are verified through experiments: LSTM-LP optimiser beats conventional and alternative deep learning models in several performance metrics, particularly in managing complicated constraints, which offers a better model choice for time-series prediction activities, experimental results on power consumption and traffic flow datasets reveal.

2 Relevant technologies

2.1 Linearly constrained numerical methods

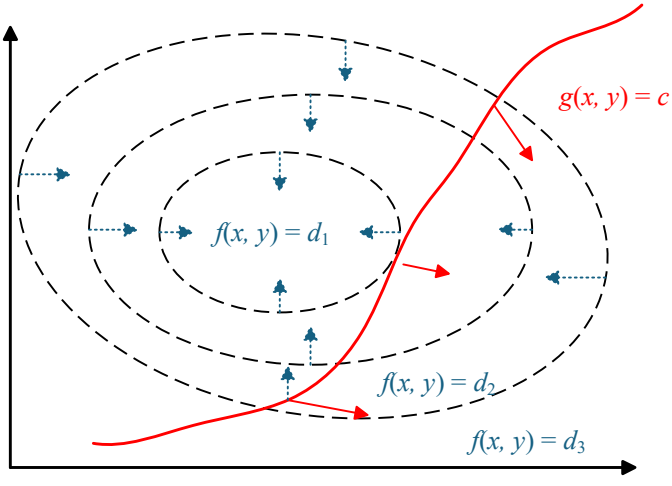
Many practical optimisation problems arise in situations with linear constraints (Necoara et al., 2019). Usually, a linearly constrained optimisation problem can be stated as follows:

$$\min_{x} f(x) \quad \text{subject to } Ax \le b, Cx = d \tag{1}$$

Usually a scalar function on the decision variable x, $f(x)$ is the objective function; A and C are the coefficient matrices establishing the equality constraints and the inequality

constraints respectively, and b and d are the associated constraint vectors. See Figure 1 for an optimal selection of x such that the objective function $f(x)$ takes the smallest value while meeting all linear constraints.

Figure 1 Linear constraint algorithm (see online version for colours)



Commonly used techniques for solving these types of optimisation issues are simplex method, interior point method and Lagrange multiplier approach (Nie et al., 2021). The fundamental concept of the Lagrange multiplier method, a basic approach for handling optimisation issues with constraints, is to include constraints into the objective function, therefore converting the initial constrained optimisation problem into an unconstrained optimisation problem. We create a Lagrangian function to accomplish this. Particularly for the aforementioned optimisation issue, Lagrange multipliers λ and μ can be introduced, respectively corresponding to the equality and inequality requirements respectively, and the Lagrange functions can be built:

$$L(x, \lambda, \mu) = f(x) + \lambda^{\top}(Ax - b) + \mu^{\top}(Cx - d) \tag{2}$$

where λ and μ help to modify the goal function so that the constraints are satisfied. Solving the gradient of the Lagrangian function and hence bringing it equal to zero will provide the ideal solution (Jin et al., 2022). More especially, three equations are needed to be solved specifically:

$$\nabla_x L(x, \lambda, \mu) = 0 \tag{3}$$

$$\nabla_\lambda L(x, \lambda, \mu) = 0 \tag{4}$$

$$\nabla_\mu L(x, \lambda, \mu) = 0 \tag{5}$$

These equations satisfy the Karush-Kuhn-Tucker (KKT) requirements, therefore defining the optimality conditions for the optimisation problem (Prado et al., 2023). Regarding inequality restrictions, the KKT conditions demand:

$$Ax \le b, \lambda_i (Ax - b)_i = 0, \quad \forall i \tag{6}$$

where complementary relaxation calls for the multipliers to be proportional to the residuals of the constraints and the Lagrange multipliers to be non-zero only if the inequality constraints are strictly activated; else, the corresponding Lagrange multipliers are zero. Regarding the inequality restrictions, the KKT condition demands that:

$$Cx = d \tag{7}$$

Solving this system of equations can help one to find the best solution for the optimisation issue.

Along with the Lagrange multiplier technique, LP is a widely used numerical solution approach in linear restricted optimisation problems (Liu et al., 2023). A linear programming problem has its standard form as:

$$\min_x c^\top x \quad \text{subject to } Ax \le b, Cx = d \tag{8}$$

where A and x are respectively the matrix of coefficients of the restrictions and the choice variables and c is the vector of coefficients of the objective function. Solving linear programming problems with the straightforward method is a traditional and extensively applied classical method. The basic concept of the simplex technique is to iterate along the constraints' borders, and each step chooses a new viable solution in the neighbourhood of the present feasible solution until the optimal solution is obtained.

