Applied Intelligence (2022) 52:808–834
https://doi.org/10.1007/s10489-021-02472-6

Estate price prediction system based on temporal and spatial features and lightweight deep learning model

Sheng-Min Chiu[1] · Yi-Chung Chen[2] · Chiang Lee[1]

Accepted: 22 April 2021 / Published online: 12 May 2021
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2021

Abstract

The development of estate price prediction systems is one of the issues that researchers are paying the most attention to. A good estate price prediction system can shorten the time it takes buyers to consider estates and invigorate the estate market. Generally speaking, an estate price prediction system considers the temporal and spatial features of the estate. In addition, the estate price prediction system can also be launched online for users to make instant online queries with, which means that it needs short run time. However, most existing studies only considered either temporal or spatial features and could not consider both, thereby resulting in questionable prediction accuracy. Although deep learning may increase prediction accuracy, it does not meet the short run time requirement. We therefore presented three ideas in this study to overcome these issues: (1) designing a novel spatiotemporal data structure, the Space-Time Influencing Figure (STIF), to quantify the influence of changes in the facilities surrounding each estate on estate price, (2) designing a novel CNN-LSTM model to go with the STIFs for estate price prediction, and (3) designing a new framework to extract the most important features to estate price for certain types of estates and combining these features with a shallow RNN for modeling. The computation cost of this model is far lower than that of a CNN-LSTM model, making it suitable for practical application. Finally, we used actual estate data from Taiwan to verify that the proposed approach can effectively and swiftly predict estate prices.

Keywords Estate price prediction · Deep learning models · Lightweight model · Spatial temporal database

1 Introduction

In recent years, estate price prediction has become a research focus for researchers. Estate prices are high and constantly fluctuate, which means that buyers must make detailed assessments before making transactions to ensure that they do not lose money in the future. An accurate estate price prediction system would cut down the pre-transactions assessment costs for buyers, accelerate estate transactions, and invigorate the estate market. Here, we give an example of an estate price facilities). Figures 1, for example, includes Estate A, obnoxious facility Landfill B, and public facility Market C. The figure also records the features of A, B, and C that influence the price of Estate A, such as the estate type, building age, floor area, and original price of A and the construction time, removal time, size, noise, and richness of B and C. Suppose a system user wants to find the future price of A; the system will show that following the construction of C, the price of Estate A will rise from $1 M to $1.5 M. Next, following the removal of B, the price of Estate A will rise from $1.5 M to $2.3 M. This example shows that estate price is influenced by temporal and spatial features in the city. Thus, an estate price prediction system must consider both types of features at the same time. Furthermore, such systems are generally placed online for users to make instant queries with. The results must therefore be given to the user instantly. In other words, in addition to prediction accuracy being a necessary consideration in the implementation of such systems, it must also be considered whether the prediction computation time is short enough to meet the user's expectations regarding the response time of the query system.

Although estate prices are influenced by the temporal and spatial features in a city at the same time, past studies generally used only one type of feature for prediction. If only temporal features are considered, time series forecasting is

✉ Yi-Chung Chen
chenyich@yuntech.edu.tw; mitsukoshi901@gmail.com

[1] National Cheng Kung University, Tainan 701, Taiwan

[2] National Yunlin University of Science and Technology, Yunlin 64002, Taiwan

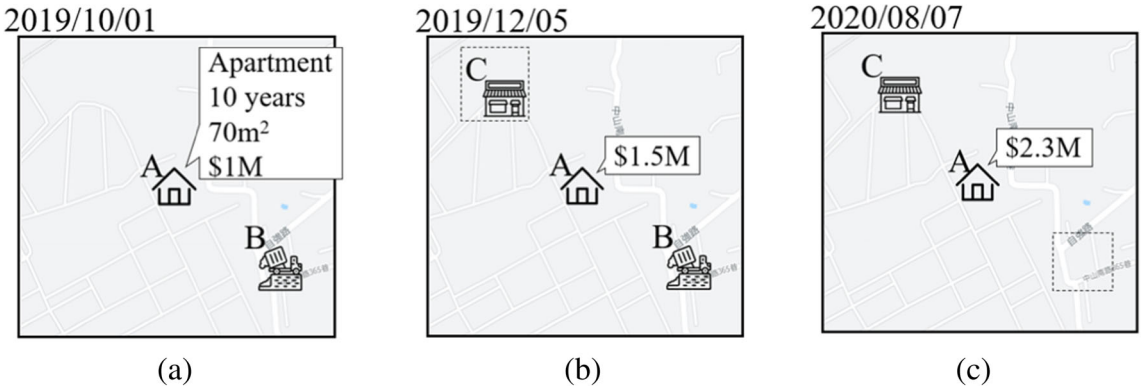

Fig. 1 Example of estate price prediction system: **a** original conditions; **b** conditions after construction of Market C on 2019/12/05; **c** conditions after removal of Landfill B on 2020/08/07

generally the most common approach. For instance, Peng [58] used the relationship between a stock price index and housing prices to predict future housing prices; Lim et al. [44] employed an autoregressive integrated moving average model and neural network models to predict the future condominium price indices of estates. However, such papers mostly used commercial or economic indices for prediction, so their predictions were limited to national fluctuations in estate prices; their methods could not forecast the prices of specific estates and were thus unsuitable for practical application. In using spatial features to predict estate prices, previous methods include using geographically weighted regression models [81] or compiling spatial features into matrix form and then making predictions using convolutional neural networks (CNNs) [6]. However, such methods do not consider temporal features and are thus unsuitable for many practical applications. In other words, existing estate price prediction methods do not consider temporal and spatial features simultaneously.

Beyond the field of real estate, spatio-temporal time series prediction (STTSP) has been applied to several other topics, such as the prediction of traffic flow [45, 68, 83], and PM2.5 levels [33, 43, 82, 85]. Previous approaches to STTSP can be divided into three types: statistical methods, machine-learning methods, and deep-learning methods (DLMs). Studies based on these methods all follow a similar procedure: they first analyze the format and characteristics of the spatio-temporal data of the target application and accordingly modify existing models or design new models. For example, the well-known statistical method Space-Time Autoregressive Integrated Moving Average Models [36] first converts the proximity relationship of different locations in the space into a weight matrix. Then the weight matrix is incorporated into the existing time series model ARIMA. Extended applications [12, 20, 52] of this method all first convert the spatial proximity relationship in the spatiotemporal dataset into various weight matrices, and then integrate these weight matrices into the ARIMA operation. In machine-learning methods, a suitable feature value selection method is designed according to the format and characteristics of the target spatio-temporal dataset, and then selected feature values are imported into various machine learning models [30, 68, 85]. Compared to statistical approaches, this method is favored by many scholars because it avoids complicated mathematical calculations and uses a data-driven perspective for modeling. In recent years, the concept of DLM has become popular, and research has focused on DLMs that accept both time and space data formats. For example, Li et al. [43] and Huang and Kuo [33] proposed applying a CNN with a long short-term memory (LSTM) to PM2.5 prediction. CNN analyzes the correlation between different grids in the space, and imports this correlation into the subsequent LSTM dedicated to time series issues for modeling. Gao et al. [23], Zhang et al. [90], and Cui et al. [15] proposed new ideas of storing temporal and spatial relationships of different road sections in graph data structures and then applied graph convolutional networks to the prediction of traffic flow. All above approaches have proven effective for different applications. However, their success is based on a match between the approach and the characteristics and format of the spatio-temporal datasets. In other words, drawing on previous studies for a new STTSP application requires careful evaluation of the new dataset. For traffic and PM2.5 applications, time data are continuous; that is, there is a data

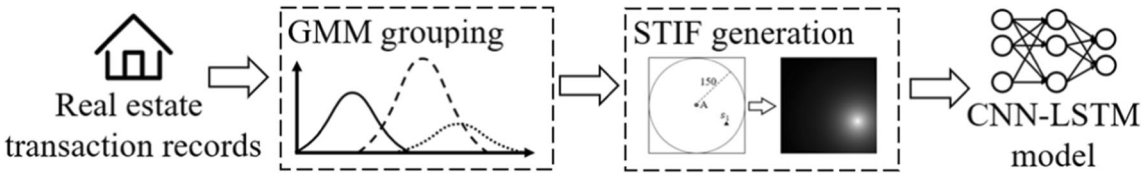

Fig. 2 Procedure of basic framework for estate price prediction

entry for every few seconds, hours, or days. For real estate applications however, transactions are infrequent and time intervals vary. Traffic, and PM2.5 applications also involve clearly-defined relationships between locations (roads or weather observation stations for example) or griddable datasets (PM2.5 levels in the environment for example). For real estate data, because the location of houses for sale is not fixed, and houses in the same district may differ greatly, the dataset is difficult to grid and the relationship between entities (i.e., houses for sale) is difficult to define. In other words, previous approaches to STTSP are not applicable to the topic of this thesis.

This study therefore proposes a Basic Framework for Estate Price Prediction (BEPP). The BEPP is based on three major concepts, as shown in Fig. 2: (1) using a Gaussian mixture model (GMM) [17, 18, 21] to divide all of the estates into groups based on their features (2) designing a novel spatiotemporal data structure, the Space-Time Influencing Figure (STIF), to quantify the influence of changes in the facilities surrounding each estate on estate price, and (3) designing a novel CNN-LSTM model to go with the STIFs and perform estate price prediction on each group produced by the GMM. The first concept was proposed because the price fluctuations of different estates in a city may vary widely. Suppose a brand new luxury house and a 30-year-old one-story house are next to each other. If a new park were built in front of them, the changes in their prices would differ. In this case, we must divide estates with different price fluctuation trends into different groups and use different models to predict the prices in each group. In this way, we can increase prediction accuracy.

Note that this work used GMM to cluster estates, because estate datasets feature two important characteristics that GMM is uniquely equipped to handle. First, there are many factors that influence estate prices, and they should all be taken into consideration for price prediction. Second, estate prices do not display clear changes when a feature changes slightly (i.e., the relationships between prices and features are fuzzy). Say that four houses, A, B, C, and D, are situated in the same community and are in similar condition. They are 7 years old, 8 years old, 10 years old, and 100 years old, respectively, so in theory, their prices should be A > B > C > D. In practice, the prices of A, B, and C are indeed higher than that of D. However, the prices of A, B, and C may be A = B=C or C > B = A, as A, B, and C differ only slightly in terms of age and thus their prices are more influenced by unquantifiable factors such as neighbors, surrounding views, and estate maintenance. The concept underlying GMM is the use of multiple Gaussian functions to individually describe the data distributions of each feature followed by a combination of Gaussian functions for grouping. Therefore, GMM is capable of solving fuzzy relationships. Furthermore, it is not affected by the number of features. The *k*-means [22, 88, 91] and DBSCAN [19, 39] algorithms on the other hand calculate the absolute distances between all the features of any two items of data. This makes grouping datasets with high-dimensional features difficult because a significant difference between the values of any of the many features in the two items of data could severely affect the distance between the two items. Clearly, these two methods cannot cope with fuzzy relationships between feature values and estate price. Decision trees [65, 72] and random forests [8, 29, 78] are well-suited to high-dimensional data, but their algorithms use precise value conditions, such as greater than and less than, for grouping. Consequently, they are unsuitable for estate datasets, where the relationships between feature values and estate price are fuzzy. Finally, although fuzzy c-means clustering [7, 34] considers the fuzzy relationships between values, since it still groups data points based on the distances between them, it is not as effective for datasets with high dimensionality.

The second concept of this paper was designing a novel spatiotemporal data structure, STIF, to quantify the influence of the features surrounding each estate. This data structure generates a set of STIFs for each Estate A, each set containing $m \times n$ STIFs with m denoting the number of features being considered and n indicating the number of additions, changes, or removals made to facilities around Estate A. Each STIF is a two-dimensional matrix with A in the center, representing the influence of the surrounding features on A. Next, each grid in a STIF reflects the influence of a feature on the estate price. If the influence is positive (i.e., increases the estate price), then the value in the matrix is greater. Figure 3 presents a simple

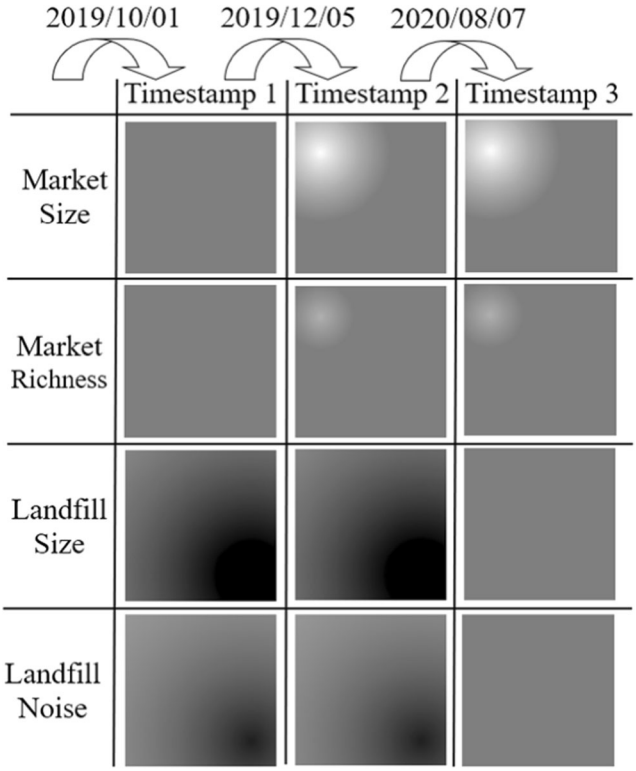

Fig. 3 STIF corresponding to Fig. 1

example, with Fig. 3 being the results of the data in Fig. 1 stored using a STIF. Figure 1 considers four features: the size and richness of the market and the size and noise of the landfill. Except for the original conditions, the features of the surrounding facilities have changed twice. Thus, three time points must be considered. Figure 3 thus produces 12 STIFs (=4 features×3 time points) of A. Furthermore, markets generally have a positive impact on estate prices, so when the market is constructed in Fig. 3, the STIF shows a white area at this location. The white thins out with distance, meaning that the impact of the market features on the prices of the surrounding estates weakens with distance. Furthermore, the range of the white area for each market feature differs, indicating that the influence of different features on the prices of the surrounding estates differs as well. As for the landfill, its influence on the prices of surrounding estates is generally negative, so it presented using black in the STIFs. The other details are identical to those of the market, so we will not repeat them here. Using the STIFs, we expect to be able to quantify the influence of each facility feature in a city on the prices of surrounding estates and store these influence values with time changes as the unit, which will facilitate estate price prediction.

