Evolving Deep CNN-LSTMs for Inventory Time Series Prediction

Ning Xue[1], Isaac Triguero[1], Grazziela P. Figueredo [2] and Dario Landa-Silva [1]

[1]Automated Scheduling, Optimisation and Planning Research Group (ASAP)
[2]Advanced Data Analysis Centre
School of Computer Science, University of Nottingham, UK
Email: {ning.xue1, isaac.triguero, grazziela.figueredo, dario.landasilva}@nottingham.ac.uk

***Abstract*—Inventory forecasting is a key component of effective inventory management. In this work, we utilise hybrid deep learning models for inventory forecasting. According to the highly nonlinear and non-stationary characteristics of inventory data, the models employ Long Short-Term Memory (LSTM) to capture long temporal dependencies and Convolutional Neural Network (CNN) to learn the local trend features. However, designing optimal CNN-LSTM network architecture and tuning parameters can be challenging and would require consistent human supervision. To automate optimal architecture searching of CNN-LSTM, we implement three meta-heuristics: a Particle Swarm Optimisation (PSO) and two Differential Evolution (DE) variants. Computational experiments on real-world inventory forecasting problems are conducted to evaluate the performance of the applied meta-heuristics in terms of evolved network architectures for obtaining prediction accuracy. Moreover, the evolved CNN-LSTM models are also compared to Seasonal Auto-regressive Integrated Moving Average (SARIMA) models for inventory forecasting problems. The experimental results indicate that the evolved CNN-LSTM models are capable of dealing with complex nonlinear inventory forecasting problem.**

***Index Terms*—Particle Swarm Optimisation, Differential Evolution, Convolutional Neural Network, Long Short-Term Memory, Time Series Analysis, Inventory Prediction**

I. Introduction

Effective planning is important in production systems that aim at effective management and coordination of related activities and resources for an organisation. A common objective in such systems is to achieve optimal production planning and inventory management to meet (often variable) products demand over the planning horizon [1]. This paper tackles the inventory forecasting problem for highly perishable food with a very limited shelf life and with variable customers demand. Much research effort has been invested in developing accurate and robust inventory prediction models. This problem has been treated from various angles, such as time-series [2], pattern recognition [3], clustering [4]. Generally, from machine learning point of view, those models can be classified into parametric or non-parametric methods [5]. Parametric methods assume a certain statistical distribution on the data and primarily rely on statistical techniques such as auto-regressive moving average models, linear and nonlinear regressions [6]. These techniques try to detect a function between the past information and the predicted state. Non-parametric approaches do not assume that the data is following any distribution and adopt computational intelligent methods such as fuzzy systems, neural networks, and evolutionary computation.

In this study, we evaluate and compare the representative methods of parametric and non-parametric approaches using real-world data from a local food shop. We chose SARIMA model [7] as the parametric method while CNN-LSTM model as the non-parametric approach. Because both methods require careful selection of parameters, we implement a grid search method for the parameter tuning of the SARIMA model; with regards to the CNN-LSTM, its parameters contains various format and the quantity of parameters is far larger than that of the SARIMA model.

To automatically design optimal CNN-LSTM network architectures, we implemented and compared two similar meta-heuristics: Particle Swarm Optimisation (PSO) and Differential Evolution (DE). More specifically, a standard PSO and two DE variants: DE/best/1/bin and DE/best/1/exp (see Section IV). The reasons for adopting PSO and DE as optimisers for this problem are that: 1) they are somewhat similar algorithms and it is interesting to compare their performance; 2) they are simple to implement (both use simple fast to execute mathematical operators) and efficient in memory requirement; 3) and they are general parameter value optimisation methods which require no assumptions about the problem being optimised; 4) there are successful applications of PSO and DE for evolving neural network architectures (see Section II), but the applications of using them to evolve CNN-LSTM neural network architectures are somewhat limited. To the best of our knowledge, this is the first work using evolutionary algorithms for automatically evolving the architectures of CNN-LSTM for time series prediction.

The remainder of this paper is as follows. A literature review is given in Section II while the subject problem is described in Section III. The encoding scheme and proposed algorithm are illustrated in Section IV while the parameter tuning is presented in Section V. Computational experiments, comparisons, and discussion are presneted in Section VI. Finally, conclusions are drawn in Section VII.

This research was supported by Innovate UK and PXtech Limited.

II. Literature review

Time series, which is a sequence of data points in time order, is being generated in a wide spectrum of domains, such as daily fluctuation of stock markets, power consumption records, performance monitoring of data centres, forecasting of sales and inventory of retailer business, etc [8].

During the past few years, deep learning based methods have achieved tremendous progress in the domains of computer vision and pattern recognition and have attracted ever-increasing research interests since it can be deployed for big-data analytics, with applications encompassing: computer vision, pattern recognition, speech recognition, natural language processing, and advanced recommendation systems. Ever since the deep convolutional neural network (DCNN) model (AlexNET) proposed by Krizhevsky, et al. [9] for the ILSVRC2012 image classification challenge displayed tremendous success, other DNN models such as VGG [10], GoogLeNet [11], and ResNets [12] with excellent performance have also been put forward.