The straightforward approach specifically iterates over the following update formula:

$$x_{k+1} = x_k + \alpha_k p_k \tag{9}$$

where α_k denotes the step size, therefore indicating the distance covered in the direction p_k. With each update, the simplex technique advances towards the optimal solution until it is discovered.

The interior point approach is yet another useful approach for linearly constrained optimisation issues. By optimising inside the constraint space, the interior point technique avoids the simplex method's search procedure across its boundary. Large-scale linear programming problems are frequently solved using the interior point method, particularly in high-dimensional environments where it displays superior efficiency. One can express the iterative phases of the interior point approach by the following equation:

$$x_{k+1} = x_k - \alpha_k \left(H_k\right)^{-1} g_k \tag{10}$$

where g_k is the objective function's gradient and H_k is the Hessian matrix of the current point. Particularly in cases of multiple restrictions, the interior point technique has the benefit in efficiently solving complicated and high-dimensional optimisation problems.

Usually, a convergence criteria is used to find whether the optimisation method has converged. One often used criterion is that the process of optimising is said to have converged when the gradient's norm is less than some preset value ε:

$$\left\|\nabla f\left(x_k\right)\right\| \le \varepsilon \tag{11}$$

where the threshold ε is rather modest. One can ensure that a near-optimal solution to the optimisation problem has been discovered by iterating until the gradient change is quite minimal.

By use of the Lagrange multiplier approach, the simplex method, and the internal point method, linearly limited numerical methods offer efficient tools for addressing

optimisation problems with linear constraints. Particularly in the domains of engineering design and neural network optimisation, these approaches have been extensively applied in pragmatic uses.

2.2 Recurrent neural network

Best defined by its capacity to transport information between time steps via cyclic connections, RNN is a neural network design for handling sequence data (Xiangxue et al., 2019). This is thus extensively applied in tasks including time series prediction and natural language processing since RNN can capture temporal connections in sequences.

Standard RNN updates the hidden layer state h_t recursively at every time step. Particularly, the hidden layer state at the current moment is computed using the hidden layer state h_{t-1} at the past moment concurrently with the current input x_t:

$$h_t = \tanh\left(W_h x_t + U_h h_{t-1} + b_h\right) \tag{12}$$

where W_h and U_h are the weight matrices of the network; b_h is the bias term; $\tanh(\cdot)$ is the activation function, hence generating nonlinearity and guaranteeing that the model can learn complicated patterns. This update lets the network create fresh hidden layer states depending on the inputs and recall past data.

Usually computed from the hidden layer state h_t at the current instant in RNN, the output y_t follows this formula:

$$y_t = W_y h_t + b_y \tag{13}$$

Long sequences cause ordinary RNN to suffer from gradient vanishing or gradient explosion, though. This is thus difficult for the network to learn long-term dependencies since during backpropagation the gradient becomes either very tiny or very large over several chained law passes (Zucchet et al., 2023).

Researchers have suggested enhanced RNN designs-that is, LSTM networks and gated recurrent unit (GRU) networks-to address this challenge (Khan et al., 2022). LSTM introduces several gates to manage the storage and information updating, therefore reducing the gradient vanishing issue. LSTM has a state update formula like this:

$$i_t = \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right) \tag{14}$$

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{15}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right) \tag{16}$$

$$h_t = o_t \odot \tanh\left(c_t\right) \tag{17}$$

where c_t is the cell state; i_t is the input gate; f_t is the forgetting gate; o_t is the output gate. These gates assist the network to keep significant information over time by regulating the input, forgetting and output of knowledge, therefore ignoring pointless stuff and helping to regulate the network.

Just two gates make GRU a condensed form of LSTM. Its state updating formula follows:

$$z_t = \sigma\left(W_z x_t + U_z h_{t-1} + b_z\right) \tag{18}$$

$$r_t = \sigma\left(W_r x_t + U_r h_{t-1} + b_r\right) \tag{19}$$

$$\tilde{h}_t = \tanh\left(W_h x_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right) \tag{20}$$