The third concept in this paper is designing a novel CNN-LSTM model to go with the STIFs for estate price prediction. This input of this model consists of all of the STIFs of all of the estates assigned to the same group by the GMM, whereas the outputs are the estate prices for each time point a STIF corresponds to. The CNN in the first half of the CNN-LSTM mainly analyzes the relationships between the target estate and the features of the facilities surrounding it. The LSTM in the second half analyzes the influence of changes in the facilities on the price of an estate at different time points. With this network, we expect to be able to consider the influence of temporal and spatial features on estate prices at the same time.

The concepts above will help us predict estate prices while taking both temporal and spatial features into account. Nevertheless, these concepts are not suitable for practical situations because prediction systems must allow users to make instant queries and cannot take too long to make predictions [10, 53, 61].

CNN-LSTM models are a type of DLM, so their computations are surely time-consuming and cannot be used for instant queries. Of course, the service provider could upgrade their equipment to reduce the run time of the CNN-LSTM model and shorten the response time of the system, but that would mean substantial equipment costs for the service provider. Suppose a real estate agency wanted to set up an online estate price prediction system for general users. If they wanted the system to handle 100 queries simultaneously, with a wait time of less than 3 s (the theories of user experience design tell us that the response time of the website should be within 10 s, and preferably within 3 s [5, 31, 56]). There exist several possible setup scenarios. In Table 1, we present the model, hardware, response time, single hardware unit process, expense, and predicted accuracy of each scenario. These details are based on prices obtained in 2021. Scenario 1 was most commonly-used by businesses before the development of DLMs, for which only shallow neural networks (SNNs) were used for prediction. Operations using SNNs have low hardware requirements, fast computing speed, low monetary costs, and a short response time for each query. For this setup, even if 100 users submitted a query simultaneously, a single server could respond to all of the queries within (100 queries × 0.02 s/query) = 2 s with hardware setup costs of only USD $1000. The drawback of this setup scenario is that it is not as accurate as the other options. Scenarios 2 and 3 both use DLMs to perform predictions, so they offer higher accuracy than Scenario 1. However, they cost considerably more in terms of equipment. In Scenario 2, the server has a single GPU. Suppose each calculation using a DLM takes the server 2.9 s, and the real estate agency still wants each user to be able to obtain results within 3 s, the business would need 100 servers to perform parallel computations to meet this requirement. Consequently, the system cost would be USD $350,000, which is 350 times the cost of Scenario 1. In Scenario 3, the server has three GPUs. Because of this upgrade, it only takes the server 0.6 s to calculate a query using a DLM. Each server could then deal with the queries of five users within 3 s, but the business would still need 20 servers of the same caliber to deal with the queries of 100 users within 3 s. In this case, the system setup cost would be $17,000 × 20 servers = $340,000. Clearly, while using a DLM for prediction would greatly increase prediction accuracy, it would significantly increase the cost of the system.

Reducing the computational costs of DLMs has become a prominent area of interest. Although DLMs offer highly accurate modeling results, the temporal and monetary costs of their computation are also higher, making them impractical for many real-world applications. At present, attempts to reduce the computational cost of DLM can be divided into four categories: (1) pruning the structures within DLM architectures that do not have significant impact on the output to reduce the amount of computation, (2) quantizing the weights in the DLM to reduce memory usage, (3) extracting features useful for analysis from the DLM beforehand to train low-cost models, and (4) improving the hardware environment. Since the fourth approach focuses on hardware, it is beyond the scope of this study, so we will not discuss it here. Pruning structures within DLM architectures that do not have significant impact on the output is considered the most intuitive approach. It has been developed in several aspects. For instance, Han et al. [27] and Li et al. [42] respectively demonstrated that removing weights or complete filter layers can effectively increase the computation speed of DLMs. Wu et al. [77] developed a channel-pruning algorithm to eliminate

Table 1 Costs and precision of shallow neural networks (SNN), deep learning models (DLMs), and lightweight deep learning models (LDLMs) for real-world online query systems

Scenarios	Model	Server	Time cost of a single query	Price of a Server(USD)	Money cost of the scenario to run 100 queries in 3 s	Accuracy
1	SNN	0 GPU	0.02 s	$1000	$1000×1 server=$1000	70%
2	DLM	1GPU	2.9 s	$3500	$3500×100 server= $350,000	95%
3	DLM	3GPUs	0.6 s	$17,000	$17,000×20 server= $340,000	95%
4	LDLMs	1GPU	0.1 s	$3500	$3500×4 server=$14,000	93%

the parts of YOLO v4 and verified that the pruned YOLO v4 was faster and more accurate than other DLMs. Roy et al. [60] discovered that DLMs must be retrained after their frameworks are pruned, and therefore, they proposed a dynamic pruning-while-training procedure to avoid this issue. These studies all show the validity of the pruning approach, but some researchers have pointed out that not all hardware can support pruned DLMs. Furthermore, in some cases, pruning does not help to increase the computation speed. As a result, researchers have investigated the quantization of DLM weights to reduce the memory usage of the model to increase computation speed. For example, Wess et al. [76] proposed a novel quantization method to convert DLM weights into a hardware-friendly format. Huang et al. [32] and Kluska and Zięba [37] each developed algorithms to assess the quantization feasibility of each DLM level for quantization. Yang et al. [79] identified the optimal solution of scalar *k*-means clustering and the key ingredient of the DLM before performing DLM quantization. Comprehensively speaking, these studies could indeed overcome the issues encountered in the pruning approach. However, they still face a fundamental problem: their models are based on DLMs, which means they are unsuitable for applications that run on low-end equipment, such as wearable devices or factory equipment that operates on field-programmable gate arrays (FPGAs). Researchers therefore proposed the extraction of features useful for analysis from the DLM beforehand to train low-cost models. The theoretical foundation of such research is that during training, the first several layers of DLMs automatically identify features that can maximize the modeling precision of the last several layers of the DLMs. Thus, if we could effectively extract the features identified by the first several layers of the trained DLM, applying these features into any AI model should theoretically produce good modeling accuracy [10]. For instance, Sani et al. [61] and Mohammad et al. [53] first disassembled DLMs to obtain features conducive to activity recognition and then employed low-cost AI models for recognition. The results were recognition rates close to those of DLMs. Chen and Li [10] first trained and then disassembled an RBF-LSTM to extract feature values that are important to PM2.5 prediction; they then verified that inputting these features into any low-cost AI model would result in prediction accuracy close to that produced by modeling directly using DLMs. On the whole, we found that compared to models based on pruning and quantizing, those based on extracting useful features for training avoid complex memory configuration problems during online calculations. Moreover, low-cost models, rather than large DLMs, can be directly applied, which can meet the low-cost and fast-computation requirements of online query systems. Thus, this is the method we selected for development of the proposed online estate price prediction system. Further details on each method are presented in Section 2.4.

Below, we consider the example described in Table 1, in which a real estate agency wants to implement an online query system capable of handling the queries of 100 users simultaneously with a response time of less than 3 s. Suppose the agency implements a lightweight DLM based on the concepts of previous research to make estate price predictions (Scenario 4). Although the lightweight DLM reduces the prediction accuracy slightly (from 95% to 90% ~ 93%), it completes a calculation in 0.1 s using one server with a single GPU. In this case, the queries of 30 users are responded to within 3 s, so the real estate agency would only need four such servers to meet the original conditions. The hardware setup cost would be only $3500 × 4 servers = $14,000, which is approximately 1/25 of the costs of Scenarios 2 and 3. While these results are promising, existing lightweight DLMs only consider temporal data and not spatial data, so they cannot be directly applied to this study. Therefore, a new approach is needed to overcome this limitation.

We designed the novel Lightweight Framework for Estate Price Prediction (LEPP) to overcome the high computational cost issue of CNN-LSTM models. This framework is achieved by establishing two models. The first model is called the extraction model, which adds a lighten dropout layer to the previously described CNN-LSTM model. This layer is used to extract key features for estate price prediction from the trained extraction model. The second model is called the lightweight model, the framework of which is that of a shallow RNN model. Its inputs are the key features extracted by the extraction model. Note that in this model, the number of key features is far lower than the total number of features. Moreover, the model is a shallow RNN model, so its computational costs will certainly be lower than those of a CNN-LSTM model,

making it more suitable for practical applications. The features input into the lightweight model are also key features of estate price prediction, so the accuracy of estate price prediction will not be affected significantly. To verify the effectiveness of the proposed approach, we used real estate transaction records and urban industry data from Taichung, Taiwan to conduct simulations.

The remainder of this paper is arranged as follows. Chapter 2 introduces the related literature. Chapter 3 introduces the proposed basic framework for estate price prediction. Chapter 4 introduces the proposed lightweight framework for estate price prediction. Chapter 5 presents the experiment simulations, and Chapter 6 contains the conclusions and future plans.

2 Related literatures

This chapter introduces literature from four major fields: estate price prediction, spatio-temporal time series prediction, CNN-LSTM models, and the acceleration of DLMs.

2.1 Estate Price prediction

Existing studies on estate price prediction used either temporal or spatial features only; so far, no studies have considered temporal and spatial features at the same time. Those that used temporal features to predict estate prices mostly considered estate prices to be a type of time series and employed established time series-based methods for prediction. For instance, Peng [58] used the Hidden Markov Model and the autoregressive moving average to predict stock and housing prices in 2009. Lim et al. [44] used the autoregressive integrated moving average and an artificial neural network model to predict the condominium price index. Unfortunately, estate prices are generally affected by so many variables that these types of analyses are not reliable. Therefore, to date, the majority of researchers have relied on various financial indexes to perform quantitative analyses and make predictions. However, there has been considerable variation in the research methods that were used, and the predictions that these studies made rarely corresponded to actual estate prices.

Common methods using spatial features for estate price prediction include statistical methods and spatial analysis. In statistical methods, Bency et al. [5] adopted a similar approach using satellite imagery to predict prices in the estate market. However, these methods do not examine temporal changes in spatial data. In spatial analysis, Yiorkas and Dimopoulos [81] analyzed estate prices and made predictions using spatial information via the OLS method, with calibration based on geographically weighted regression.

2.2 Spatio-temporal time series prediction

The topics of Spatio-temporal time series prediction (STTSP) have recently attracted many scholars' attention because such topics like traffic flow prediction [45, 68, 83] and PM2.5 prediction [33, 43, 82, 85] are mostly related to human activities. If scholars can come up with methods to accurately predict these processes, the government, industry, and the public can all benefit from that. Compare to the common time series prediction in the past, STTSP differs most in that the method must consider the time and space characteristics of the data at the same time. And the characteristics and formats of spatial features are usually different from those of temporal features. Therefore, STTSP cannot directly use the time series prediction method, and must modify the original time series prediction method or redesign it. Existing approaches to STTSP can be roughly divided into three types: statistical methods, machine-learning methods, and DLMs. The most well-known statistical methods are the Spatial Vector Autoregressive Model (SpVAR) and the Space-Time Autoregressive Integrated Moving Average Model (STARIMA). Both integrate spatial conditions into existing time series forecasting models. SpVARs use Vector Autoregressive Models (VAR) [40, 41]. For example, Anselin [3] put the neighboring point data of target locations into VAR to verify its effectiveness for economic issues. Chudik and Pesaran [14] extended the practice of [3], additionally considering the static and dynamic factors of nearby location data. LeSage and Pan [41] and LeSage and Krivelyova [40] put the time series of several important locations together in Bayesian VAR. Lacombe and Michieka [38], Gupta and Das [24], and Das et al. [16] verified that this approach was also effective for the prediction of industrial output, real estate downturn, and real estate inflation. STARIMA [36] first converts the hierarchical ordering proximity relationship of different points in the space into an order weight matrix and then incorporates this matrix into the Autoregressive Integrated Moving Average model (ARIMA). Cheng et al. [12] and Min and Wynter [53] suggested that the original STARIMA only considers static conditions and does not work in practice. Therefore, they developed different weight matrix establishment methods to express dynamic spatial relationships and integrated these matrices into ARIMA. Recently, Duan et al. [20] developed a STARIMA method that considers peak and off-peak traffic conditions.

In machine-learning approaches, scholars regard time and space conditions as the goal of model learning or treat the two conditions as the input of the model to enable it to learn the characteristics of the spatio-temporal data. For example, Zhan et al. [85] established a spatial smoothing kernel for PM2.5 prediction as the loss function of the gradient boosting machine to improve prediction accuracy. In addition to time data,

Hengl et al. [30] listed the distance between the target location and the observation location as one of the input variables of the random forest algorithm. Yang et al. [82] developed the Gauss vector weight function to investigate the correlation of PM2.5 value changes among different small areas. The most relevant values are input to a support vector regression model. Tang et al. [68] extracted the time correlation and equivalent spatial distances of the flow data of different road sections for highway flow prediction, and input these factors into type-2 fuzzy c-means for classification. They used a neural network for traffic prediction on road sections in each category. Yue and Yeh [83] and Liu et al. [45] studied road flow prediction and proposed different spatio-temporal feature value extraction methods. Liu et al. [46] proposed the novel idea of first converting the traffic flow data into graphs, and then using a neural network for training.

In recent years, a large number of scholars have begun to discuss how to use DLMs for high-precision STTSP. Unlike machine-learning approaches trying to reach their goals by changing the input or the loss function of their models, DLMs build networks that consider the relationship between time and space. For example, on the topic of PM2.5 prediction, scholars [33, 43] have directly imported meteorological observation data from different locations into a CNN with LSTM. CNN analyzes the correlation between different grids in the space, finds the most effective factor for predicting PM2.5 levels, and imports it into the subsequent LSTM dedicated to time series issues for modeling. Next, in the prediction of traffic flow, many scholars try to organize the relationship between roads into Graph and use Graph Convolutional Networks to make predictions. For example, the methods proposed by Gao et al. [23], Zhang et al. [90], and Cui et al. [15] belong to this type of research and have achieved remarkable success. Looking at various researches using DLM concepts to solve STTSP, we find that this type of approach is different from traditional machine learning approaches. It allows all spatio-temporal data to be imported into the model intact and allows modeling to be done in the same model. Therefore, this kind of method can ensure that all the information contained in the spatio-temporal data will not be lost, and thus achieve a better modeling effect.

2.3 CNN-LSTM models

In the past, CNN-LSTM models were often used to process signals and images. For the former, signals were input into 1D CNNs for feature extraction. For instance, Zhang et al. [87] combined a 1D CNN and a CNN-LSTM model with signal time domain waveform for automatic modulation classification. Wang et al. [74] utilized a CNN-LSTM for speech recognition. Shahbazi and Aghajan [63] used a CNN-LSTM to process EEG signals.