Moreover, DNNs have demonstrated superiority in numerous machine learning tasks in time series analysis from which CNNs and recurrent neural networks have been successfully applied in several applications. Recurrent neural networks are powerful in discovering the dependency in sequence data and particularly the LSTM works well on sequence data with long-term dependencies [8], [13], [14] due to the internal memory mechanism. Lipton et al. [15] presented a study to empirically evaluate the ability of LSTM to recognise patterns in multivariate time series of clinical measurements. A study that empirically evaluated RNN can be found at [13]. CNN is a type of DNN that is commonly applied to analysing image data. One of the key attributes of CNN is to conduct different processing layers that yield an effective representation of local salience of the signals. The deep architecture of CNN allows multiple layers of these processing units to be stacked, so that this deep learning model can characterise the salience of signals in different scales [16].

Hybrid neural networks combining the strength of several types of neural networks are receiving increasing interest in the domain of computer vision [8], such as its successful applications in image captioning [17], image classification [18], action recognition [19] and so on. However, the exploitation of hybrid architectures for time series analysis is somewhat limited. There are various ways to combine CNN and LSTM and the models can be trained separately and then combined together or vice versa [20]. The convolutions of the CNN can be used directly as part of reading input into the LSTM units themselves. This combination can be referred to as convolutional LSTM or ConvLSTM for short [21]. Du et al. [22] proposed a traffic flow forecasting framework based on hybridising LSTM and CNN, which focuses on the impact of local spatial features and long dependency features and spatial-temporal correlations in traffic flow data. Lin et al. [8] proposed an end-to-end hybrid neural network (CNN and LSTM) to learn local and global contextual features for predicting the trend of time series. CNN and LSTM are complementary in their modelling capabilities as CNN is good at reducing frequency variations while LSTM is good at temporal modelling [23].

In practice, given a new problem, researchers tend to add pre- and/or post-processing step(s) to improve solution quality without changing network architecture or to try to redesign the architectures and incorporate unique task-specific properties in order to obtain better results. However, designing network architecture and tuning parameters can be challenging. For example, specific problems like image classification would require expert knowledge, hence the need for consistent human supervision. Therefore, it would be crucial to design a framework that can automatically search for optimised network architecture.

This idea of applying evolutionary algorithms as a means for developing architectures of neural networks has been adopted. With the advance of high-performance computing such as GPU and other related cognitive computing clusters, the computational time can be significantly reduced. In fact, by adopting components such as convolution, pooling, batch normalisation etc., Real et al. [24] developed an intuitively mutation-capable evolutionary neural network that uniquely does not require any human participation once the mutation has commenced. To the best of our knowledge, Bin et al. [25] proposed the first work using PSO for automatically evolving the architectures of CNNs. Dragoi et al. [26] applied DE to determine the optimal neural network topology. Studies applied DE for designing of different variants of neural network can be found at [27] and [28].

III. Problem description

The problem concerned in this paper is part of a real-world production planning of highly perishable foods [29] and staff scheduling [30] in an environment with highly variable customers demand.

We predict the hourly sales of a food shop that produces and sells fresh prepared dishes, sandwiches and desserts. The shop usually opens at 7:00AM and closes at 10:00PM. For the purpose of using forecasting for inventory planning, it is crucial to predict at least one week sales ahead. Unreliable prediction could have negative impact on food preparation and inventory planning since conservative estimation of sales can result in low efficiency due to over prepared causing wastage while underestimation of sales may lead to customers reneging (i.e. customers leaving because the preferred item is out of stock). Therefore, the goal is to build a forecasting model that accurately predicts hourly sales of different items 1 week ahead.

The data for the time series prediction is obtained by transaction records held by the system used by the store. The transaction records contain information of items transition time-stamps and quantity. For each item, we grouped the data by sum of hourly sales. Subsequently, we pre-processed the data by removing outliers, system errors and retrospectively adjust quantity caused by unexpected transactions (e.g. returned product). As the store operates 16 hours per day and 7 days a week, the goal is to use the pre-processed historical data to predict the 112 hourly sales for each item with one week in advance. In total, we selected 10 case studies for testing our algorithm.

The obtained data are usually highly nonlinear and non-stationary, which are affected by different effects of store promotions, weather, trafc conditions and special events (e.g. holiday, football match). Therefore, the demand for items may change dramatically within the day or week as shown in Fig.1 and 2, respectively. In addition, the signal-to-noise ration is very low. In this problem, demand forecasting by CNN-LSTM model is motivated by the combination of CNN and LSTM neural networks, which considers the spatial-temporal dependency features of our data. CNN is used to capture the local trend features of demand (e.g. demand change from 9:00AM to 9:00PM in Fig.1) and LSTM is utilised to learn features of both short-term time variation and long-term dependency periodicity (e.g. the pattern of peak sales at noon time in Fig.2.

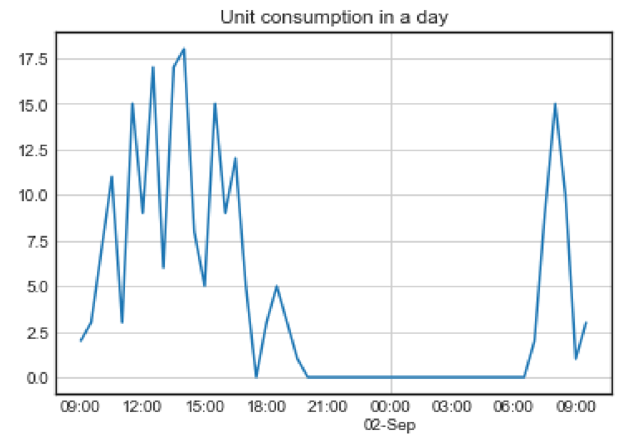


Fig. 1: Demand fluctuates considerably in a day.