$$h_t = \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{21}$$

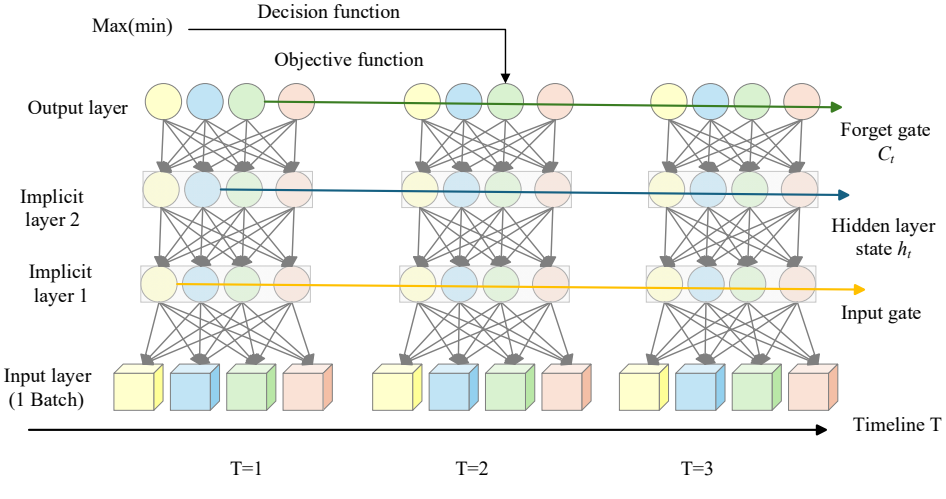
Though the performance is almost that of LSTM, GRU is more efficient in computation and appropriate in cases when computational resources are restricted (Fu and Wang, 2022). These two enhanced RNN designs help neural networks to avoid gradient issues and capture interdependence in extended sequences.

Especially in application domains like natural language processing, speech recognition and time series prediction, RNN, LSTM, and GRU excel in many sequence modelling tasks and have grown to be crucial tools for processing sequence data in deep learning.

3 Recurrent neural network optimisation model based on linearly constrained numerical methods

Aiming to improve the performance and computational efficiency of the model when dealing with complex time-series data by essentially merging time-series modelling with constrained optimisation, in this work we present the LSTM-LP optimiser model, which combines LSTM and LP optimisation approaches. The model is not only able to efficiently capture long-term dependencies in the time-series data during the training process but also able to optimise when faced with computational and resource constraints, so ensuring efficient and stable model training, see Figure 2 by introducing a linear constraint numerical method.

Figure 2 Structure of LSTM-LP optimiser model (see online version for colours)



The model comprises of five main modules: the LSTM modelling module for extracting the dependency information of the time-series data, the LP optimisation module for

optimising the parameter updates of the LSTM by linear constraints, the training module for combining the LSTM and LP for co-training, and the output module for generating the final prediction results and post-processing. By means of the synergy among these five modules, the model is able to efficiently manage the limited resources while preserving the accuracy and so attaining the aim of optimal training.

3.1 Input module

Data preparation and processing for LSTM-LP optimiser falls to this module. Usually mean normalised during the time series data preparation, the data is so that every feature has a mean of 0 and a standard deviation of 1. The formulae are as follows:

$$\hat{x}_t = \frac{x_t - \mu_x}{\sigma_x} \tag{22}$$

where $\hat{x}_t$ is the normalised data; x_t is the original input data; μ_x and σ_x are respectively the mean and standard deviation of the dataset.

The input module also must slide windows on the time-series data to divide the continuous time-series data into training samples fit for LSTM input (Nizam et al., 2022). One sets a window length w, and the sliding window can be produced using the following equation:

$$X_{t:w} = \{x_t, x_{t+1}, \ldots, x_{t+w-1}\} \tag{23}$$

where $X_{t:w}$ represents the window data spanning length w beginning from time step t. By use of this sliding window technique, the model may forecast the current time step's output depending on historical time step input.

3.2 LSTM modelling module

Extensive timing information from the timing data is extracted in this module to generate accurate expected outputs for the next optimisation step. By means of its special gating structure, which addresses gradient vanishing in conventional RNNs, LSTM is able to efficiently capture long-term dependencies (Zucchet and Orvieto, 2024). The LSTM modelling module not only concentrates on the relationships between time steps but also takes into account the limitations on the outputs to be altered at each instant to achieve more precise optimisation in order to combine LSTM with linearly limited numerical approaches.

LSTM's computational mechanism consists on a sequence of gating operations. First, computed as the forgetting gate controls how much of the past memories should be thrown away in the present:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{24}$$

The LSTM then updates the memory cells depending on input gate equation: control of the forgetting gate.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{25}$$

In the LSTM-LP optimiser model, the output of the LSTM module must be merged with the LP optimiser module to guarantee that the model satisfies particular resource limitations during training. Particularly, this tuning process is accomplished by the following equation:

$$h_t = LP\left(o_t \cdot \tanh\left(C_t\right), A \cdot h_t \le b\right) \tag{26}$$

To guarantee that the training process of the model does not surpass the resource constraint, $LP(\cdot)$ indicates the constraint adjustment of the current hidden layer state by the linear programming module, $A \cdot h_t \le b$ is the linear constraint connected with the resource constraint, and A and b are the coefficient matrix and the constant vector of the linear constraint, respectively.