In terms of image processing, Aravindkumar et al. [4] proposed a CNN-LSTM model for image captions. Zainudin et al. [84] performed histopathology cancer image classification using a CNN-LSTM model. Many researchers have also discussed the conversion of features with geographical relations into image format with the hope that CNNs can aid in extracting features from images, thereby achieving task objectives. For example, Sun et al. [67] combined weather data, MODIS Land Surface Temperature data, and MODIS Surface Reflectance data, converted them into histogram-based tensors, and constructed a CNN-LSTM model to predict historical soybean yield data. Moudhgalya et al. [55] proposed a CNN-LSTM model that uses environmental features and biological taxonomy to predict the hierarchical species that can most likely be observed at a given location.

2.4 Methods to accelerate DLM calculations

The high computational costs of DLMs greatly affect their usability in real-world applications. Researchers have proposed a number of methods to accelerate DLM calculations; these methods can be roughly divided into four categories: (1) DLM framework simplification, (2) quantization of DLM, (3) key feature extraction to reduce network size, and (4) DLM computation hardware improvement.

The first category eliminates a small portion of neuronal connections to reduce the size of the DLM. Early methods assessed and chose connections that were not important. For instance, Hassibi and Stork [28] employed second-order derivative information to determine whether connections between neurons were important and then pruned those that were not. However, the calculations of this assessment method are extremely complicated and not suitable for practical applications. Another approach involved pruning low weights from the DLM. For instance, Han et al. [25] used L1/L2-norm regularizations to eliminate weights. Scardapane et al. [62] and Wen et al. [75] regularized DNN parameters using Lasso penalties. These researchers all claimed that their approaches were effective in theory. However, they are still unsuitable for practical DLM applications because the weight matrices of the models will become sparse matrices once the DLM eliminates some of the weights in the fully connected layers. Currently existing software and hardware cannot accelerate the computation speeds of DLMs with sparse matrices [26, 57, 86], which means that these methods can only increase DLM computation speed to a limited extent. In view of this, another faction of researchers presented the idea of pruning whole layers of neurons rather than just some of the weights. For example, Li et al. [42] used an approach similar to that in [25] but focused on calculating the absolute weight sums of filter layers because the weight sum can represent the activation of the layer. Those with weaker activation have less impact on the DLM output and can thus be directly

eliminated. Molchanov et al. [54] proposed using the Taylor expansion to estimate the influence of filter layers on loss functions and determine whether said filter layers should be removed. Luo et al. [48] incorporated a greedy algorithm combined with a loss function to determine which filter layers should be kept. These methods claimed to be effective in pruning, but their calculation methods were also becoming increasingly complex, making them difficult to realize in practical applications and leaving room for improvement. To improve the efficiency of apple blossom detection, Wu et al. [77] proposed a novel approach to prune YOLO v4 frameworks. They first established a YOLO v4 under the CSPDarknet53 framework and then developed a channel-pruning algorithm to prune the trained YOLO v4. Through a case study, they verified that the pruned model was more accurate and more efficient than other DLMs commonly used in image processing, including Faster R-CNN, Tiny-YOLO v2, YOLO v3, SSD 300, and EfficientDet-D0. Roy et al. [60] consequently discovered that DLMs must be retrained after their frameworks are pruned, and they therefore proposed a dynamic pruning-while-training procedure to simplify the system.

The second approach to accelerating DLM calculations is quantization of the DLM. Compared to the first approach, this method reduces complex calculations without modifying or pruning the model framework. This makes it highly favored by researchers. The core concept is quantization of weights in the DLM to reduce memory use and increase calculation speed. For instance, Kluska and Zięba [37] proposed a novel approach for the quantization of each layer in a CNN and verified that this approach could accelerate the computation speed of the network without reducing its accuracy. Abdi and Fekri [1] used factorization to conduct the indirect quantization and compression of stochastic gradients and verified that their method was effective in increasing the computation speeds of DLMs. Using a dynamic programming algorithm, Yang et al. [79] identified the optimal solution of scalar K-means clustering to determine the key ingredient of the DLM for quantization. Wess et al. [76] combined two quantization methods (dynamic fixed point and power-of-two) with layer-wise precision scaling to directly convert DLM weights into a hardware-friendly format. Based on weight quantization, input quantization, and partial sum quantization, Huang et al. [32] developed a mixed precision quantization scheme to assess the quantization feasibility of each DLM level and then employed the concept of deep reinforcement learning to develop an automated process to search for the best quantization configuration in the large design space of the DLM. They demonstrated that their proposed method was well-suited to the specific hardware. Tung and Mori [71] combined the pruning method of the first approach with quantization, with good results.

As for reducing network size via key feature extraction, researchers have advocated that if suitable features could be found and input into conventional machine learning models, they should be able to achieve the same level of performance as DLMs. For instance, Sani et al. [61] disassembled DLMs to obtain features conducive to activity recognition and paired these features with a low-cost *k*NN approach for recognition. The resulting recognition rate was close to that of a DLM. Mohammad et al. [53] developed a set of mathematical methods to obtain multiple features useful for activity recognition from a trained DLM and applied the features to a conventional machine learning model for activity recognition, thereby deriving a low-cost solution with good recognition rates. In view of the difficulty of obtaining electroencephalography (EEG) data samples, Liu et al. [47] developed a semi-supervised learning model to train a CNN using existing EEG data and created an algorithm to obtain the deep features identified by the CNN. These deep features were then used to categorize EEG data. Their experiment results demonstrated that this approach can effectively extract important features from a small quantity of EEG data and improve the analytical abilities of models trained using a small quantity of data. Recently, Chen and Li [10] verified that these processing concepts are applicable to complex PM2.5 prediction as well. In their study, they used historical PM2.5 datasets to train a DLM called the RBF-LSTM model. Next, they developed a set of algorithms to swiftly obtain the important features identified by the RBF-LSTM model to function as predictors of PM2.5. Finally, they demonstrated that no matter which low-cost AI model these features are input to, they derive prediction accuracy close to that of direct modeling using a DLM. These approaches have currently become mainstream in relevant fields; nevertheless, we found that these feature extraction methods are mostly limited to time series analysis and are not directly applicable to this study. Thus, some new methods are needed to assist with estate price prediction problems.

In addition to modifying and pruning the model framework or decreasing the model inputs, many researchers have discussed a fourth method to accelerate DLM calculations: improving the hardware environment of model execution. Two directions exist for improvement. The first involves the development of a memory hardware framework dedicated to DLMs to accelerate the operating speed of DLMs on servers. The second attempts to accelerate the operations of DLMs on low-end embedded systems or mobile phones. Currently the most well-known achievement in the first category is the development of resistive random-access memory (ReRAM), which can effectively accelerate matrix-vector product calculations and is thus well-suited to DLMs that have massive amounts of matrix-vector product calculations. For instance, Song et al. [66] found that although ReRAM can accelerate

matrix-vector product calculations, it is not suitable for CNN training. For this reason, they designed PipeLayer and verified that their approach can effectively improve the computational performance of CNNs on ReRAM. Chen et al. [11] observed that generative adversarial networks operate inefficiently on ReRAM, so they upgraded ReRAM to ReGAN. Ankit et al. [2] indicated that the outer products of existing ReRAM are limited to processing low precision operations and developed a bit-slicing technique to overcome this issue. The second category of hardware improvement methods seeks to increase the computation speeds of DLMs on low-end hardware. Many real-world applications can only be operated on low-end embedded systems or mobile phones, and the DLM calculation speeds in such applications cannot be increased by modifying the model. Yao et al. [80], for instance, analyzed the nonlinear relation between DLM structure and execution time and improved the calculation speed of the DLM in embedded systems or mobile phones based on this relation. Marchesan et al. [50] accelerated DLM calculations while considering the memory space, storage space, and power consumption of embedded systems. More recently, Zhang et al. [89] and Tian et al. [69] respectively designed accelerators specifically for YOLO and 3D-CNN operating under FPGAs.

3 Basic framework for estate Price prediction

This chapter introduces the proposed basic framework for estate price prediction (BEPP), which includes four parts. Figure 4 describes the relationships and execution processes of these four parts, and Table 2 presents the pseudo code. The first part uses a GMM to divide estates in the same area with different price trends into different groups (line 1 of Table 2). The second and third parts use the grouping results to generate STIFs (input of CNN-LSTM model) for each estate and the estate price corresponding to each STIF (output of CNN-LSTM model) (lines 3–4 of Table 2). The fourth part establishes the CNN-LSTM model for the BEPP (line 5 of Table 2).

3.1 Gaussian mixture model for grouping

This section explains how to use a Gaussian mixture model (GMM) to divide the estates in a city into groups based on their price fluctuation trends. Note that our approach is similar to that in [59], but it is not the core of this paper. Below, we only introduce the focal points, and readers may refer to [59] for more details.

The GMM used in this study mainly uses estate transaction record datasets to group estates. Suppose the transaction record dataset of an estate is $\mathbf{F} = \{\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_{|F|}}\}$, including |F| items of data. Each transaction record $\mathbf{f_i}$ can be further expressed as $\mathbf{f_i} = (F_i.time, F_i.position, F_i.fattr_1, F_i.fattr_2, \ldots, F_i.fattr_\alpha)$, denoting three types of data features in the transaction datasets, including transaction time, position, and α general features (i.e., transaction price, transacted floor area, building age, etc.…). Note that we separated transaction time and position from the other features because grouping with these two features does not have much meaning, and we excluded them from our GMM discussion.

Grouping the datasets using the GMM and the general features comprises seven steps. We present pseudo code for these steps in Table 3: Steps 1, 2, 3, 5, and 7 respectively correspond to lines 2, 3, 4, 7, and 8 of the pseudo code.

- **Step 1:** Extract the information involving the *j*th general feature, $attr_j$, from each transaction record in F to form new dataset $\mathbf{F.attr_j} = (F_1.attr_j, F_2.attr_j, \ldots, F_{|f|}.attr_j)$. Note that at the very beginning of the algorithm, we set *j* as 1.
- **Step 2:** Use a probability density function to describe $\mathbf{F.attr_j}$:

$$G\left(F.attr_j; \Psi\right) = \sum_{k=1}^{g} w_k g_k\left(F.attr_j; \Psi_k\right), \tag{1}$$

where *G* denotes the number of Gaussian components in $attr_j$; w_k and g_k $(F.attr_j; \Psi_k)$ are the weight and probability density

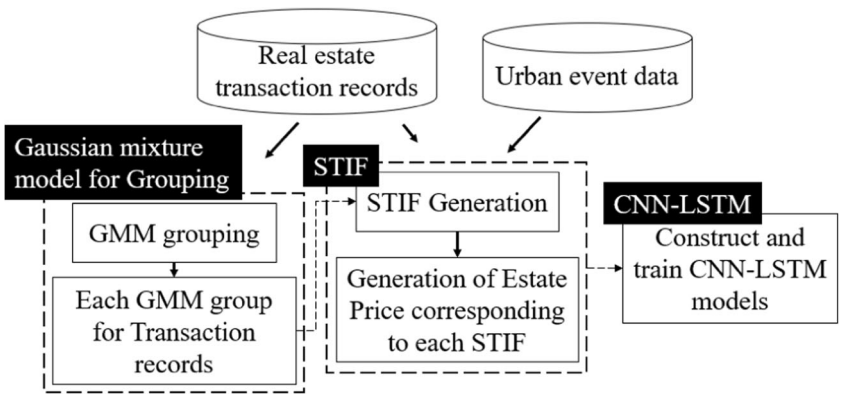

Fig. 4 Architecture of BEPP

Table 2 Pseudo code of BEPP

Input: Real estate transaction records **F** and facility change dataset **E**.
Output: the CNN-LSTM models list M

Estate group set E_G ←group_estate(**F**);
For each estate group g_i in E_G **do**
    S_i←generate_STIF(**F**, **E**);
    O_i←generate_output(S_i, **F**);
    m_i←construct_and_train_CNN-LSTM(S_i, O_i);
    M.append(m_i);
end
Return the CNN-LSTM models list M

function of Gaussian component k; Ψ_k represents the set of the mean m_k and covariance matrix Σ_k of Gaussian component k, that is, $\Psi_k = (m_k, \Sigma_k)$.

- **Step 3:** Using Eq. (2) and an expectation-maximization algorithm [59], we can obtain the optimal solution of Ψ_k with rehard to **F.attrj** in Eq. (1). With the completion of this step, we complete the modeling of the GMM for the jth general feature.

$$g_k(F.attrA; \Psi_k) = \frac{1}{(2\pi)^{\frac{a}{2}}|\sum_k|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(F.attrA-m_k)^T\sum_k^{-1}(F.attrA-m_k)\right). \quad (2)$$

- **Step 4:** Model the GMM for the $j+1$th general feature in the dataset. This step is repeated until the GMMs of all α general features are modeled.
- **Step 5:** After the GMMs of all the general features are modeled, calculate the number of Gaussian functions used to describe the model in each GMM. Finally, cross-combine the distributions of the Gaussian functions of different dimensions to generate the grouping ranges.

If only three general features in the transaction records of an estate, namely, transaction price, transacted floor area, and building age, are considered in the establishment of their separate GMMs, and the GMMs of the three features respectively use 2, 3, and 4 Gaussian functions to describe them, then the estate transaction records will be divided into $2 \times 3 \times 4 = 24$ groups.

- **Step 6:** Sequentially calculate which Gaussian function the value of the kth general feature of each data item $\mathbf{f_i}$ in **F** is in and assign $\mathbf{f_i}$ to a suitable group.
- **Step 7:** After the step above is completed, we can obtain multiple groups, each containing multiple estate transaction records. Next, we concatenate the multiple transaction records for the same estate together to represent the transaction history of each estate to facilitate STIF calculations. Note that in practice, each estate may not have many transaction records, so reasonable time series cannot be formed, which can be a problem for subsequent

Table 3 Pseudo code for grouping estates

Input: Real estate transaction records **F**.
Output: Estate group set E_G

Init list D;
For each transaction record F_j in **F** **do**
    d_j←G($F.attr_j$; Ψ);
    optimal_with_EM_algorithm(d_j);
    D.append(d_j)
end
E_G ←cross_combine_and_divide(D);
E_G ←Concatenate_transaction_records_for_same_estate(E_G);
Return the Estate group set E_G;

calculations. We therefore suggest that users regard the properties with similar conditions in the same community or building as the same estate for the subsequent calculations.