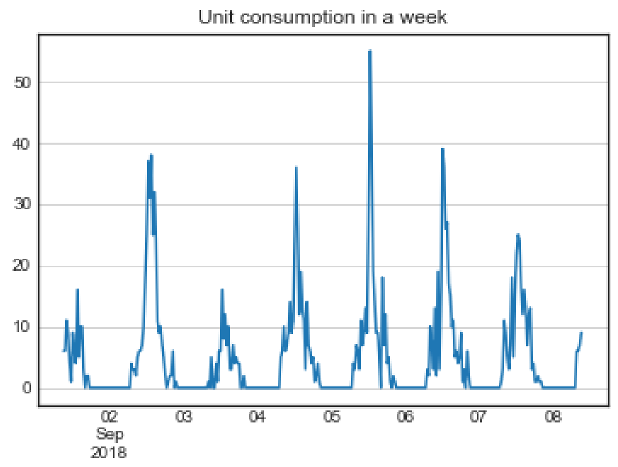


Fig. 2: Demand fluctuates considerably in a week.

IV. The proposed optimisation algorithms

Each time of searching the optimal neural network with certain parameters values (e.g. number and type of layers, number of neurons), we calculate the Mean Absolute Error (MAE), which is the solution fitness in the meta-heuristic (i.e. PSO or DE), and update parameters values. Here, we implemented 3 meta-heuristics: a regular PSO (see Section IV-B), a DE/best/1/bin and a DE/best/1/exp (see Section IV-C).

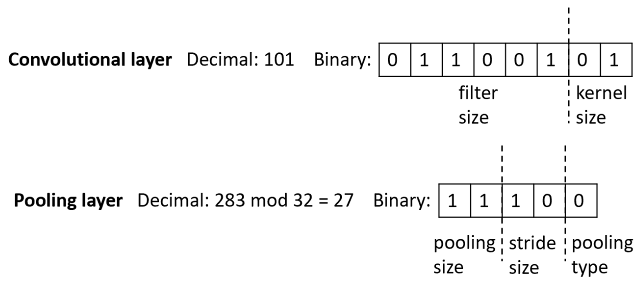
A. The encoding scheme

Each parameters values is encoded by a n-tuple $s = \{l_0, l_1, l_2, ...l_n\}$ representing layers in a neural network. l_i represents the ith layer and its value is an integer, more specifically, an integer expressed in decimal.

1) Convolutional layer: An integer value is not sufficient to encode the multiple parameters that required by the convolutional or pooling layer. To ensure a layer represents by only one integer value, the integer value representing convolutional or pooling layer is converted to a binary string. In this encoding scheme, we use an integer ranging from 1 to 256 to represent a convolutional layer as a convolutional layer require two key parameters - filter size and kernel size. Some preliminary experiments suggest that the filter size up to 64 (2^6) and kernel size up to 4 (2^2) are sufficient to obtain good results. In terms of integer and binary string conversion, integer range *[1,64]* and *[1,4]* can be converted by a 6 bits and 2 bits binary string respectively. Therefore, in total, we need a 8 bits binary string (range *[1,256]*) to represent the two parameters of a convolutional layer.

2) Pooling layer: Similar as the encoding of convolutional layer, the pooling layer has three key parameters - pooling size, stride size and pooling type. The range of pooling size and stride size are defined as *[1,4]* while the pooling type are $\{1, 2\}$ where 1 stands for max pooling and 2 stands for average pooling, therefore, the range of pooling layer can be represented by a 5 bits length binary string. Because *[1,256]* of range in the solution encoding has been taken by the convolutional layer, the pooling layer can take the placeholder of range *[257,288]* (size 32) next to *[1, 256]*. Since the placeholder size of the pooling layer (size 32) is only 1/8 of that of the convolutional layer (size 256), leading to much smaller possibility of pooling layer to be chosen comparing with the convolutional layer. To ensure each type of layer have comparable opportunity to be chosen by the meta-heuristics, an uniform distribution has been utilised by increasing the place holder size of pooling layer from 32 to 256, hence increase the range from *[257, 288]* to *[257, 512]*. Let the integer value before increasing range be b and after increasing the range be a, then $b = a \mod l_a$ where l_a is the placeholder size before increasing its value. An example of convolutional and pooling layer encoding scheme is given in Fig.3.

3) Dropout layer: Dropout layer has only 1 key parameter, is encoded without integer and binary string conversion. We use placeholder size 8 to represent dropout rate starting form 0.05 and with 0.05 increment (i.e. $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4\}$), similar with the



Convolutional layer Decimal: 101 Binary:
0 1 1 0 0 1 0 1
filter size
kernel size
Pooling layer Decimal: 283 mod 32 = 27 Binary:
1 1 1 0 0
pooling size
stride size
pooling type

Fig. 3: Encoding

pooling layer, the placeholder size of dropout layer is increased from 8 to 256 (i.e. range is increased to *[513, 768])* to increase the possibility to be chosen by the meta-heuristics. Again, a modulo operation is required to obtain its original value.