3.3 LP optimisation module

This module ensures that the LSTM model not only learns the patterns of the temporal data effectively during the training process but also follows certain computational and resource constraints by means of linear programming methods, so combining the outputs of the LSTM model with predefined resource constraints and optimising the model. This module's major objective is to solve the linear programming problem so as to satisfy the specified constraints and increase the efficiency and stability of the training process, so adjusting the hidden layer states of the LSTM model.

Combining the hidden layer states of the LSTM model with the resource constraints produces an optimisation issue initially in the LP optimisation module. Usually linear, the problem consists in the hidden layer state h_t and a set of constraints. LP optimisation specifically aims to satisfy the linear constraints and maximise or reduce the hidden layer state h_t in respect to an objective function (Samanipour and Poonawala, 2023). One may represent the optimisation process as follows:

$$\text{maximise/minimise } c^T h_t, \quad \text{subject to } A \cdot h_t \le b \tag{27}$$

where h_t is the hidden layer state output by the LSTM module; c is the coefficient vector of the objective function; A is the coefficient matrix of the linear constraints; b is the constant vector of the constraints.

The LP optimisation module dynamically changes the hidden layer state h_t at every moment t and guarantees that the model generates valid outputs considering the constraints, therefore helping to better manage the linear constraints in the optimisation process. One can engage in this adjustment process by adding a fresh constraint formulation:

$$A \cdot h_t + d \le b \tag{28}$$

where d is a minor adjustment vector connected with the state adjustment of the hidden layer, therefore reflecting the possible changes during model optimisation. This new restriction guarantees that the model may satisfy resource restrictions by means of effective computations and state changes.

This method helps the LP optimisation module to efficiently control computational resources during the optimisation process and raise the general stability and efficiency of the model.

3.4 Training module

Handling the flow of input data, loss computation, model parameter optimisation, and resource constraint management is the responsibility of the training module of the LSTM-LP optimiser model. By means of backpropagation and linear programming, the training procedure aims to raise the performance and computational efficiency of the model on challenging time-series data.

First getting input from the LSTM modelling module, the training module delivers prediction results by updating the hidden layer state using LSTM. The loss function is computed depending on the discrepancy between the actual value and the projected result. Apart from the conventional prediction error, the loss function incorporates the constraint loss produced during the LP optimisation process, therefore aggregating into the total loss function. There is a formula for total loss as follows:

$$L_{total} = L_{prediction} + \lambda \cdot L_{LP} \tag{29}$$

where $L_{prediction}$ is the prediction error; L_{LP} is the constraint loss; λ is a hyperparameter meant to balance the prediction error and constraint loss.

The LP optimisation module changes its value at every cycle of model training based on the hidden layer states output from the LSTM such that the states satisfy the linear constraints. The following equation captures this process of adjustment:

$$h_t = P \cdot \tanh\left(C_t\right) + Q \tag{30}$$

where $\tanh(C_t)$ is the memory cell following the activation function; P and Q are the weight matrix of the linear constraints; h_t is the hidden layer state of the LSTM. Thus, the training module guarantees that the linear requirements are satisfied at every time step in addition to making sure the LSTM produces data in line.

3.5 Output module

First getting the hidden layer state h_t at the present instant from the LSTM module, the output module processes it to produce the final model output. The hidden layer state is further changed following linear programming optimisation throughout processing such that the result satisfies the established requirements. Specifically, the output module must carry the last computation to derive the prediction result y_t depending on the current hidden layer state, so it may be stated as:

$$y_t = W_{out} \cdot h_t + b_{out} \tag{31}$$

where W_{out} is the output layer's weight matrix; b_{out} is the bias term. The raw model output produced by this method is subsequently handled in more detail by other stages.

Together with LP optimisation, the output module changes the output value y_t to satisfy the resource restrictions therefore guaranteeing that the model output fulfils the established requirements (Cao et al., 2020). One achieves this change process by means of the following equation:

$$y_t = LP\left(W_{out} \cdot h_t,\, A \cdot y_t \le b\right) \tag{32}$$

where y_t is the optimised output; A is the matrix of coefficients of the linear constraints; b is the constant vector of constraints. By means of LP optimisation, the output module guarantees not only conformity with the computational and resource limitations but also with the data patterns.

In the end, the output module uses the produced prediction results y_t as the last output of the model, which finds use in several contexts like time series forecasting, decision support, etc. The design of the output module guarantees, however, that the model preserves excellent computational efficiency and accuracy even under limited resources.