3.2 STIF generation

STIFs are a novel spatiotemporal data structure designed in this study and are mainly used to quantify the influence of the changes in facility features on the price of the estate they surround. The design of STIFs includes three concepts. (1) A set of STIFs is generated for each Estate A, each set containing $m \times n$ STIFs with m denoting the number of features being considered and n indicating the number of additions, changes, or removals made to facilities around Estate A. (2) Each STIF is a two-dimensional matrix with A in the center, the values in each grid of the matrix representing the influence of the facility feature on the price of A. If the influence is positive (i.e., increases the estate price), then the value in the matrix is greater than 0; if not, then the value is less than 0. (3) The sizes of the STIFs in a city and the ranges described in each STIF grid are determined by real estate agencies, thereby indicating how far the influence of a facility feature in this city can reach in terms of estate prices.

To generate STIFs for each estate, we must first define the facility change dataset $\mathbf{E} = (\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_{|E|}})$ in a city, where the ith $\mathbf{e_i}$ can be further expressed as $\mathbf{e_i} = \{eindex, epos_i, etime_i, \mathbf{efea_i}\}$, which respectively represent the number and position of the facility and the time points of additions, changes, or removals. $\mathbf{efea_i}$ indicates the influence of the features of $\mathbf{e_i}$ when $\mathbf{e_i}$ changes. If m_i features are considered for $\mathbf{e_i}$, then $\mathbf{efea_i} = \{efea_{i1}, efea_{i2}, \ldots, efea_{imi}\}$. Furthermore, we record that a total of m features are considered for all of the facilities in the facility change dataset, where $m = m_1 + m_2 + \ldots + m_{|\mathbf{E}|}$.

After defining facility change dataset **E**, we can use **E** to generate the STIFs of target Estate A using three steps. Detailed pseudo code for these steps are presented in Table 4: Steps 1–3 respectively correspond to lines 1, 2, and 4–12 of the pseudo code.

- **Step 1:** Find all of the facility change data within R meters of A from **E**. In this city, only facility changes within R meters of A have an impact on the price of A. Suppose that this step obtains β items of facility change data near A; we can then define the facility change dataset associated with A: $\mathbf{E_A} = \{\mathbf{e_{A1}}, \mathbf{e_{A2}}, \ldots, \mathbf{e_{A\beta}}\}$, where $\mathbf{E_A} \subset \mathbf{E}$.
- **Step 2:** Find the time point set of all facility changes within $\mathbf{E_A}$: $\mathbf{Time_A} = etime_{A1} \cup etime_{A2} \cup \ldots \cup etime_{A\beta} = \{t_1, t_2, \ldots, t_n\}$, where the number of time points in $\mathbf{Time_A}$ must be less than or equal to β because multiple facilities may change at the same time point. With this step, we can derive all of the time points at which STIFs will be generated.
- **Step 3:** Suppose all of the facilities have a total of m features and that there are n time points in $\mathbf{Time_A}$. We can then sequentially generate one STIF for each feature and time point combination. For feature j and time point t_k, we first generate a two-dimensional zero matrix. One by one, it checked whether the time point $etime_{Ai}$ of the ith item in $\mathbf{E_A}$, $\mathbf{e_{Ai}}$, overlaps t_k. If so, it means that at time point t_k, $\mathbf{e_{Ai}}$ is a facility existing within R meters of A and

Table 4 Pseudo code for generating STIFs

```
Input:  Each Estate A and facility change dataset E.
Output: Each Estate A's STIFs list S
1       E_A ← find_change_with_range(A, E, R);
2       Time_A ← sequence_larger_than_β(E_A);
3       Init list S;
4       For each ith time item e_i in Time_A do
5           Init zeros matrix s_i;
6           If overlap with e_Ai in E_A do
7               For feature j in e_Ai do
8                   s_i ← record_influence_value(s_k, efea_ij);
9               end
10          influence_decay_function(s_i);
11          S.append(s_i);
12      end
13      Return the STIFs list S;
```

must be recorded in the STIF. It is recorded by directly recording the influence value $efea_{ij}$ of the jth feature of $\mathbf{e_{Ai}}$ at position $epos_{Ai}$ in the matrix. Each feature exerts influence on the the prices of surrounding estates, and the degree of this influence decreases as the distance between them increases. Using an influence decay function [73], we can calculate the influence of the jth feature of $\mathbf{e_{Ai}}$, $efea_{ij}$, on its matrix surrounding grid q:

$$f\left(epos_{Ap}, q\right) = efea_{ij} \times e^{-\gamma d(eposAp,q)} \tag{3}$$

where $epos_{Ai}$ denotes the position of facility $\mathbf{e_{Ai}}$; q is the position of the grid being calculated; γ is the decay speed ($\gamma > 0$), and $d(epos_{Ap}, q)$ is the Euclidean distance between $epos_{Ai}$ and q. Once this procedure is completed, we have also completed the STIF for the jth feature at the kth time point. These steps are then repeated $m \times n$ times until all of the feature and time point combinations have been addressed.

3.3 Generation of estate Price corresponding to each STIF

After generating the STIFs for each estate based on all estate price features and facility change time points in the previous section, we next explain how to calculate the target estate price corresponding to each STIF. This step is needed because the time points of facility changes are generally different from those of estate transactions, and they do not directly correspond to each other, as shown in Fig. 5.

Pseudo code for the estimation of the estate price corresponding to each STIF is presented in Table 5. Suppose the estate we are considering is A. As mentioned in the previous section, we know that the set of time points of facility changes surrounding A is $\mathbf{Time_A} = \{t_1, t_2, \ldots, t_n\}$. Thus, the estate price corresponding to time point t_k is determined by the transaction prices between time points t_{k-1} and t_k. Note that for the sake of convenience, we additionally define a time point t_0 as the time point of the earliest transaction record in the dataset. Below are the cases that may be encountered when calculating the estate prices corresponding to STIFs.

- Case 1, exactly one transaction record for A exists between t_{k-1} and t_k: In this case, the price of said transaction record is directly designated as the transaction price of t_k (see Table 5, lines 5–6).
- Case 2, two or more transaction records for A exist between t_{k-1} and t_k: In this case, we calculate the mean price of these transaction records and designate the result as the transaction price of t_k (see Table 5, lines 7–8).
- Case 3, no transaction records for A exist between t_{k-1} and t_k: In this case, we cannot calculate the price, so the price of the transaction record at t_{k-1} is directly designated as the transaction price of t_k. If this happens to occur with $k = 1$, it is impossible for us to have the transaction price at t_0 and we cannot use any given number as the transaction price. Thus, we directly eliminate any STIFs at time point t_1 from the dataset (see Table 5, lines 9–15).

Figure 5 presents a simple example of the calculations. First, the estate price corresponding to the STIF at t_1 is $fprice_1$. Because there is only one transaction record between t_0 and t_1, said record is directly designated as the corresponding estate price at t_1. Next, the estate price corresponding to the STIF at t_2 is $fprice_2$. Because there are no transaction records between t_1 and t_2, we designate the corresponding estate price at t_1 as the corresponding estate price at t_2. Finally, the estate price corresponding to the STIF at t_4 is $fprice_4$. Because there are two transaction records between t_3 and t_4, the corresponding estate price is $(fprice_3 + fprice_4)/2$.

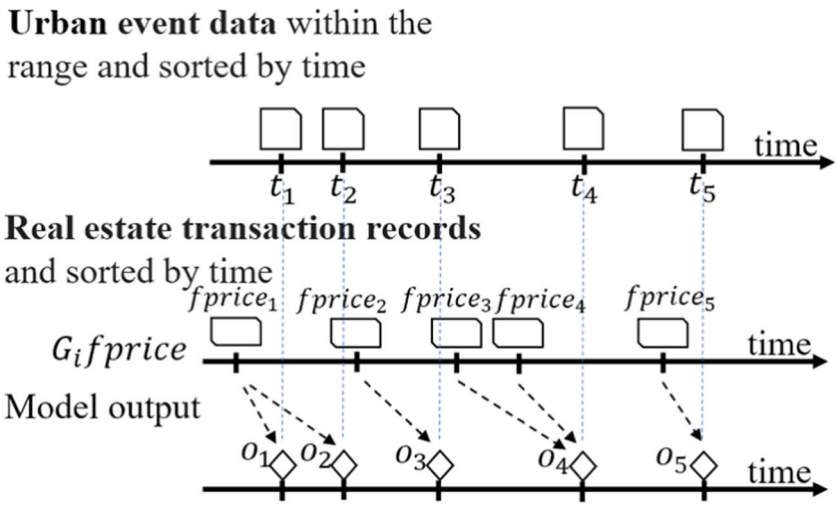

Fig. 5 Schematic of estate price generation

Table 5 Pseudo code for generating estate price corresponding to each STIF

Input: STIFs list S and Real estate transaction records **F**.
Output: STIFs' output list O

```
Init list O;
For each estate A and ith item si in S do
    Init output list oi with same length with si;
    For each time points tk and kth item oik in oi do
        If only_one_record(F, A, tk-1, tk) do
            oik←get_transaction_price(F, A, tk, tk+1);
        Else if large_than_one_record(F, A, tk-1, tk) do
            oik← mean(get_transaction_price(F, A, tk, tk+1));
        Else
            If k equal 1 do
                Drop kth item in si and oi;
            Else do
                oik←get_transaction_price(F, A, tk);
            end
        end
    end
    O.append(oi);
end
Return the STIFs' output list O;
```

3.4 CNN-LSTM

This section explains the details of the CNN-LSTM models used in the BEPP. A CNN-LSTM model is established for all of the estates in each group created using the GMM. Suppose Group G contains $|G|$ items of estate data. There are $m \times n_i$ STIFs for the ith estate g_i in G (m denotes the number of features, and n_i indicates the number of changes that occurred in the facilities surrounding g_i). Thus, the CNN-LSTM models exclusive to G will have $|G|$ items of time series data, and the ith time series will have n_i time points. Each time point has m STIFs and the estate price corresponding to this time point. When a CNN-LSTM model is in use, its inputs are the m STIFs of the same time point, and the output is the estate price at said time point.

Figure 6 displays the architecture of a CNN-LSTM model, which comprises 7 layers: an input layer, a combination of two convolution layers and two max-pooling layers, two convolutional LSTM layers, and two fully-connected layers. During training, the input layer mainly passes on the m STIFs to the subsequent convolution layers and max-pooling layers. The convolution layers and max-pooling layers visit these STIFs using convolution and max-pooling to derive the relationships between the target estate and the surrounding facilities influencing it. Note that map data can be more complex, so we executed two convolution and max-pooling operations. After the data in the m STIFs are processed by these previous layers, the information most useful for predicting the price of the target estate should be extracted. We can then input the information into the two ConvLSTM2D layers and the two full connection layers for time series modeling.

Below, we introduce the equations of each layer in the CNN-LSTM model in detail.

- **Input Data**

Suppose the vendor sets the size of each STIF to be $h \times h$, and our system considers a total of m features. Thus, the input data is an $h \times h \times m$ matrix.

- **Input Layer**

The input layer mainly inputs the STIFs into the model. Thus, the mathematical equation for node i in this layer is

$$O_i^{input} = I_i^{input}. \tag{4}$$

- **Convolution Layer**

The proposed model contains two convolution layers: Convolution-A and Convolution-B. The numbers of kernelsrespectively in these two layers are $2^{\log_2\lceil m*2\rceil}$ and

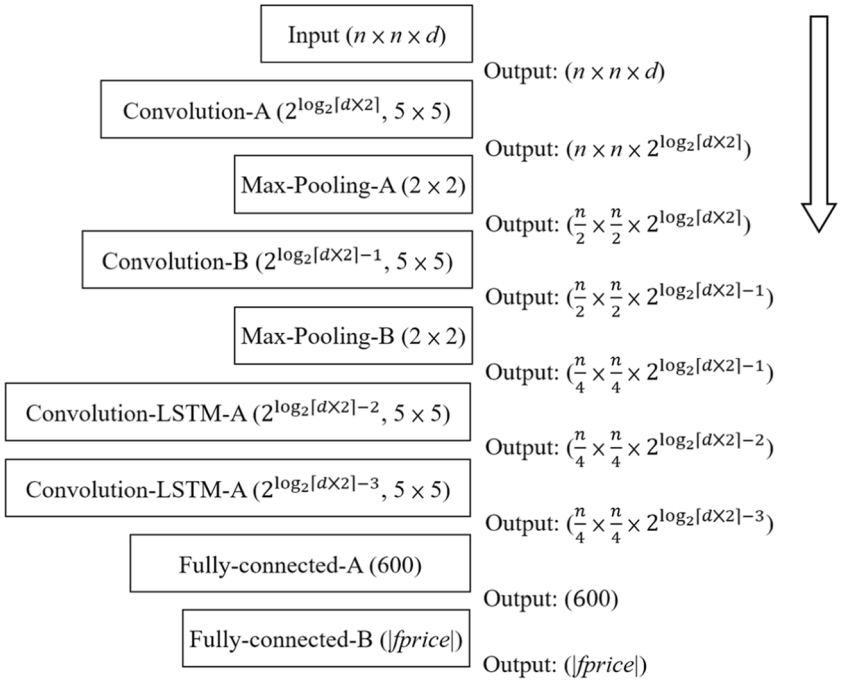

Fig. 6 Architecture of CNN-LSTM model

$2^{\log_2\lceil m*2\rceil} - 1$. Each node in the two layers has k kernels, and the size of each kernel is $s \times s$. For the ith node, the output at position (x, y) can be expressed as:

$$O^i_{(x,y)} = act\left(b_i + \sum_{j=1}^{s}\sum_{k=1}^{s} k^i_{(j,k)} I_{(x+j,y+k)}\right). \tag{5}$$

where $O^i_{(x,y)}$ refers to the output value of the ith node at the (x, y). b_i is the bias vector of the ith node, k^i is the kernel of the ith node, I is the input of the layer, and $act(\bullet)$ refers to the activation function. Note that in this paper, we use the Relu function as the activation function

- **Max-Pooling Layer**:

Suppose the input dimensions of the previous layer are (L_i, W_i, D_i), and the pool size of this layer is (p, p) and stride is s. The mathematical equation for this layer is written as follows:

$$\mathrm{L}_o = \frac{L_i - p_l}{s} + 1, \tag{6}$$

$$\mathrm{W}_o = \frac{W_i - p_w}{s} + 1, \tag{7}$$

$$\text{and } D_o = D_i. \tag{8}$$

where (L_o, W_o, D_o) refer to the output size.