4) Empty, dense, and LSTM layer: In order to cope with the variable-length of CNN-LSTM architectures can be detected in the evolving process, a range of placeholder is used [25], representing an empty layer if the value of a layer fall within that range. To collaborate with the idea of all kinds of layers have similar opportunity to be chosen by the meta-heuristics, the range of placeholder size of empty layer is set to *[769, 1024]* (size 256) and the range of dense layer (i.e. fully connected neural network layer) is *[1025, 1278]* (size 256) which represents maximum 256 neurons. The placeholder range of LSTM layer is *[1025, 1278]* (size 256) which represents maximum 256 LSTM units.

The layer type and its range (based on best manual configurations) is summarised in table I below:

TABLE I: Layer, placeholder range and parameter list

Layer	Range	Parameters
Conv	[1, 256]	filter size: 1-64 kernel size: 1-4
Pooling	[257, 512]	pooling size: 1-4 stride size:1-4 pooling type:maxpooling, average pooling
Dropout	[513, 768]	[0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4]
Empty	[769, 1024]	place holder
Dense	[1025, 1280]	number of neurons
LSTM	[1281, 1536]	number of LSTM units

B. Particle Swarm Optimisation

PSO is a population-based meta-heuristic introduced by Eberhart and Kennedy [31] that has been successfully applied to many parameter value optimisation methods. PSO is initialised with a group of random particles (i.e. solutions, population members) and then the algorithm searches for optima by updating generations of particles. In each iteration of PSO, each particle is updated within its given bounds by following two "best" values. The first one is the best solution that the particle and its neighbours have achieved so far (local best). The other one is the best solution found so far by any particle in the whole population (global best). Based on these two best values, a particle updates its velocity and positions. More details of how PSO works can be found in [31]. There are many variants [32] of PSO, but in this paper we implemented the standard one [31].

C. Differential Evolution

DE optimises a problem by maintaining a population of candidate solutions and creating new candidate solutions by combining existing ones according to its simple formulae, and then keeping whichever candidate solution has the best score or fitness on the optimisation problem at hand. Usually, DE generates new parameter vectors by adding the weighted difference between two other vectors to a third vector in the population. If the resulting vector yields a lower fitness value than a predetermined member in the population, the newly generated vector replaces the predetermined vector that was compared in the next generation; otherwise, the old one is retained [33]. This basic principle has been extended to many variants of DE.

A DE algorithm can be marked as DE/x/y/z, where x denotes how the differential mutation base is chosen, y denotes the number of vector differences added to the base vector and z indicates the crossover method. Binomial and exponential crossover operators are two commonly used crossover variants, among which the former has been more often implemented [34]. For example an existing vector can be perturbed by mixing the parameters of the randomly selected old vector with those of the perturbed one before comparing the objective function value, such DE is notated as DE/rand/1/*, with binomial crossover, such mutant is then notated as DE/rand/1/bin. Similarly, if a perturbed vector is mixed with the best vector in the population using exponential crossover, such mutant is notated as scheme DE/best/1/exp. Please refer to [34], [35] and [33] for more detailed explanation about DE variants.

This paper evaluated both DE/best/1/bin and DE/best/1/exp for our problem. The motivation is two-fold: Firstly, before utilising DE/best/1/bin, we did some preliminary experiments on DE/rand/1/bin [33] which performs worse than DE/best/1/bin. This give us a hint to focus more on intensification (exploitation) instead of diversification (exploration) for the searching scheme. Secondly, We believe that high quality parameter vectors may share some common structures which could be evolved more efficiently through the exponential crossover in the evolution process. As a matter of fact, many CNN architectures follow the same general design principles of successively applying convolutional layers to the input, then periodically down-sampling the spatial dimensions while increasing the number of feature maps.

V. Parameter tuning

The parameters of optimisation algorithms, which are presented below, need to be tuned for good performance.

A. Particle swarm optimisation parameters

A standard implementation of the PSO algorithm requires 4 parameters:

- Swarm size (n): Number of particles in the population.

- Inertia (ω): Inertia weight in a particle's movement.
- Personal best attraction (phi_p): Weight for particle's pull towards its own best solution achieved so far.
- Neighbour best attraction (phi_g): Weight for pull towards the global best solution.

B. DE/best/1/bin parameters

The implementation of the DE/best/1/bin requires 3 parameters:

- Population size (n): Number of members in the population.
- Crossover rate (CR): controls the probability of crossover.
- Differential rate (F): controls the amplication of the differential variation.

C. DE/best/1/exp parameters

Similarly, the implementation of the DE/best/1/exp requires 3 parameters:

- Population size (n): Number of members in the population.
- Crossover rate (CR): controls the probability of crossover.
- Differential rate (F): controls the amplication of the differential variation.

The Irace package [36] was applied to help parameter tuning. In order to obtain parameters in reasonable computation time, small (n = 5) population size, only 3 month's training data and 1 month's test data of instance I1 were used for the parameter exploring experiments. We utilised online learning strategy (batch size=1) and epoch=100 when training neural networks. The length of parameter vector (i.e. number of layers) is limited to 12. The maximum number of allowed iterations (generations) is set to 100.

The parameters, their range considered in the tuning (based on some preliminary runs) and the best values found are given in Table II, III and IV. For Irace, the maximum number of experiments is set to 500 and other parameters are set to default values.