4 Experimental results and analyses

4.1 Datasets

Two genuine and extensively used datasets were used for this work in order to assess the LSTM-LP optimiser model performance.

The first dataset is the Individual Household Power Consumption Dataset from the UCI Machine Learning Repository, which records, minute-by-minute data covering information on the power consumption of appliances utilised, during a four-year period. Derived from the California Traffic Surveillance System, the second dataset is the Performance Measurement System (PeMS) Traffic Flow Dataset, which records information including traffic flow and speed by means of multiple locations and multiple motorways recorded at five-minute intervals.

Table 1 shows specifics about the dataset.

Table 1 Summary of datasets for experimentation

Dataset name	*Electricity consumption dataset*	*Traffic flow dataset*
Data source	UCI Machine Learning Repository	Performance Measurement System (PeMS)
Data content	Contains electricity consumption data, recorded every minute over four years	Traffic flow data from multiple locations, recorded every five minutes across multiple highways
Data type	Power consumption (in watts)	Traffic flow, vehicle speed, lane occupancy
Task type	Load forecasting, energy demand prediction	Traffic flow prediction, intelligent traffic system optimisation
Data size	207,525 instances	Millions of records, covering data from multiple cities and highways

4.2 Impact of iterative optimisation on performance in time series prediction

This experiment aims to assess, in particular how the LSTM-LP optimiser model performs predictions on two distinct time-series datasets, namely how the model performance varies with increasing iteration count. First the experiments will be carried out on the PeMS traffic flow dataset and then the electricity usage dataset. We first load the PeMS traffic flow dataset and the dataset on individual household electricity use. To create a format fit for model input, the data were preprocessed including cleaning, normalisation and sliding window processing. We then selected the LSTM-LP optimiser model and defined starting hyperparameters including learning rate, hidden layer size,

etc. The other hyperparameters were maintained constant throughout the tests so that one may concentrate on the impact of the iteration count.

Figure 3 Results of model performance variation on power consumption dataset (see online version for colours)

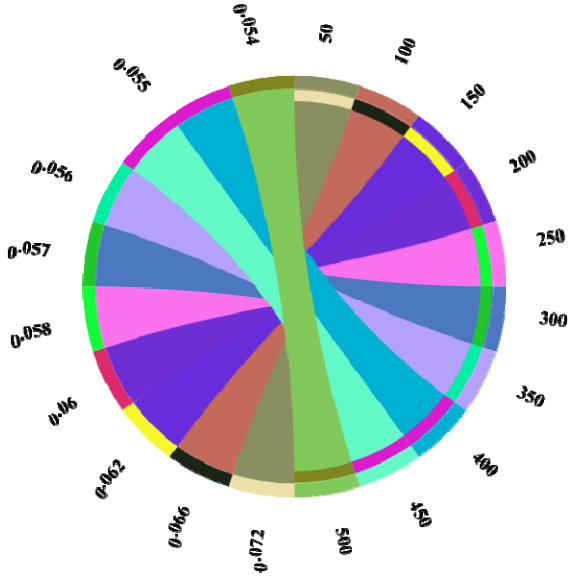
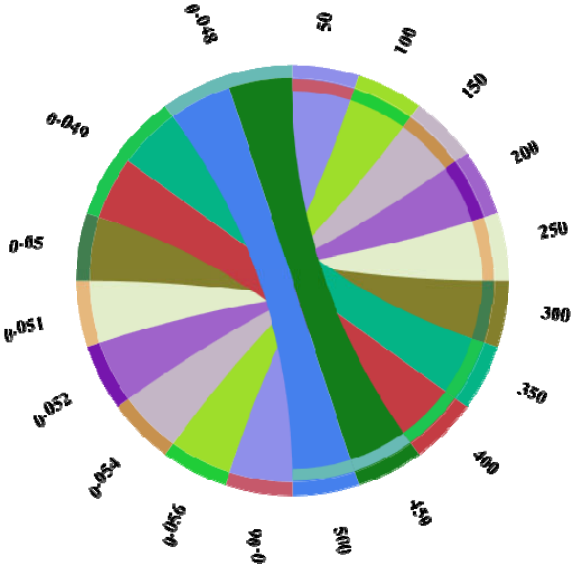


Figure 4 Results of model performance variation on traffic flow dataset (see online version for colours)



We trained the LSTM-LP optimiser model on each of the two datasets in the model training and assessment phase using a number of iterations beginning from 50 and rising

by 50 every time until 500. At every iteration, the models were assessed; a performance indicator was mean square error (MSE) (Yıldız and Karakuş, 2020). At every iteration, note the MSE value; then, examine the model performance trend. Determine the number of repetitions at which the MSE value starts to stabilise or the rate of decrease slows down noticeably, therefore suggesting a near convergence of the model.