- **ConvLSTM2D Layer**

This layer was formed using the renowned ConvLSTM concept [64] rather than the conventional LSTM concept. Compared to LSTM, ConvLSTM includes an additional convolution operation, so aside from time series information, it can also obtain useful spatial information, which meets the needs of this study. The mathematical equation for this layer is as follows:

$$i_t = \sigma(w_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \circ C_{t-1} + b_i), \tag{9}$$

$$f_t = \sigma\left(w_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \circ C_{t-1} + b_f\right), \tag{10}$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(w_{xc} * X_t + W_{hc} * H_{t-1} + b_c), \tag{11}$$

$$o_t = \sigma(w_{xo} * X_t + W_{ho} * H_{t-1} + W_{co} \circ C_t + b_o), \tag{12}$$

$$H_t = o_t \circ \tanh(C_t). \tag{13}$$

where σ is sigmoid function (i.e., $\sigma(x) = 1/1 + e^{-x}$), $tanh(\bullet)$ is hyperbolic tangent function, '$*$' denotes the convolution operator and '$\circ$' denotes the Hadamard product.

- **Fully-Connected Layer**

Finally, the model uses two fully-connected layers for modeling with the features extracted above. The mathematical equation from input node i to node j in this layer is written as follows:

$$o_j = \tanh\left(t_j \times w_{ij}\right) + b_j. \tag{14}$$

where o_j represent the output of a given node, $tanh(\bullet)$ is the tangent sigmoid function, w_{ij} is the corresponding weight, and b_i is the bias vector corresponding to the jth neuron.

4 Lightweight framework for estate Price prediction

The BEPP introduced in the previous chapter can consider both the temporal and spatial features of facility changes surrounding estates for estate price prediction. However, the CNN-LSTM models used by the BEPP must calculate the two-dimensional matrix data of numerous STIFs, making them time-consuming in both training and usage. Consequently, it is not suitable for users to make instant queries with online. To address this issue, we propose a Lightweight Framework for Estate Price Prediction (LEPP). The framework of the LEPP includes two major stages, for which pseudo code is presented in Table 6. The first stage uses an extraction model to rank the features influencing estate prices in each estate group (lines 1–4 of Table 6), which is achieved by modifying the CNN-LSTM model with the dropout concept. The second stage extracts the top k most important features from the first stage and converts the STIFs of these features into time series which are then input into the lightweight model for modeling (lines 5–8 of Table 6). With this method, we anticipate that the significant decrease of price-influencing features being input into the model and the conversion of the data format of the input data from two-dimensional STIFs into one-dimensional time series can greatly reduce the computational costs. As for the accuracy of the model, the lightweight model is established using the features recognized by the extraction model as having the most importance for estate prices, so its effectiveness will not be too far from the original CNN-LSTM models. On the whole, the lightweight model will also be more suitable for practical applications than the CNN-LSTM models. Below, we introduce the two stages of the LEPP.

4.1 Extraction model for ranking the importance of features

This section is divided into two parts: (1) the establishment of the extraction model and (2) the ranking of the features in the extraction model based on their importance to estate prices. The architecture of the extraction model that we designed is similar to that of the CNN-LSTM model previously described. The only difference is an additional lighten dropout layer between the input layer and the Convolution-A layer, as shown in Fig. 7. Note that the concept of the dropout layer is extended from the design concept of [9]. However, the approach in [9] cannot be applied to matrix data, so we designed a new method that can be applied to matrix data. Suppose that the system considers a total of m estate price features, which means that we will use m nodes in the lighten dropout layer of the extraction model, each node corresponding to an estate price feature. Next, each node will contain a parameter θ and random variable u, which can be used to calculate lighten dropout value X corresponding to the estate price feature being processed. Once the STIFs of each feature are input into this layer, X strengthens or weakens them, which influences the calculations in the subsequent network layers. For instance, Fig. 8 presents the two possible lighten dropout value processes for the STIF of an estate price feature at a time point; Fig. 8(b) shows the weakening process, and Fig. 8(c) displays the strengthening process. It is clear that after weakening, the resulting values in the matrix are all close to 0. Under these circumstances, this STIF will have almost no impact on the calculation results of the layers after the lighten dropout layer. We can thus say that the feature corresponding to the STIF does not influence the final estate price prediction. In contrast, the values in the matrix of

Table 6 Pseudo code of LEPP

Input: CNN-LSTM models list M, STIFs list S, STIFs' output list O and import feature size k.
Output: the Lightweight models list L

```
1   For each estate group CNN-LSTM model m_i in M do
2       i_i ← get_lighten_dropout_layer_init_value(m_i);
3       e_i ← construct_and_train_extraction_model(S_i, O_i);
4       r_i ← rank_importance_feature(e_i);
5       r_i ← top(r_i, k);
6       I_i ← generate_1d_time_series_input(S_i, r_i);
7       O_i ← pick(O_i, r_i)
8       l_i ← construct_and_train_lightweight_model(S_i, O_i);
9       L.append(l_i);
10  end
11  Return the Lightweight models list L
```

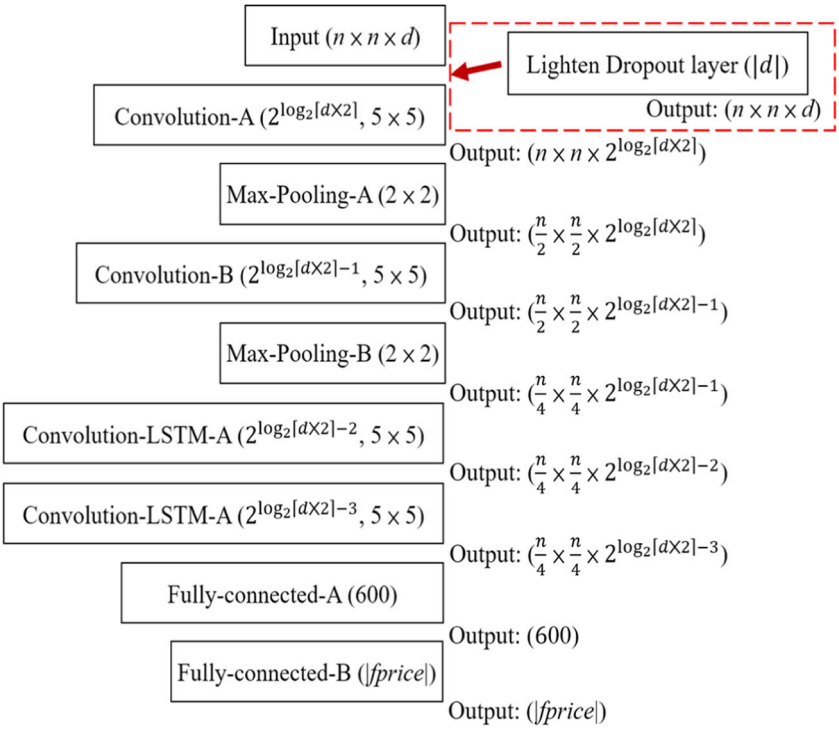

Fig. 7 Extraction model architecture

the strengthened STIF are extremely high and thus exert a significant influence on the calculation results of the layers after the lighten dropout layer. In other words, the feature corresponding to this STIF influences the final estate price prediction. Below, we introduce the details of the extraction model architecture. However, most of it is similar to that of the CNN-LSTM models, so we will only introduce the equations of the newly added lighten dropout layer.

The lighten dropout layer that we designed has two equations. The equations of the ith neuron in this layer can be written as:

$$X = sigmoid\left(\frac{1}{\eta}(\log\theta_i - \log(1-\theta_i) + \log u_i - \log(1-u_i))\right) \quad (15)$$

$$O_i^{dropout} = I_i \times (1-X) \quad (16)$$

Eq. (15) is used to calculate X. In the equation, θ_i is the target learning parameter that will be adjusted as the network is being trained; u_i is a random variable in Uniform(0, 1), and η is a value between 0 and 1. Here, we set η as 0.1 as suggested by [9] to obtain optimal results. Readers may refer to [9] for more details. Eq. (16) uses X to strengthen or weaken the input STIFs, and I_i and $O_i^{dropout}$ respectively denote the input and output of the ith node.

We next discuss how the importance ranking of the estate price features is obtained using the extraction model. In theory, we can use the magnitude of X to directly rank the estate price features by importance [9]. According to Eq. (16), a smaller X means that the estate price feature corresponding to this X is more important, whereas a greater X means that this estate price feature is less important. However, we found

Fig. 8 Lighten dropout value processing: **a** STIF of an estate price feature at a time point; **b** weakening of STIF; **c** strengthening of STIF

that this approach is flawed because the u_i in Eq. (15) is a random variable that is not adjusted as the network is trained. Under these circumstances, the convergence results of X in Eq. (16) will be greatly influenced by u_i, and the value of X will be different in every training session, which will prevent us from obtaining the importance ranking of the estate price features later on. To overcome this issue, we suggest that the target learning objective θ_i in Eq. (15) be calibrated before network training to avoid the problems encountered in [9].

This study proposes using the output matrix values of the convolution layer in the CNN-LSTM model to calibrate θ_i beforehand. This method can be divided into three steps. Suppose the output matrix of the convolution layer is **Oc**. The first step is to derive the absolute values of the all the values in **Oc**. This step exists because the STIFs input into the model may contain both positive values and negative values. However, our discussion of estate prices may neglect positive values and only consider those with a greater impact on estate prices. Suppose the output of the first step is **Oc'**. The second step is to derive the k largest values in **Oc'** and average them. Note that this step does not consider all of the values in **Oc'**; only the k largest values are considered. This is because in a real environment, there is rarely many facilities surrounding an estate, which means that most of the values in the STIF are 0. Taking all of these 0 values into account would distort the results. Suppose the average values corresponding to all m features in the second step are $v_1, v_2, \ldots, v_m$. The third step is to rank these values in descending order of magnitude. Next, the initial value of each feature is determined based on the ranking results; those with higher rankings have higher initial values, whereas those with lower rankings have lower initial values. The purpose of this was to give higher initial values to more important features at the very beginning so that poorer random results from the random variable u does not prevent the more important features from being selected by the extraction model. As for the features that may be less important, their initial values are smaller to begin with, so no matter what the random results of the random variable u are, they will not be selected by the extraction model. This then guarantees that the extracted features will be important for modeling. The user can determine how the initial values are set, such as setting the initial value of the top ranking feature as 1 and then decreasing the initial value at equal differences until the initial value of the last feature is 0, or normalizing $v_1, v_2, \ldots, v_m$ to (0, 1). The more important features just have to have higher initial values than the other features. Once the initial values are set, we can input the same dataset as that of the CNN-LSTM model into the extraction model for training and then obtain the ranking of the X values corresponding to the various estate price features after training. Higher ranking features are more important in estate price prediction.

4.2 Lightweight model construction

After obtaining the importance ranking of the estate price features, we next discuss how the lightweight model is established. This can be divided into two parts: (1) the conversion of the original STIFs into time series data and (2) the introduction to the architecture of the lightweight model, which can use time series data for training.

In the method to convert the STIFs into time series data, suppose an Estate A has $m \times n$ STIFs (m denoting the number of features and n indicating the number of changes made to facilities surrounding Estate A). The STIFs regarding the same feature but with different time points can be organized into a time series. In other words, we can generate m time series for A, each of which has values for n time points. The value corresponding to each time point is the value of the grid where A is situated in the STIF corresponding to said time point, which represents influence of the facilities surrounding A on the estate price of A at this time point. Of course, this approach will cause the spatial information stored in the STIFs to be lost, which may reduce the accuracy of the model. However, to reduce the computational costs of the model, we believe that this information loss is acceptable, which our experiment simulations will also demonstrate.

After obtaining the aforementioned time series data, we next establish the lightweight models of the targets, one for each group established by the GMM. Suppose Group G contains $|G|$ items of estate data, with m features for the ith estate g_i in G and n_i changes having occurred in the facilities surrounding g_i. When g_i is input into the lightweight model, m time series will be sequentially input, each with n_i time points. The output of the lightweight model is the estate price of g_i at each time point (derived using the calculations in Section 3.3). The lightweight model of a target is the simplest RNN model, the architecture of which is displayed in Fig. 9. Note that the model is extremely simple and common, so we will not go into its mathematical equations here. Readers may refer to [13, 49, 70] for the mathematical equations of this model.

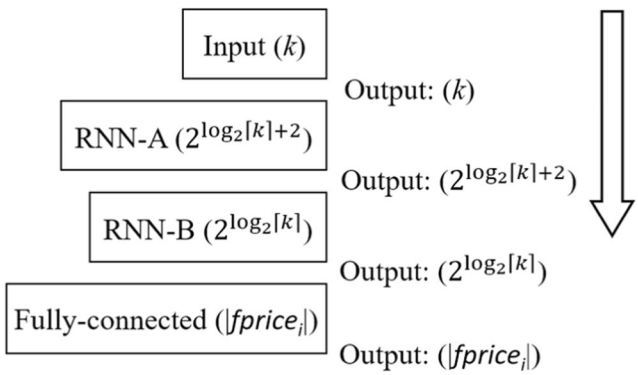

Fig. 9 Lightweight model architecture

Compared to the previously mentioned CNN-LSTM models, fewer features are input into the lightweight models, and the amount of data for feature is significantly reduced as well (changed from two-dimensional matrices to one-dimensional time series). For these reasons, the computational costs of the lightweight model should be much lower than those of the CNN-LSTM model. Furthermore, many researchers have demonstrated that simple lightweight RNN models can be directly applied to query webpages, and this shows that lightweight models can be applied to actual estate price queries.

5 Simulations

This chapter verifies that the BEPP and LEPP proposed in this study can indeed improve the accuracy of estate price prediction while reducing the number of parameters. The chapter is divided into four sections, including (1) an introduction to the experimental datasets collected for this study, (2) an examination of the reasonableness of the pre-grouping results of the estate transaction records, (3) the prediction accuracy, performance, and number of parameters of the two framework models BEPP and LEPP, as well as the reasonableness of the feature ranking, and (4) a comparison of the prediction accuracy of BEPP and LEPP with that of existing estate price prediction models. In this study, all the experiments were implemented in python running Windows 10 on 64-bit OS and an Intel Core i7–8700 processor at 3.2GHz with 6 cores, 64 GB of memory, and graphics card for EVGA GeForce GTX 1080 SC GAMING ACX 3.0. Furthermore, the experiment data provided are the means of the results of the experiments performed 5 times.