TABLE II: Parameter Tuning for the PSO Using Irace

Parameters	Type	Range	Best
ω	R	[-2, 2]	-0.572
phi_p	C	(-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2)	-0.5
phi_g	C	(-2, -1.5, -1, -0.5, 0, 0.5, 1, 1.5, 2)	-1

C: Categorical
R: Real

TABLE III: Parameter Tuning for the DE/best/1/bin Using Irace

Parameters	Type	Range	Best
CR	R	[0, 1]	0.654
F	R	[-2, 2]	0.836

R: Real

TABLE IV: Parameter Tuning for the DE/best/1/exp Using Irace

Parameters	Type	Range	Best
CR	R	[0, 1]	0.352
F	R	[-2, 2]	0.781

R: Real

VI. COMPUTATIONAL EXPERIMENTS

A. Performance baseline

A baseline in forecast performance provides a point of comparison. We choose SARIMA as a point of reference for the proposed algorithms. SARIMA is an approach for modelling univariate time series data that may contain trend and seasonal components. SARIMA is one of variants of Auto-regressive Integrated Moving Average (ARIMA) model which is the most widely used forecasting methods for univariate time series data forecasting.

Time series models come in three kinds: Moving Average (MA) models, Auto-regressive (AR) models and Auto-regressive Moving Average (ARMA) models. The data studied in this paper is built upon on ARIMA due to the non-stationary and stationary property of the data obtained. As its name interpreted, ARIMA(p,d,q) model is a combination of three terms, an auto-regressive term (p), a moving-average term (d) and an integrating term (q). Moreover, as the data contain repeating cycle, SARIMA can explicitly supports time series data with a seasonal component by adding additional seasonal terms in the ARIMA. The seasonal terms is used to specify the seasonal autoregressive order (P), Seasonal difference order (D), seasonal moving average order (Q), and the number of time steps for a single seasonal period (m).

Together, an SARIMA model can be formalised as SARIMA(p,d,q)(P,D,Q)m. Due to SARIMA is a type of regression models which are very sensitive to some design choices such as outliers removal (we applied Tukey's method [37]), how much previous data to use (we use 3 months and 1 month traing and test data split) and model parameters (i.e. (p,d,q)(P,D,Q)). Here, we set m to 16 for a daily seasonal cycle as an store operates 16 hours in a day. Then a grid search searching for the lowest Akaike Information Criteria (AIC) values is adopted to configure the 6 parameters (p,d,q)(P,D,Q). The AIC [38] is a widely used measure of a statistical model by quantifying the goodness of fit and the parsimony of the model. According to some preliminary experiments, each value of p, d, q, P, D, Q are set within range [0,3] for the grid search. The obtained results is given in table V.

TABLE V: SARIMA solution

Instance	SARIMA parameters	MAE
I1	(1, 0, 0)x(1, 0, 1, 16)	7.30
I2	(1, 0, 0)x(1, 0, 1, 16)	9.82
I3	(1, 1, 1)x(1, 1, 1, 16)	6.38
I4	(1, 1, 1)x(1, 1, 1, 16)	6.11
I5	(1, 1, 1)x(1, 1, 1, 16)	6.07
I6	(1, 1, 1)x(1, 1, 1, 16)	4.88
I7	(1, 1, 1)x(1, 1, 1, 16)	4.70
I8	(1, 0, 0)x(1, 0, 1, 16)	43.14
I9	(1, 0, 0)x(1, 0, 1, 16)	25.12
I10	(1, 0, 1)x(1, 0, 1, 16)	18.88

B. Performance of proposed algorithms

In order to analyse the performance of the proposed solution method, we test the algorithm on 10 problem instances using the best-suited parameter values given in section V. Population size n increased to 30, also the length of training and test data are increased to 12 and 2 months respectively. In order to obtain statistically sound results, all experiments are conducted with 10 independent runs (mean values were recorded) over all 10 problem instances. For each instance, all 10 runs start with the same initial solution created at random.

The algorithms was implemented in Keras [39] and run on a 8 cores PC with Intel i7 2.99GHZ processor and 16G RAM. Due to the very large amount of computational time required, we set the maximum number of allowed iterations (generations) to 100 for all instances. Each run of experiment on each instance takes around 30 to 40 minutes' computational time. Table VI presents the obtained MAE values. The best results are highlighted in bold. The average MAE show an increasing order as follows: DE/best/1/exp $<$ DE/best/1/bin $<$ PSO $<$ SARIMA. Generally, these experimental results suggest that DE/best/1/exp is be able to find better solution than others. The SARIMA performs the worst because the very low signal-to-noise ratio is in the data, especially for instances I8, I9 and I10.

It can be seen that DE/best/1/exp outperformed DE/best/1/bin at 9 instances except I10. The mean MAE obtained by DE/best/1/exp ($\mu_{DE/best/1/exp}$) is close but less than that of DE/best/1/bin ($\mu_{DE/best/1/bin}$). To test the significant of difference, we formulate the following hypotheses: $H_0 : \mu_{DE/best/1/exp} \geq \mu_{DE/best/1/bin}, H_a : \mu_{DE/best/1/exp} < \mu_{DE/best/1/bin}$ and performed a one-tail two samples t-test. P-value of this test is 0.005 ($<\alpha$=0.01), therefore we reject H_0 and a conclusion can be drawn that the DE/best/1/exp obtained better result compared with DE.