Figure 3 and 4 display, for varying numbers of iterations, the LSTM-LP optimiser model's performance on both datasets.

The experimental results indicate that the MSE value of the model progressively declines with increasing iteration on the power consumption dataset, therefore showing improving model prediction performance. Particularly from 50 to 500 iterations, the MSE value drops from 0.072 to 0.054, indicating ongoing model performance improvement. From 350 iterations on, the MSE value changes very little and stays practically constant, implying that the model may have reached or is close to its optimal performance and that additional iteration has limited effect in the MSE. The MSE of the model similarly drops on the PeMS traffic flow dataset as number of iterations increases. The change in the MSE value levels off as the number of iterations hits 350, showing that the model performance is near to ideal, same like the power consumption dataset.

These results reveal that the LSTM-LP optimiser model shows good convergence on both the electricity consumption dataset and the PeMS traffic flow dataset; hence, the suitable number of iterations may balance the training duration and prediction performance of the model to prevent overfitting. Practically, the best number of iterations can be found depending on model performance on the validation set to provide good generalisation capacity without overconsuming computational resources. These results guide applications in related time-series data analysis projects and offer direction for additional optimisation of the LSTM-LP optimiser model.

4.3 Performance comparison of different timing prediction models

This experiment intends to evaluate on traffic flow and electricity consumption datasets the performance of several time-series prediction methods. Six time-series prediction models were chosen; LSTM, GRU, transformer, LSTM-LP optimiser, ARIMA and prophet using MSE, mean absolute error (MAE) and coefficient of determination (R^2) as evaluation metrics. Each model then had reasonable hyperparameters selected for it. Their MSE, MAE, and R^2 values were noted following separate training of these models on each dataset.

Figure 5 and 6 exhibit the experimental outcomes.

With its MSE of 0.05, MAE of 0.15, and R^2 of 0.92, which indicates a reduced error and better goodness of fit, the LSTM-LP optimiser model performs the best on the power consumption dataset from the experimental findings on all the evaluation indexes. Regarding the traffic flow dataset, the LSTM-LP optimiser model performs best in terms of R^2, with a R^2 of 0.89; the GRU model performs best in terms of MSE and MAE, with an MSE of 0.10 and MAE of 0.25. These findings show that the performances of several models vary significantly on various datasets, hence the model selection should be optimised in line with the features of the dataset.

Figure 5 Comparison of the performance of each time-series prediction model on the power consumption dataset (see online version for colours)

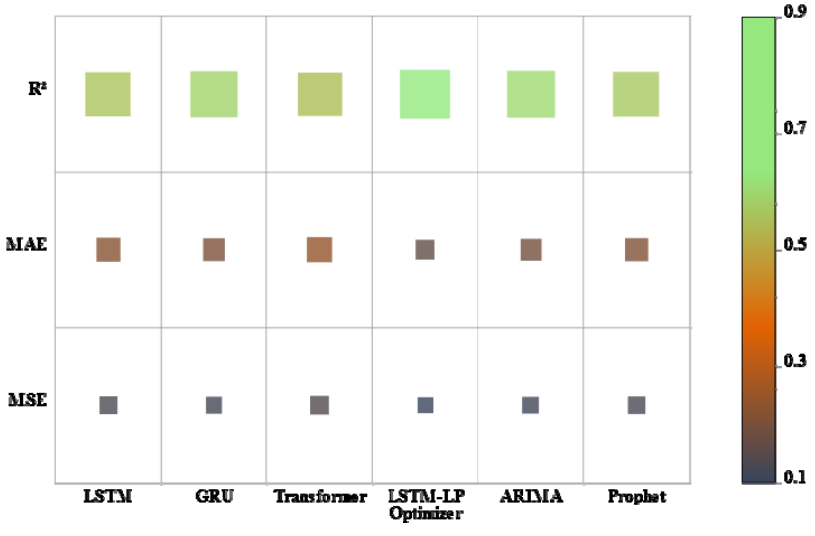
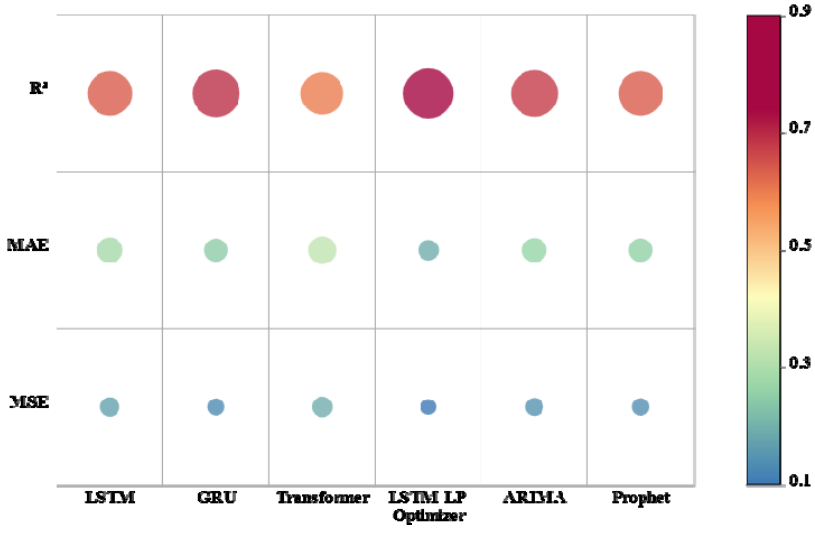