5.1 Introduction to datasets

The estate transaction records in this paper come from the Ministry of the Interior of Taiwan [94]. We collected a total of 27,988 transaction records involving estates in the Xitun District of Taichung City, Taiwan between 2007/06/05 and 2018/01/27. Each transaction record includes eight features, as shown in Table 7. Facility change data were obtained from two sources: Business registration data from the Department of Commerce of the Ministry of Economic Affairs – Commerce Industrial Service Portal [93], and the construction license stub from the OpenData Platform of the National Building Management Information System [92]. The time period of the dataset in [93] was from October, 2006 to January, 2018, and the time period of the dataset in [92] was from 1998 to January, 2018. The two datasets contained a total of 33,556 items of data. To enrich the data, we also included the estate transactions from the Ministry of the Interior in Taiwan [94] as part of the facility change dataset. Table 8 displays the features of the facility change dataset. Note that business type is categorical and cannot be directly used in the simulations. We therefore added a feature for every category and used to 0/1 to express whether a facility has this feature.

Table 7 Features of estate transaction records

Features	Unit
land shifting total area	m^2
build shifting total area	m^2
number of bedrooms	Unit
number of living/dining rooms	Unit
number of bathrooms	Unit
Building price	USD
Building age	Years
Unit price	USD/m^2

We chose Xitun District of Taichung City, Taiwan as our study area. This choice was made because Taichung City is the second most populous municipality in Taiwan, and Xitun District has the most frequent estate transactions and most prosperous commerce in Taichung City. As shown in Fig. 10, there are two main roads going through the downtown area of Taichung. The horizontal dotted line is Provincial Highway No. 12, and the vertical dashed line is Provincial Highway No. 74. There are two important landmarks in this area: Taichung Railway Station in the lower right corner and the National Taichung Theater in the lower half of the figure. The downtown area of Taichung can basically be divided into two zones. Zone A is where the older estates area. Most of the estates in this area were built during the old Taichung period. It is closer to Taichung Railway Station, and there are numerous businesses and amenities here. Zone B is Taichung's 7th Redevelopment Zone [95]. Since 2000, it has been gradually replacing the old downtown area with intensive public facility development and numerous department stores and hotels opening there. It is near the highway interchange and has a good public transport network. Currently, the land value there is the highest in Taichung City, and it is the most exclusive area. There are many large luxury estates in Zone B, and this has prompted refurbishing and rebuilding in the areas around Zone A. The above shows that many different types of estates exist in this district at the same time, including old estates, luxury houses, refurbished buildings, and new construction projects, thereby making it suitable for the study area in this paper.

5.2 Results of GMM grouping

In this section, we discuss the rationality of using GMM to group the estates by some features listed in Table 7, including build shifting total area, and the price and age of the building.

Table 8 Facility change dataset

Column name	Explanation
Business type: branch	Whether the business is a branch
Business type: partner	Whether the business is a partner
Business type: branch of partnering organization	Whether the business is a branch of a partnering organization
Business type: branch of foreign company	Whether the business is a branch of a foreign company
Business type: limited company	Whether the business is a limited company
Business type: sole proprietorship	Whether the business is a sole proprietorship
Business type: branch of sole proprietorship	Whether the business is a branch of a sole proprietorship
Business type: company limited	Whether the business is a company limited
Business capital	Amount of business capital
Building: foundation area	Foundation area of said building (m^2)
Building: building area	Building area of said building (m^2)
Building: total floor area	Total floor area of said building (m^2)
Building: height	Height of said building (m)
Building: area of underground shelter	Area of underground shelter in said building (m^2)
Building: legal space area of building	Legal space area of said building (m^2)
Building: number of floors above ground	Number of floors above ground in said building (Floors)
Building: number of floors below ground	Number of floors below ground in said building (Floors)
Building: number of buildings	Number of buildings in said building
Building: number of units	Number of units in said building
Unit price	Unit price of estate (USD/m^2)

In the GMM parameter settings, we know there are three primary types of estates are in the study area based on the introduction in Section 5.1. For this reason, we set the number of Gaussian functions used by each feature in the GMM as 3.

Note that if too few intervals are used, we will not be able to accurately group the transaction records with the same trends; if too many intervals are used, the data groups will be more detailed, but the lower numbers of data items will mean inadequate training data for the model, which will affect the final modeling results. Table 9 presents the ranges of the feature groups derived with this parameter. The intervals for the total transaction prices of the estates were 1 K to 400 K, 430 K to 1.76 M, and greater than 1.76 M. For estate age, the intervals were − 5 years to 11 years, 11 years to 27 years, and greater than 27 years. The negative values for estate age were due to the common practice of selling and buying presale housing in Taiwan. Thus, an estate may be sold long before it has been constructed. The intervals for the total transacted floor area

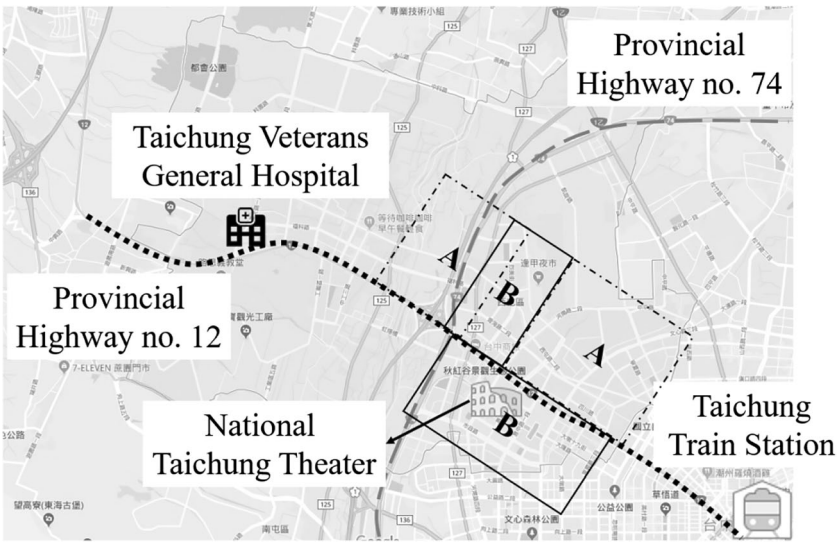

Fig. 10 Introductory map of Xitun District in Taichung City, taiwan

Table 9 Transaction feature ranges in GMM

Transaction feature	Range L	Range M	Range H
Total price (USD)	1 K~400 K	430 K~1.76 M	> 1.76 M
Age (Years)	−5~11	11~27	> 27
Shifting total area(m^2)	1~220	220~710	> 710

were 1 m^2 to 220 m^2, 220 m^2 to 710 m^2, and greater than 710 m^2. With the feature groups above, we can ultimately derive $3^3 = 27$ groups. The numbers of transaction records in each group and their Pareto Chart are as shown in Table 10 and Fig. 11. From Table 10 and Fig. 11, we can see that the collected datasets include three major transaction groups (Groups 18, 7, and 19). The numbers of transaction records in these three groups account for over 80% of the total, which statistically means that these three groups are adequately representative of the whole. We therefore only discussed the modeling effectiveness of BEPP and LEPP when applied to these three groups. Below we explain the implications and areas of these three data groups.

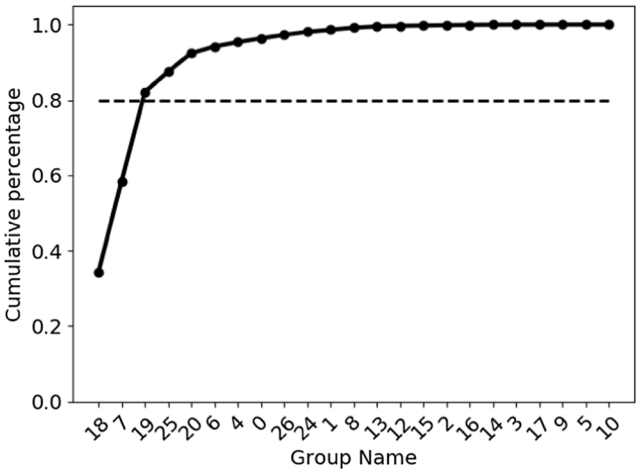

Fig. 11 Pareto chart of Table 10

5.2.1 Group 18

From Table 11, we can see that the area of the transactions associated with the largest group of clusters account for only a small portion of the properties with lower prices and smaller area 1.22 ~ 222.82 (m^2); however, this group actually makes up a large proportion of the total estate sales. In other words, the most common sales are those associated with general public trading within an area of moderate size and relatively low prices (i.e., low price area). Group 18 involves the trading of middle-aged estate by the general public. The distributions of this phenomenon are identical to those in this group. Figure 12 exhibits the distributions of the estates in this group. As can be seen, they are right within Zone A in Xitun District and fit the description of this zone, which contains many of the old residences in Taichung City. This phenomenon is identical to the data conditions of Group 18 obtained above.

5.2.2 Group 7

The area of the transactions in group 7 was $223m^2 \sim 710\ m^2$, the total transaction values were in the middle of the overall numerical range, and the houses are newer. From this we can infer that this group is engaged in transactions involving luxury houses. From the map in Fig. 13, we can see that most of the estates in this group fall within Zone B and part of Taichung's 7th Redevelopment Zone around the National Taichung Theater and along Provincial Highway No. 74. The estates in this area are all large, high-priced luxury houses or commercial buildings. This shows that

Table 10 GMM grouping results

Group No.	Group size	Number of points after conversion	Group No.	Group size	Number of points after conversion
18	9438	240	13	93	2
7	6625	100	12	43	0
19	6511	81	15	25	0
25	1456	30	2	24	0
20	1365	16	16	21	0
6	489	2	14	14	0
4	321	9	3	6	0
0	271	1	17	2	0
26	259	1	9	2	0
24	217	1	5	1	0
1	154	2	10	1	0
8	151	0			

Table 11 Deconstruction of GMM groups

Group No.	Shifting total area (m^2)	Total price (USD)	Build age(year)
18	L(1.2~222.8)	L($1000~$430,000)	M(12~27)
7	M(223.1~709.9)	M($430,000~$ 2.1 million)	L(−5~12)
19	L(5.9~222.5)	L($2000~$430,000)	L(−3~12)

the circumstances in the area of this group are identical to our grouping results.

5.2.3 Group 19

Groups 19 and 18 are similar in terms of property area and price. Thus, the properties in both groups are suitable for the general public. However, the age of the properties in group 19 (3–12 years) is less than those in group 18; i.e., the properties are newer. In terms of location, as shown in Fig. 14, we can see that most of the estates in this group are scattered throughout and around Zones A and B. Most of the estates in this group are part of residential clusters that were already there. Refurbishing and rebuilding later on increased the amenities in these areas, attracting even more new residences to be building in the surrounding areas. The grouping results and the distributions of this group prove thus.

5.3 Results and explorations of BEPP and LEPP

In this section, we examine the performance of BEPP and LEPP with regard to different groups, including feature ranking, and the prediction accuracy and the number of parameters of the BEPP and LEPP. Note that to ease our discussions on this simulation, we will only extract five features for LEPP.

As shown in Table 12, we can see the top 5 features of each group. The features unit price of surrounding estates and partner-type business are among the top features in all three groups. The unit price of surrounding estates is reasonable because the price of an estate is highly correlated with the prices of surrounding estates. As for why partner-type business was selected by the extraction model, we believe it is because Xitun District is a place where businesses thrive; there are businesses surrounding almost every estate in this area. Most of these businesses are small businesses or user self-employed partner-type businesses. Aside from these two features, the remaining features shown in Table 12 are commerce-related or building-related. First of all, the other three features selected by the extraction model for Group 18 are all commerce-related. This is extremely reasonable because, as we discussed in Section 5.1, Group 18 mainly consists of estates within old communities. Users buying or selling such estates generally care more about the amenities than about building-related features. As a result, only commerce-related features were selected by the extraction model, and no building-related features were selected. As for Group 7, the estates in this group are mostly luxury houses and business-type estates. Users that purchase luxury houses generally pay more attention to building-related features such as building statutory area and building build area than to commerce-

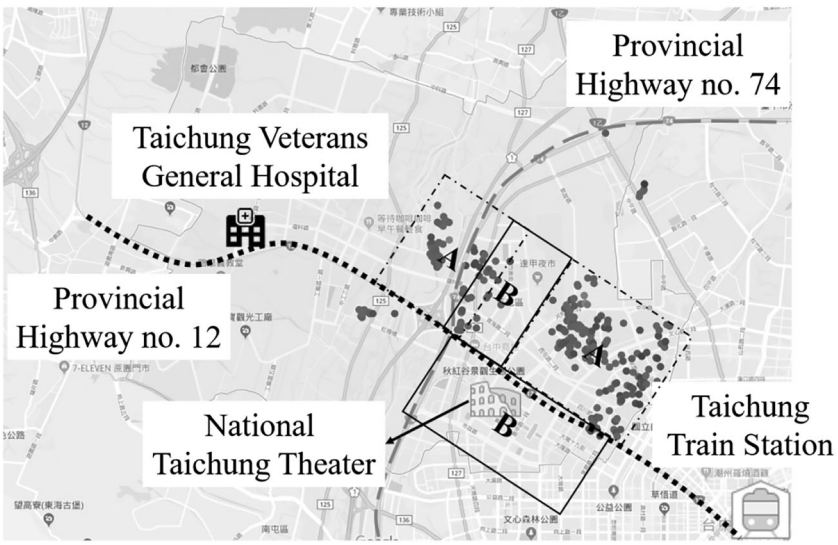

Fig. 12 Distributions of estates in Group 18

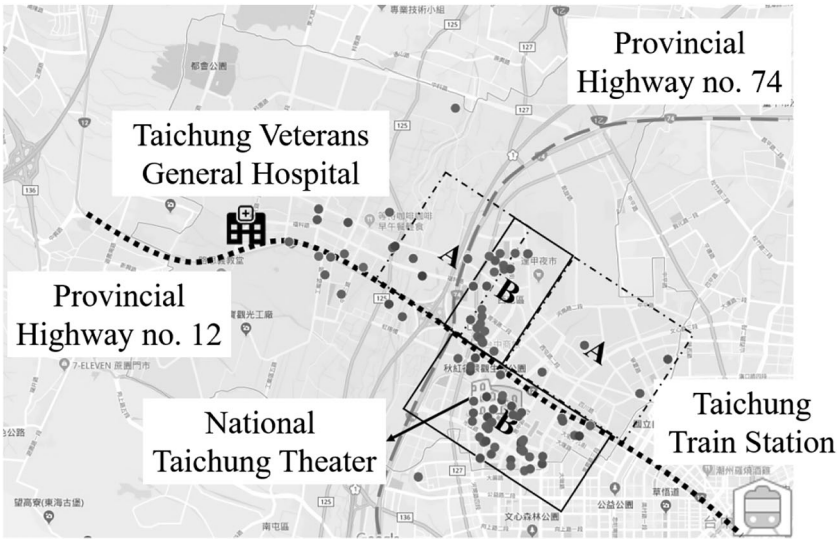

Fig. 13 Distributions of estates in Group 7

related features. Users that purchase business-type estates generally purchase them for companies, so they pay attention to both the size, such as building statutory area and building build area, and the nearby surroundings (business partnerships). It is thus reasonable that the three other features selected by the extraction model for Group 7 were building statutory area, building build area, and business partnerships. Finally, for Group 19, we found that the other three features selected by the extraction model were associated with public space, including building number builds, building under layers, and building base area. These are also extremely reasonable, considering the actual circumstances; most of the estates in this group are newly constructed or refurbished estates. Users that purchase such estates generally attach more importance to building-related features, such as a larger courtyard, public facilities, and more parking spaces. This is also reflected in the results of Table 12, thereby demonstrating the effectiveness of our proposed approach even further.