Compared with the results obtained by the PSO and the DE/best/1/bin, the DE/best/1/exp is superior due to exponential crossover is more effective when linkages exist between the neighbouring decision variables [34] [40]. Evidence can be found at Fig.4, 5 and 6 which show PSO has no improvement since the beginning of iteration process, but DE/best/1/exp can consistently improve. This is confirmed with our assumption of high quality parameter vectors may share some common structures which could be evolved more efficiently through the exponential crossover in the evolution process. The DE method performs better than PSO for this problem. We believe the reason is that DE/best/1/* scheme focus more on intensification instead of diversification when searching. Although PSO can be exploitative as well, but the qualified parameters might be difficult to be found for the regular PSO in the parameter tuning process in section V. In the end, the evolved neural network architecture by DE/best/1/exp for instances I1, I4 and I7 are given in Fig.7, 8 and 9 respectively.

TABLE VI: Experiments results

Instance	SARIMA	DE/best/1/bin	PSO	DE/best/1/exp
I1	7.30	5.28	5.30	**4.43**
I2	9.82	9.41	9.45	**8.44**
I3	6.38	4.63	4.55	**4.32**
I4	6.11	5.32	5.22	**4.98**
I5	6.07	4.57	4.62	**4.40**
I6	4.88	3.10	3.15	**2.81**
I7	4.70	3.25	3.48	**3.20**
I8	43.14	36.57	36.33	**35.66**
I9	25.12	19.76	19.70	**19.54**
I10	18.88	**14.55**	14.82	14.69
AVG	13.24	10.64	10.66	**10.25**

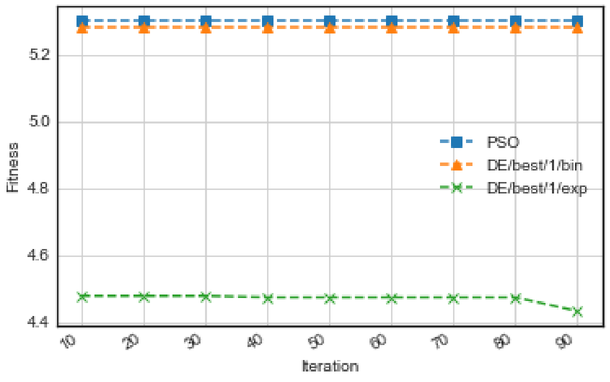


Fig. 4: Convergence rate of solving instance I1

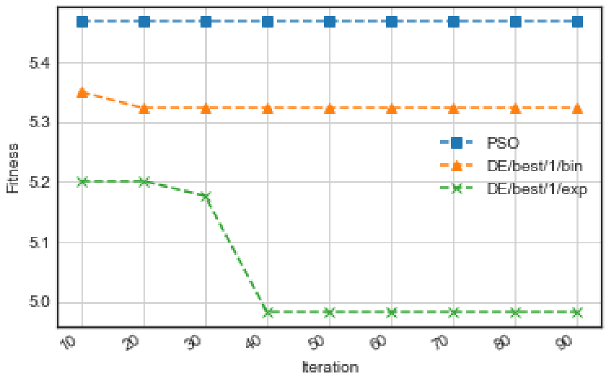


Fig. 5: Convergence rate of solving instance I4

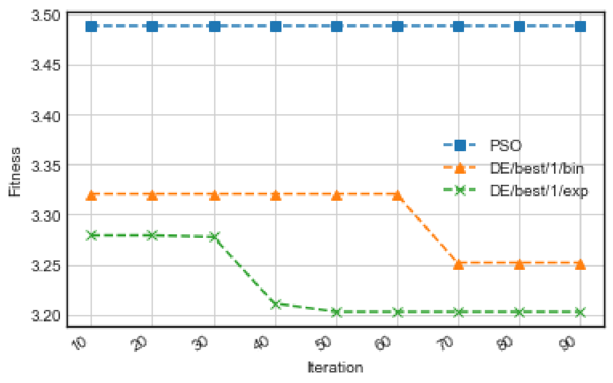


Fig. 6: Convergence rate of solving instance I7

VII. Conclusions

We have presented meta-heuristic approaches that succeeds in automatically evolving CNN-LSTM architectures for time series forecasting problems using real-world data from a local food shop. The goal of the proposed meta-heuristics is to identify new network architectures to improve forecasting accuracy. Three meta-heuristics, PSO, DE/best/1/bin, and DE/best/1/exp were evaluated. For a fair comparison, all the meta-heuristics were tuned and tested in the same experimental conditions. The results obtained from the inventory forecasting problem using time series data indicate that the proposed evolutionary approaches outperformed the SARIMA's prediction accuracy (baseline solution). Additionally, DE/best/1/exp outperformed the DE/best/1/bin and PSO in terms of evolved network architecture for obtaining prediction accuracy. We believe the present findings have the potential implications for efficiently detecting the architectures of deep neural networks.

Further work will be undertaken to improve the current method to obtain better results. Firstly, external dataset will be included for multivariate time series prediction, rather than just focusing on existed historical data set held by the store. Secondly, we will use high performance computing which allows much more generations of the evolutionary algorithms to be evolved and wider parameter range of layers to be explored.

References

[1] R. Ramezanian, D. Rahmani, and F. Barzinpour, "An aggregate production planning model for two phase production systems: Solving with genetic algorithm and tabu search," *Expert Systems with Applications*, vol. 39, no. 1, pp. 1256–1263, 2012.

[2] S. Nahmias and Y. Cheng, *Production and operations analysis*. McGraw-hill New York, 2009, vol. 6.