Figure 6 Comparison of the performance of each time-series prediction model on the traffic flow dataset (see online version for colours)



Overall, the LSTM-LP optimiser model has the best prediction performance on the electricity consumption dataset whereas the GRU model is more beneficial on the traffic flow dataset. These results offer direction for model selection in comparable time-series data analysis projects and a significant reference for the choice and optimisation of time-series prediction models. By changing the hyperparameters of the models or choosing models better fit for the features of the data, one can considerably raise the prediction performance of the models.

5 Conclusions

This work proposes to handle the challenge of handling complicated constraints in time series data analysis using an RNN optimisation model, LSTM-LP optimiser, grounded on the linear constrained numerical approach. On traffic flow and power consumption datasets, the model shows good performance by integrating the optimisation characteristics of LP with the time-series modelling capacity of LSTM. On the power consumption dataset, experimental results reveal that the LSTM-LP optimiser satisfies resource limitations and captures long-term dependencies, therefore attaining the lowest MSE, MAE and greatest R^2. Although GRU performs somewhat better on several measures on the traffic flow dataset, LSTM-LP optimiser still scores the best on R^2, therefore proving even more relevance under challenging limitations. Furthermore, the performance of the model gradually improves and stabilises after a given number of iterations as the number of iterations increases, so indicating that the model can efficiently converge under resource constraints and so offers a significant reference for the optimisation of computational resources in practical applications.

Still, the LSTM-LP optimiser model has certain constraints. First of all, the model is rather complex, particularly following the launch of the linear programming optimisation module; hence, the computational process gets more complicated and could result in longer training times, which could cause issues in large-scale datasets or situations requiring high real-time demand. Second, although the model performs well on both datasets, its generalisation capacity still has to be more confirmed. For other kinds of time-series data, such as medical or financial time series, the performance of the model could be impacted by the features of the data. Furthermore, the application of the model depends on domain knowledge and becomes more challenging since the environment of linear constraints has to be modified to particular issues.

Future investigations could go in the following lines:

1 Explore more efficient optimisation algorithms to reduce the computational complexity of the model: future studies should investigate more effective optimisation algorithms, such integrating approximation optimisation approaches (e.g., versions of stochastic gradient descent or approximate linear programming algorithms) to minimise processing overheads while guaranteeing optimisation accuracy.

2 Further research on the generalisation ability of the model: in order to fully evaluate the relevance of the model in many spheres, future studies can do tests on more varied kinds of information, including financial time series, medical, health, and meteorological data. Simultaneously, the model structure and parameter values are tuned for various data features, including changing the number of layers, the number of hidden units or the requirements of linear constraints of the LSTM, to better fit varied time series data.

3 Research on hyperparameter optimisation and automated tuning mechanisms for model: to streamline the use of the model, future studies could investigate hyperparameter optimisation strategies such automated hyperparameter tuning systems based on Bayesian optimisation, evolutionary algorithms, or lattice searches.

Declarations

All authors declare that they have no conflicts of interest.

References

Ahmed, S.F., Alam, M.S.B., Hassan, M., Rozbu, M.R., Ishtiak, T., Rafa, N., Mofijur, M., Ali, A.S. and Gandomi, A.H. (2023) 'Deep learning modelling techniques: current progress, applications, advantages, and challenges', *Artificial Intelligence Review*, Vol. 56, No. 11, pp.13521–13617.