Table 13 presents the accuracy of using CNN-LSTM models and lightweight models for modeling in each group. As can be seen, the CNN-LSTM models performed best in Group 19 and then in Group 7 and Group 18. The true errors were 188.5 (USD/m^2), 268.8 (USD/m^2), and 371.6 (USD/m^2), respectively. The true error in Group 19 with regard to the highest unit price 5181.80 (USD/m2) was approximately 3.64%. The true error in Group 7 with regard to the highest unit price 5574.08 (USD/m2) was approximately 4.82%, and the true error in Group 18 with regard to the highest unit price 4826.77 (USD/m2) was approximately 7.70%. These results demonstrate the effectiveness of pairing CNN-LSTM models

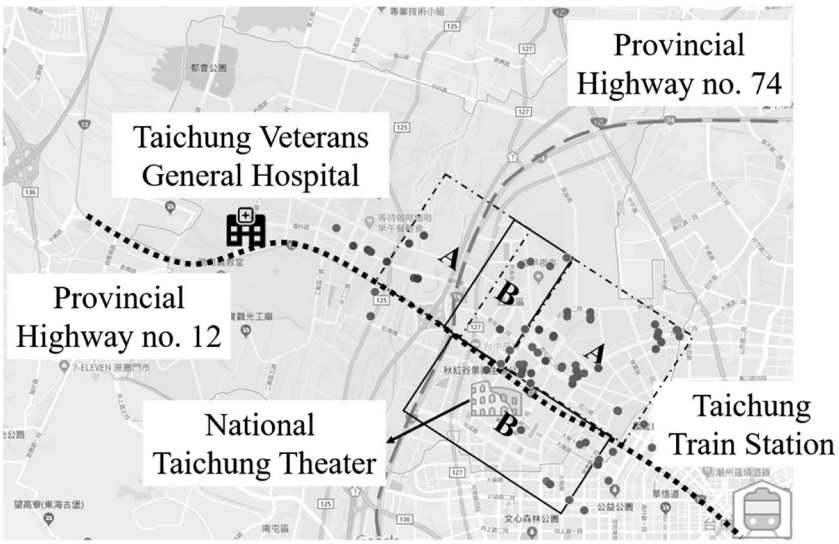

Fig. 14 Distributions of estates in Group 19

Table 12 Each Group TOP-5 features

Group 18	Group 7	Group 19
around unit price	around unit price	around unit price
type: sole proprietorship	type: partner	type: partner
store money	type: branch	building number builds
type: partner	building statutory area	building under layers
type: branch	building build area	building base area

with STIFs. The lightweight models performed best in Group 18 and then in Group 19 and Group 7. The true errors were 313.5 (USD/m^2), 313.9 (USD/m^2), and 426.9 (USD/m^2), respectively. Compared to the respective CNN-LSTM models, the accuracy of the lightweight models was −18.53%, 39.95%, and 37.03%. We found that the lightweight model in Group 18 presented the best performance and performed superior to its CNN-LSTM model. Although the lightweight models in Groups 19 and 7 performed more poorly than the CNN-LSTM models, the comparison of the model frameworks in Table 14 clearly show a substantial difference between the sizes of the CNN-LSTM model and the lightweight model. The former has 12 layers and about 27 M parameters, whereas the latter has only 4 layers and only 1177 parameters, which is 0.0043% of the parameters in the former. With such a great difference in model size, the performance of the lightweight models in the three groups counts as already pretty good, and still, the lightweight model in Group 18 had better prediction accuracy than the CNN-LSTM model. Furthermore, we can also seen in Table 14 that the average run time of the lightweight model is 0.7 s, which is 1/15 of that of the CNN-LSTM model, thereby demonstrating that the proposed LEPP can effectively shorten the run time. In this case, for the example described in Table 1, using the CNN-LSTM model would require four servers performing parallel computations to process a single query in 3 s, which means that 100 queries would require 400 servers. However, with the lightweight model proposed in this study, a single server could handle four queries in 3 s, so the established model would only need 25 servers, which is 1/16 of the number of servers needed for the CNN-LSTM model. This would significantly reduce the hardware requirements for system setup. The experiment results above demonstrate the efficacy and effectiveness of the proposed approach.

Table 13 Prediction accuracy of models in each group

Group	Model	Actual forecast difference(USD/m^2)
18	CNN-LSTM	371.6
	Lightweight	313.5
7	CNN-LSTM	268.8
	Lightweight	426.9
19	CNN-LSTM	188.5
	lightweight	313.9

5.4 Comparison of BEPP, LEPP, and other existing estate price prediction methods

In the previous section we introduced the estate price prediction results for three groups of estates in the Xitun District of Taichung City, Taiwan. This section compares the prediction results of BEPP and LEPP with those of four other existing methods applied to the same dataset. These four methods include the Artificial Neural Network (ANN) method proposed by Lim et al. [44], the linear regression (LG) and support vector regression (SVR) methods proposed by Jain et al. [35], and the Automated Machine Learning Tree-Based Pipeline Optimization Tool (AML TPOT) developed by Masrom et al. [51]. We selected these methods because they were all published in recent papers and provide clear details on the input features and prediction models. They also made kits available on GitHub, which allowed us to obtain and reproduce these algorithms easily. However, the formats of the estate datasets used in the studies above differ from that considered in this study, so in our application the design of the input features was not identical to that of the original studies. For instance, [44, 51] considered the distances from target estates to metro stations and central commercial districts. However, the city selected in this study had no metro prior to 2018, and the area considered is the central commercial district, so we did not consider these two features. In addition, in [35] building materials, including architectural style, outer wall materials, the type of power system, and the type of heating and air-conditioning, were included as input features and this information was not available in the estate datasets used in this study. Aside from the features of the estates themselves, [44]

Table 14 Comparison of CNN-LSTM and lightweight mode

Model	Number of layers	Number of params	Run time(s)
CNN-LSTM	12	27,180,593	11.1723
lightweight model	4	1177	0.7435

also took into account the various economic variables of Singapore, such as the Singapore Real Gross Domestic Product, Singapore Consumer Price Index, prime lending rate, and average monthly wages. Although the government of Taiwan does releases similar information, it is calculated using all of Taiwan as a single unit. Taiwan is 50 times larger than Singapore, and the scope of a single city's economy in Taiwan is several times that of the whole of Singapore. Thus, we did not adopt such information in our simulations. Finally, after removing the fields that our datasets did not include, we selected land shifting total area, building shifting total area, number of bedrooms, number of living/dining rooms, number of bathrooms, and build age as the inputs for comparison.

Table 15 compares the results of the compared methods. The results in the table are not divided by group as are those in Table 13; rather we present the results of the three groups combined. None of the four existing methods grouped the estates in their experiments, so we did the same by combining the data in Groups 7, 18, and 19 in the previous sections into one group and then input it into the existing methods for prediction. The results of AML TPOT are the best among existing methods, which matches expectations since the ANN, LR, and SVR methods did not conduct feature selection before training and input all of the features into the model for training. Under such circumstances, the model will include many fields which do not facilitate price prediction in its calculations, thereby reducing prediction accuracy. AML TPOT on the other hand uses a genetic algorithm to select the most suitable features for prediction before training. However, AML TPOT was still far less accurate than the BEPP proposed in this study. This is due to the differences in input between AML TPOT and BEPP. Only single values are considered for model inputs in AML TPOT, which means only parameters of the estates themselves, and a few quantifiable fields regarding the relationships between surrounding facilities and the target estate such as absolute distance, were considered. BEPP, however, can accept inputs in matrix format. This means it can take into account more information than the AML TPOT, such as the position of surrounding facilities and the target estates, the interaction effect of two different surrounding facilities on the target estates, or changes in the prices of nearby estates. As for the differences between the prediction results of LEPP and AML TPOT, both models only consider single values, so their performance should be similar. However, in Table 15 we see that the prediction accuracy of LEPP is slightly higher than that of AML TPOT. This is because although the inputs of LEPP are single values, these input features were obtained after training and dismantling the BEPP model and thus contain numerous spatial relationships between target estates and surrounding facilities that the inputs of AML TPOT could not consider. These results indicate that the proposed BEPP and LEPP are indeed superior in terms of the accurate prediction of real estate prices.

Table 15 Performance comparison of four existing methods and proposed models

Group	Model	Actual forecast difference(USD/m^2)
7+18+19	ANN [44]	2379.5
	LR [35]	648.1
	SVR [35]	732.7
	AML TPOT [51]	405.5
	BEPP	299.5
	LEPP	374.7

6 Conclusions

Estate price prediction systems are currently one of the most needed systems in the market. Generally speaking, both temporal and spatial features surrounding the target estate must be considered when predicting its price. However, we observed that most of the existing studies only considered one of the two and cound not consider both, thereby resulting in poorer prediction accuracy. We therefore proposed two frameworks to overcome this issue. The first framework, BEPP, predicts estate prices using three steps: (1) dividing estate transaction records with different fluctuation trends into different groups using a GMM, (2) generating STIFs for each estate to quantify the influence of facility changes on estate price, and (3) conducting estate price prediction using a novel CNN-LSTM model designed to go with the STIFs. However, the CNN-LSTM model used in the first framework is a type of DLM, which may be time-consuming when making predictions and thus cannot be directly applied to online queries. We therefore proposed a second framework, LEPP, to reduce the time needed by the estate price prediction system to make predictions. This framework first uses an extraction model to rank all of the temporal and spatial features by their influence on estate price. The LEPP then inputs the most influential features ranked by the extraction model into a lightweight model. On the whole, the lightweight model has fewer input features, and only a shallow RNN was used for modeling, so its run time will certainly be shorter than that of the CNN-LSTM model in BEPP, making it more suitable for online queries. Finally, we used actual estate data from Taiwan to verify that the proposed approach can effectively and swiftly predict estate prices.

References

1. Abdi A, Fekri F (2020) Indirect stochastic gradient quantization and its application in distributed deep learning. Proceedings on AAAI Conference on Artificial Intelligence 34(4):3113–3120
2. Ankit A, Hajj IE, Chalamalasetti SR, Agarwal S, Marinella M, Foltin M, Strachan JP, Milojicic D, Hwu WM, Roy K (2020)