[3] H. R. Fogler, "A pattern recognition model for forecasting," *Management science*, vol. 20, no. 8, pp. 1178–1189, 1974.

[4] P. K. Bala, "Improving inventory performance with clustering based demand forecasts," *Journal of Modelling in Management*, vol. 7, no. 1, pp. 23–37, 2012.

[5] M. J. Anderson, "A new method for non-parametric multivariate analysis of variance," *Austral ecology*, vol. 26, no. 1, pp. 32–46, 2001.

[6] W. Shen and L. Wynter, "Real-time road traffic fusion and prediction with gps and fixed-sensor data," in *Information Fusion (FUSION), 2012 15th International Conference on*. IEEE, 2012, pp. 1468–1475.

[7] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[8] T. Lin, T. Guo, and K. Aberer, "Hybrid neural networks for learning the trend in time series," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2273–2279.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, "Learning to diagnose with lstm recurrent neural networks," *arXiv preprint arXiv:1511.03677*, 2015.

[16] J. Yang, M. N. Nguyen, P. P. San, X. Li, and S. Krishnaswamy, "Deep convolutional neural networks on multichannel time series for human activity recognition." in *Ijcai*, vol. 15, 2015, pp. 3995–4001.

[17] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2625–2634.

[18] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2285–2294.

[19] N. Ballas, L. Yao, C. Pal, and A. Courville, "Delving deeper into convolutional networks for learning video representations," *arXiv preprint arXiv:1511.06432*, 2015.

[20] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *Fifteenth Annual Conference of the International Speech Communication Association*, 2014.

[21] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[22] S. Du, T. Li, X. Gong, Y. Yang, and S. J. Horng, "Traffic flow forecasting based on hybrid deep learning framework," in *2017 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, Nov 2017, pp. 1–6.

[23] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak, "Convolutional, long short-term memory, fully connected deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4580–4584.

[24] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, "Large-scale evolution of image classifiers," *arXiv preprint arXiv:1703.01041*, 2017.

[25] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," *arXiv preprint arXiv:1803.06492*, 2018.

[26] E.-N. Dragoi, S. Curteanu, A.-I. Galaction, and D. Cascaval, "Optimization methodology based on neural networks and self-adaptive differential evolution algorithm applied to an aerobic fermentation process," *Applied Soft Computing*, vol. 13, no. 1, pp. 222–238, 2013.

[27] H. Dhahri, A. M. Alimi, and A. Abraham, "Hierarchical multi-dimensional differential evolution for the design of beta basis function neural network," *neurocomputing*, vol. 97, pp. 131–140, 2012.

[28] S.-K. Oh, W.-D. Kim, and W. Pedrycz, "Design of optimized cascade fuzzy controller based on differential evolution: Simulation studies and practical insights," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 3, pp. 520–532, 2012.

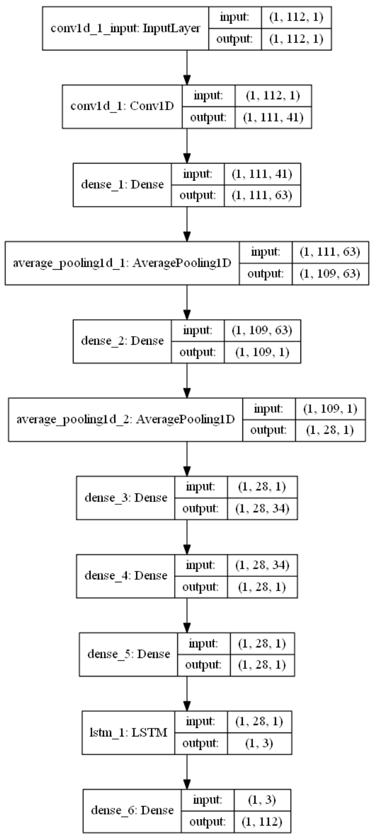

conv1d_1_input: InputLayer | input: (1, 112, 1) | output: (1, 112, 1)
conv1d_1: Conv1D | input: (1, 112, 1) | output: (1, 111, 41)
dense_1: Dense | input: (1, 111, 41) | output: (1, 111, 63)
average_pooling1d_1: AveragePooling1D | input: (1, 111, 63) | output: (1, 109, 63)
dense_2: Dense | input: (1, 109, 63) | output: (1, 109, 1)
average_pooling1d_2: AveragePooling1D | input: (1, 109, 1) | output: (1, 28, 1)
dense_3: Dense | input: (1, 28, 1) | output: (1, 28, 34)
dense_4: Dense | input: (1, 28, 34) | output: (1, 28, 1)
dense_5: Dense | input: (1, 28, 1) | output: (1, 28, 1)
lstm_1: LSTM | input: (1, 28, 1) | output: (1, 3)
dense_6: Dense | input: (1, 3) | output: (1, 112)