Bhogade, V. and Nithya, B. (2024) 'Time series forecasting using transformer neural network', *International Journal of Computers and Applications*, Vol. 46, No. 10, pp.880–888.

Cao, X., Tang, G., Guo, D., Li, Y. and Zhang, W. (2020) 'Edge federation: towards an integrated service provisioning model', *IEEE/ACM Transactions on Networking*, Vol. 28, No. 3, pp.1116–1129.

Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L. and Muller, P-A. (2019) 'Deep learning for time series classification: a review', *Data Mining and Knowledge Discovery*, Vol. 33, No. 4, pp.917–963.

Fu, Y. and Wang, X. (2022) 'Traffic prediction-enabled energy-efficient dynamic computing resource allocation in CRAN based on deep learning', *IEEE Open Journal of the Communications Society*, Vol. 3, pp.159–175.

Han, Z., Zhao, J., Leung, H., Ma, K.F. and Wang, W. (2019) 'A review of deep learning models for time series prediction', *IEEE Sensors Journal*, Vol. 21, No. 6, pp.7833–7848.

Jin, L., Wei, L. and Li, S. (2022) 'Gradient-based differential neural-solution to time-dependent nonlinear optimization', *IEEE Transactions on Automatic Control*, Vol. 68, No. 1, pp.620–627.

Khan, I.A., Moustafa, N., Razzak, I., Tanveer, M., Pi, D., Pan, Y. and Ali, B.S. (2022) 'XSRU-IoMT: explainable simple recurrent units for threat detection in internet of medical things networks', *Future Generation Computer Systems*, Vol. 127, pp.181–193.

Landi, F., Baraldi, L., Cornia, M. and Cucchiara, R. (2021) 'Working memory connections for LSTM', *Neural Networks*, Vol. 144, pp.334–341.

Liu, X-W., Dai, Y-H., Huang, Y-K. and Sun, J. (2023) 'A novel augmented Lagrangian method of multipliers for optimization with general inequality constraints', *Mathematics of Computation*, Vol. 92, No. 341, pp.1301–1330.

Luo, X., Yuan, X., Zhu, S., Xu, Z., Meng, L. and Peng, J. (2019) 'A hybrid support vector regression framework for streamflow forecast', *Journal of Hydrology*, Vol. 568, pp.184–193.

Necoara, I., Nesterov, Y. and Glineur, F. (2019) 'Linear convergence of first order methods for non-strongly convex optimization', *Mathematical Programming*, Vol. 175, pp.69–107.

Nie, J., Wang, L., Ye, J.J. and Zhong, S. (2021) 'A Lagrange multiplier expression method for bilevel polynomial optimization', *SIAM Journal on Optimization*, Vol. 31, No. 3, pp.2368–2395.

Nizam, H., Zafar, S., Lv, Z., Wang, F. and Hu, X. (2022) 'Real-time deep anomaly detection framework for multivariate time-series data in industrial IoT', *IEEE Sensors Journal*, Vol. 22, No. 23, pp.22836–22849.

Prado, R.W., Santos, S.A. and Simões, L.E. (2023) 'On the fulfillment of the complementary approximate Karush-Kuhn-Tucker conditions and algorithmic applications', *Journal of Optimization Theory and Applications*, Vol. 197, No. 2, pp.705–736.

Samanipour, P. and Poonawala, H.A. (2023) 'Stability analysis and controller synthesis using single-hidden-layer ReLU neural networks', *IEEE Transactions on Automatic Control*, Vol. 69, No. 1, pp.202–213.

Wen, J., Yang, J., Jiang, B., Song, H. and Wang, H. (2020) 'Big data driven marine environment information forecasting: a time series prediction network', *IEEE Transactions on Fuzzy Systems*, Vol. 29, No. 1, pp.4–18.

Xiangxue, W., Lunhui, X. and Kaixun, C. (2019) 'Data-driven short-term forecasting for urban road network traffic based on data processing and LSTM-RNN', *Arabian Journal for Science and Engineering*, Vol. 44, pp.3043–3060.

Yıldız, S. and Karakuş, C.B. (2020) 'Estimation of irrigation water quality index with development of an optimum model: a case study', *Environment, Development and Sustainability*, Vol. 22, pp.4771–4786.

Zucchet, N. and Orvieto, A. (2024) 'Recurrent neural networks: vanishing and exploding gradients are not the end of the story', *Advances in Neural Information Processing Systems*, Vol. 37, pp.139402–139443.

Zucchet, N., Meier, R., Schug, S., Mujika, A. and Sacramento, J. (2023) 'Online learning of long-range dependencies', *Advances in Neural Information Processing Systems*, Vol. 36, pp.10477–10493.