Fig. 7: The evolved architecture for instance I1

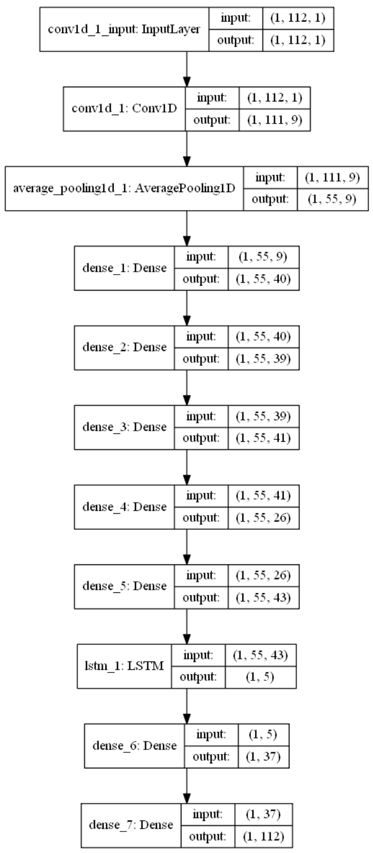

conv1d_1_input: InputLayer | input: (1, 112, 1) | output: (1, 112, 1)
conv1d_1: Conv1D | input: (1, 112, 1) | output: (1, 111, 9)
average_pooling1d_1: AveragePooling1D | input: (1, 111, 9) | output: (1, 55, 9)
dense_1: Dense | input: (1, 55, 9) | output: (1, 55, 40)
dense_2: Dense | input: (1, 55, 40) | output: (1, 55, 39)
dense_3: Dense | input: (1, 55, 39) | output: (1, 55, 41)
dense_4: Dense | input: (1, 55, 41) | output: (1, 55, 26)
dense_5: Dense | input: (1, 55, 26) | output: (1, 55, 43)
lstm_1: LSTM | input: (1, 55, 43) | output: (1, 5)
dense_6: Dense | input: (1, 5) | output: (1, 37)
dense_7: Dense | input: (1, 37) | output: (1, 112)

Fig. 8: The evolved architecture for instance I4

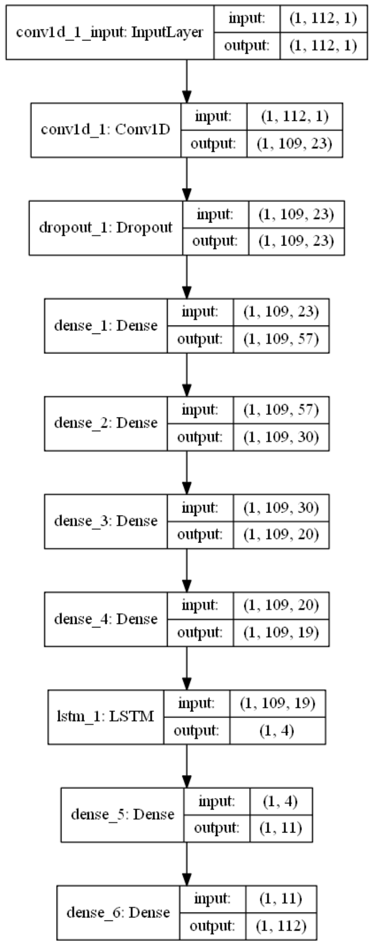

conv1d_1_input: InputLayer | input: (1, 112, 1) | output: (1, 112, 1)
conv1d_1: Conv1D | input: (1, 112, 1) | output: (1, 109, 23)
dropout_1: Dropout | input: (1, 109, 23) | output: (1, 109, 23)
dense_1: Dense | input: (1, 109, 23) | output: (1, 109, 57)
dense_2: Dense | input: (1, 109, 57) | output: (1, 109, 30)
dense_3: Dense | input: (1, 109, 30) | output: (1, 109, 20)
dense_4: Dense | input: (1, 109, 20) | output: (1, 109, 19)
lstm_1: LSTM | input: (1, 109, 19) | output: (1, 4)
dense_5: Dense | input: (1, 4) | output: (1, 11)
dense_6: Dense | input: (1, 11) | output: (1, 112)

Fig. 9: The evolved architecture for instance I7

[29] N. Xue, D. Landa-Silva, G. P. Figueredo, and I. Triguero, "A simulation-based optimisation approach for inventory management of highly perishable food," in *8th International Conference on Operations Research and Enterprise Systems (ICORES 2019)*. Springer, in press.

[30] N. Xue, D. Landa-Silva, I. Triguero, and G. P. Figueredo, "A genetic algorithm with composite chromosome for shift assignment of part-time employees," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[31] R. Kennedy, "J. and eberhart, particle swarm optimization," in *Proceedings of IEEE International Conference on Neural Networks IV, pages*, vol. 1000, 1995.

[32] D. P. Rini, S. M. Shamsuddin, and S. S. Yuhaniz, "Particle swarm optimization: technique, system and challenges," *International journal of computer applications*, vol. 14, no. 1, pp. 19–26, 2011.

[33] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[34] S. Das, S. S. Mullick, and P. N. Suganthan, "Recent advances in differential evolution–an updated survey," *Swarm and Evolutionary Computation*, vol. 27, pp. 1–30, 2016.

[35] R. Storn, "On the usage of differential evolution for function optimization," in *Biennial conference of the North American fuzzy information processing society (NAFIPS)*, vol. 519. IEEE Berkeley, 1996.

[36] M. López-Ibánez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles , Tech. Rep., 2011.

[37] J. N. Miller, "Tutorial reviewoutliers in experimental data and their treatment," *Analyst*, vol. 118, no. 5, pp. 455–461, 1993.

[38] H. Akaike *et al.*, "Likelihood of a model and information criteria," *Journal of econometrics*, vol. 16, no. 1, pp. 3–14, 1981.

[39] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[40] R. Tanabe and A. Fukunaga, "Reevaluating exponential crossover in differential evolution," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2014, pp. 201–210.