PANTHER: a programmable architecture for neural network training harnessing energy-efficient ReRAM. IEEE Trans Comput 69(8):1128–1142
3. Anselin L (2010) Thirty years of spatial econometrics. Pap Reg Sci 89(1):3–25
4. Aravindkumar S, Varalakshmi P, Hemalatha M (2020) Generation of image caption using CNN-LSTM based approach. Thermal Stresses—Advanced Theory and Applications:465–474
5. Barber S (2007) "How fast does a website need to be?" PerfTestPlus Inc. Florida, United States of America
6. Bency AJ, Rallapalli S, Ganti RK, Srivatsa M, Manjunath BS (2017) Beyond spatial auto-regressive models: predicting housing prices with satellite imagery. Proceeding on IEEE Winter Conf Applications of Computer Vision:320–329
7. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum Press
8. Cavallaro G, Mura MD, Benediktsson JA, Plaza A (2016) Remote sensing image classification using attribute filters defined over the tree of shapes. IEEE Trans Geosci Remote Sens 54(17):3899–3911
9. Chang CH, Rampasek L, Goldenberg A (2018) Dropout feature ranking for deep learning models, arXiv:1712.08645
10. Chen YC, Li DC (2020) Selection of key features for PM2.5 prediction using a wavelet model and RBF-LSTM, to be published by Applied Intelligence
11. Chen F, Song L, Chen Y (2018) ReGAN: a pipelined ReRAM-based accelerator for generative adversarial networks. Proceeding on Asia and South Pacific Design Automation Conference
12. Cheng T, Wang J, Haworth J, Heydecker B, Chow A (2014) A dynamic spatial weight matrix and localized space-time autoregressive integrated moving average for network modeling. Geogr Anal 46(1):75–97
13. Chin TL, Chen KY, Chen DY, Lin DE (2020) Intelligent real-time earthquake detection by recurrent neural networks. IEEE Trans Geosci Remote Sens 58(8):5440–5449
14. Chudik A, Pesaran MH (2011) Infinite-dimensional VARs and factor models. J Econ 163(1):4–22
15. Cui Z, Henrickson K, Ke R, Wang Y (2020) Traffic graph convolutional recurrent neural network: a deep learning framework for network-scale traffic learning and forecasting. IEEE Trans Intell Transp Syst 21(11):4883–4894
16. Das S, Gupta R, Kabundi A (2011) Forecasting regional house price inflation: a comparison between dynamic factor models and vector autoregressive models. J Forecast 30:288–302
17. Dong Y, Long T, Jiao W, He G, Zhang Z (2018) A novel image registration method based on phase correlation using low-rank matrix factorization with mixture of Gaussian. IEEE Trans Geosci Remote Sens 56(1):446–460
18. Dong W, Liang J, Xiao S (2020) Saliency analysis and Gaussian mixture model-based detail extraction algorithm for Hyperspectral Pansharpening. IEEE Trans Geosci Remote Sens 58(8):5462–5476
19. Du J, Chen D, Wang R, Peethambaran J, Mathiopoulos PT, Xie L, Yun T (2019) A novel framework for 2.5-D building contouring from large-scale residential scenes. IEEE Trans Geosci Remote Sens 57(6):4121–4145
20. Duan P, Mao G, Zhang C, Wang S (2016) STARIMA-based traffic prediction with time-varying lags. Proceeding on IEEE Int Conf on Intelligent Transportation Systems:1610–1615
21. Feng R, Luthi S, Gisolf D, Angerer E (2018) Reservoir lithology determination by hidden Markov random fields based on a Gaussian mixture model. IEEE Trans Geosci Remote Sens 56(11):6663–6673
22. Gadhiya T, Roy AK (2020) Superpixel-driven optimized Wishart network for fast PolSAR image classification using global k-means algorithm. IEEE Trans Geosci Remote Sens 58(1):97–109
23. Guo S, Lin Y, Feng N, Song C, Wan H (2019) Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. Proceeding on AAAI Conf on Artificial Intelligence 33(1):922–929
24. Gupta R, Das S (2010) Predicting downturns in the US housing market: a Bayesian approach. J Real Estate Financ Econ 41:294–319
25. Han S, Pool J, Tran J, Dally W (2015) Learning both weights and connections for efficient neural network. Adv Neural Inf Proces Syst:1135–1143
26. Han S, Liu X, Mao H, Pu J, Pedram A, Horowitz MA, Dally WJ (2016) EIE: efficient inference engine on compressed deep neural network. Proceeding on ACM/IEEE Int Symp on Computer Architecture 44:243–254
27. Han S, Mao H, Dally WJ (2016) Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. Proceeding on Int. Conf. on Learning Representations
28. Hassibi B, Stork DG (1993) Second order derivatives for network pruning: Optimal brain surgeon. Advances in neural information processing systems, pp. 164–171
29. Havens S, Marshall HP, Pielmeier C, Elder K (2013) Automatic grain type classification of snow micro penetrometer signals with random forests. IEEE Trans Geosci Remote Sens 51(6):3328–3335
30. Hengl T, Nussbaum M, Wright MN, Heuvelink GBM, Gräler B (2018) Random forest as a generic framework for predictive modeling of spatial and spatio-temporal variables. PeerJ 6:e5518
31. Hoxmeier JA, DiCesare C (2000) System response time and user satisfaction: an experimental study of browser-based applications. Proceeding on Americas Conf. on Information Systems
32. Huang S, Ankit A, Silveira P, Antunes R, Chalamalasetti SR, Hajj IE, Kim DE, Aguiar G, Bruel P, Serebryakov S, Xu C, Li C, Faraboschi P, Strachan JP, Chen D, Roy K, Hwu WM, Milojicic D (2021) Mixed precision quantization for ReRAM-based DNN inference accelerators. Proceeding on Asia and South Pacific Design Automation Conf:372–377
33. Huang CJ, Kuo PH (2018) A deep CNN-LSTM model for particulate matter (PM2.5) Forecasting in Smart Cities. Sensors 18(7):2220
34. Ichihashi H, Honda K, Notsu A, Miyamoto E (2008) FCM classifier for high-dimensional data. Proceeding on IEEE Int Conf on Fuzzy Systems:200–206
35. Jain M, Rajput H, Garg N, Chawla P (2020) Prediction of house pricing using machine learning with Python. Proceeding on Int Conf on Electronics and Sustainable Communication Systems: 570–574
36. Kamarianakis Y, Prastacos P (2006) Spatial time-series modeling: a review of the proposed methodologies. University of Crete, Department of Economics, Working Papers
37. Kluska P, Zięba M (2020) Post-training quantization methods for deep learning models. Proceeding on Asian Conf on Intelligent Information and Database Systems:467–479
38. Lacombe DJ, Michieka NM (2018) Forecasting China's industrial output using a spatial Bayesian vector autoregressive model. Growth Chang 49(4):712–742
39. Lang H, Xi Y, Zhang X (2019) Ship detection in high-resolution SAR images by clustering spatially enhanced pixel descriptor. IEEE Trans Geosci Remote Sens 57(8):5407–5423
40. LeSage JP, Krivelyova A (1999) A spatial prior for Bayesian vector autoregressive models. J Reg Sci 39(2):297–317
41. LeSage JP, Pan Z (1995) Using spatial contiguity as Bayesian prior information in regional forecasting models. Int Reg Sci Rev 18(1): 33–53
42. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2017) Pruning filters for efficient convnets. Proceeding on Int. Conf. on Learning Representations
43. Li T, Hua M, Wu X (2020) A hybrid CNN-LSTM model for forecasting particulate matter (PM2.5). IEEE Access 8:26933–26940

44. Lim WT, Wang L, Wang Y, Chang Q (2016) Housing price prediction using neural networks. Proceeding on Int. Conf, Natural Computation, Fuzzy Systems and Knowledge Discovery
45. Liu L, Jia N, Lin L, He Z (2019) A cohesion-based heuristic feature selection for short-term traffic forecasting. IEEE Access 7:3383–3389
46. Liu Z, Zhou P, Li Z, Li M (2019) Think like a graph: real-time traffic estimation at City-scale. IEEE Trans Mob Comput 18(10): 2446–2459
47. Liu M, Zhou M, Zhang T, Xiong N (2020) Semi-supervised learning quantization algorithm with deep features for motor imagery EEG Recognition in smart healthcare application. Applied Soft Computing 89:106071
48. Luo J, Wu J, Lin W (2017) ThiNet: a filter level pruning method for deep neural network compression. Proceeding on IEEE Int Conf on Computer Vision:5068–5076
49. Maggiori E, Charpiat G, Tarabalka Y, Alliez P (2017) Recurrent neural networks to correct satellite image classification maps. IEEE Trans Geosci Remote Sens 55(9):4962–4971
50. Marchesan GC, Carara EA, Zanetti MS, De O, Leonardo L (2019) Exploring the training and execution acceleration of a neural network in a reconfigurable general-purpose processor for embedded systems. Proceeding on IEEE Int New Circuits and Systems Conf:1–4
51. Masrom S, Mohd T, Jamil NS, Rahman ASA, Baharun N (2019) Automated machine learning based on genetic programming: a case study on a real house pricing dataset. Proceeding on Int Conf on Artificial Intelligence and Data Sciences:48–52
52. Min W, Wynter L (2011) Real-time road traffic prediction with spatio-temporal correlations. Transportation Research Part C: Emerging Technologies 19(4):606–616
53. Mohammad YFO, Matsumoto K, Hoashi K (2018) Deep feature learning and selection for activity recognition. Proceeding on Annual ACM Symposium on Applied Computing:930–939
54. Molchanov P, Tyree S, Karras T, Aila T, Kautz J (2017) Pruning convolutional neural networks for resource efficient transfer learning. Proceeding on Int. Conf. on Learning Representations
55. Moudhgalya NB, Sundar SS, Divi S, Mirunalini P, Aravindan S, Jaisakthi SN (2018) Convolutional Long short-term memory neural networks for hierarchical species prediction. Proceeding on Conf. and Labs of the Evaluation Forum
56. Nah FFH (2003) A study on tolerable waiting time: how Long are web users willing to wait?. Proceeding on Americas Conf. on Information Systems
57. Parashar A, Rhu M, Mukkara A, Puglielli A, Venkatesan R, Khailany B, Emer J, Keckler SW, Dally WJ (2017) Scnn: An accelerator for compressed-sparse convolutional neural networks. Proceeding on Annual Int. Symp. on Computer Architecture, pp. 27–40
58. Peng J (2009) Study on dynamic relation between share price index and housing price: co-integration analysis and application in share price index prediction. Advances in Intelligent and Soft Computing 56:837–846
59. Quost B, Denoeux T (2016) Clustering and classification of fuzzy data using the fuzzy EM algorithm. Fuzzy Sets Syst 286:134–156
60. Roy S, Panda P, Srinivasan G, Raghunathan A (2020) Pruning filters while training for efficiently optimizing deep learning networks. Proceeding on Int Joint Conf on Neural Networks:1–7
61. Sani S, Wiratunga N, Massie S (2017) Learning deep features for knn basedhuman activity recognition. Proceeding on Int Conf on Case-Based Reasoning Workshops:95–103
62. Scardapane S, Comminiello D, Hussain A, Uncini A (2017) Group sparse regularization for deep neural networks. Neurocomputing 241:81–89
63. Shahbazi M, Aghajan H (2018) A generalizable model for seizure prediction based on deep learning using CNN-LSTM architecture. Proceeding on IEEE Global Conf on Signal and Information Processing:469–473
64. X. Shi, Z. Chen, H. Wang, D. Y. Yeung, W. K. Wong, and W. C. Woo (2015) Convolutional LSTM Network: a machine learning approach for precipitation nowcasting," Proceeding on Int. Conf. on Neural Information Processing Systems, pp. 802–810
65. Simard M, Saatchi S, Grandi GD (2000) Use of decision tree and multiscale texture for classification of JERS-1 SAR data over tropical forest. IEEE Trans Geosci Remote Sens 38(5):2310–2321
66. Song L, Qian X, Li H, Chen Y (2017) PipeLayer: a pipelined ReRAM-based accelerator for deep learning. Proceeding on IEEE Int. Symposium on High Performance Computer Architecture
67. Sun J, Di L, Sun Z, Shen Y, Lai Z (2019) County-level soybean yield prediction using deep CNN-LSTM model. Sensors 19(20):4363
68. Tang J, Li L, Hu Z, Liu F (2019) Short-term traffic flow prediction considering Spatio-temporal correlation: a hybrid model combing Type-2 fuzzy C-means and artificial neural network. IEEE Access 7:101009–101018
69. Tian T, Jin X, Zhao L, Wang X, Wang J, Wu W (2020) Exploration of memory access optimization for FPGA-based 3D CNN accelerator. Proceeding on Design, Automation & Test in Europe Conference & Exhibition:1650–1655
70. Titos M, Bueno A, García L, Benítez MC, Ibañez J (2019) Detection and classification of continuous volcano-seismic signals with recurrent neural networks. IEEE Trans Geosci Remote Sens 57(4):1936–1948
71. Tung F, Mori G (2018) CLIP-Q: deep network compression learning by in-parallel pruning- quantization. Proceeding on IEEE/CVF Conf Comput Vis Pattern Recognit:7873–7882
72. Van De Vlag DE, Stein A (2007) Incorporating uncertainty via hierarchical classification using fuzzy decision trees. IEEE Trans Geosci Remote Sens 45(1):237–245
73. Wang X, Zhang Y, Zhang W, Lin X (2016) Distance-aware influence maximization in geo-social network. Proceeding on IEEE Int Conf on Data Engineering:1–12
74. Wang K, Guan D, Li B (2018) Deep group residual convolutional CTC networks for speech recognition. Advanced Data Mining and Applications:318–328
75. Wen W, Wu C, Wang Y, Chen Y, Li H (2016) Learning structured sparsity in deep neural networks. Adv Neural Inf Proces Syst:2074–2082
76. Wess M, Dinakarrao SMP, Jantsch A (2018) Weighted quantization regularization in DNNs for weight memory minimization toward HW implementation. IEEE Trans Comput-Aided Design Integr Circuits Syst 37(11):2929–2939
77. Wu D, Lv S, Jiang M, Song H (2020) Using channel pruning-based YOLO v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments. Comput Electron Agric 178:105742
78. Xia J, Ghamisi P, Yokoya N, Iwasaki A (2018) Random forest ensembles and extended multiextinction profiles for hyperspectral image classification. IEEE Trans Geosci Remote Sens 56(1):202–216
79. Yang D, Yu W, Mu H, Yao G (2021) Dynamic programming assisted quantization approaches for compressing Normal and robust DNN models. Proceeding on Asia and South Pacific Design Automation Conference:351–357
80. Yao S, Zhao Y, Shao H, Liu SZ, Liu D, Su L, Abdelzaher T (2018) FastDeepIoT: towards understanding and optimizing neural network execution time on mobile and embedded devices. Proceeding on Conf on Embedded Networked Sensor Systems: 278–291
81. Yiorkas C, DImopoulos T (2017) Implementing GIS in estate price prediction and mass valuation: the case study of Nicosia District. Proceeding on Int. Conf. on Remote Sensing and Geoinformation of the Environment
82. Yang W, Deng M, Xu F, Wang H (2018) Prediction of hourly PM2.5 using a space-time support vector regression model. Atmos Environ 181:12–19

83. Yue Y, Yeh AGO (2008) Spatiotemporal traffic-flow dependency and short-term traffic forecasting. Environment and Planning B: Planning and Design 35(5):762–771
84. Zainudin Z, Shamsuddin SM, Hasan S (2020) Convolutional neural network Long short-term memory (CNN + LSTM) for histopathology Cancer image classification. Machine Intelligence and Signal Processing:235–245
85. Zhan Y, Luo Y, Deng X, Chen H, Grieneisen ML, Shen X, Zhu L, Zhang M (2017) Spatiotemporal prediction of continuous daily PM2.5 concentrations across China using a spatially explicit machine learning algorithm. Atmos Environ 155:129–139
86. Zhang S, Du Z, Zhang L, Lan H, Liu S, Li L, Guo Q, Chen T, Chen Y (2016) Cambricon-x: an accelerator for sparse neural networks. Proceeding on Annual IEEE/ACM Int Symp on Microarchitecture:1–12
87. Zhang D, Ding W, Zhang B, Xie C, Li H, Liu C, Han J (2018) Automatic modulation classification based on deep learning for unmanned aerial vehicles. Sensors 18
88. Zhang X, Zhang J, Li C, Cheng C, Jiao L, Zhou H (2018) Hybrid unmixing based on adaptive region segmentation for hyperspectral imagery. IEEE Trans Geosci Remote Sens 56(17):3861–3875
89. Zhang S, Cao J, Zhang Q, Zhang Q, Zhang Y, Wang Y (2020) An FPGA-based reconfigurable CNN accelerator for YOLO. Proceeding on IEEE Int Conf on Electronics Technology:74–78
90. Zhang Y, Cheng T, Ren Y (2019) A graph deep learning method for short-term traffic forecasting on large road networks. Computer-Aided Civil and Infrastructure Engineering 34:877–896
91. Zhong Y, Zhu Q, Zhang L (2015) Scene classification based on the multifeature fusion probabilistic topic model for high spatial resolution remote sensing imagery. IEEE Trans Geosci Remote Sens 53(11):6207–6222
92. "DATA Building Open data," http://dbuild.cpami.gov.tw/
93. "Department of commerce, moea, commerce industrial service portal," https://gcis.nat.gov.tw/
94. "Dept of Land Administration M. O. I.," http://lvr.land.moi.gov.tw/
95. "Urban Development Bureau of Taichung City Government," https://www.ud.taichung.gov.tw/

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Sheng-Min Chiu received the B.S. degree in the Department of Information Engineering and Computer Science from Feng Chia University, Taichung, Taiwan, in 2017 and the M.S. degrees in Department of Computer Science and Information Engineering from National Cheng Kung University, Tainan, Taiwan, in 2019. He is currently pursuing the Ph.D. degree in Department of Computer Science and Information Engineering from National Cheng Kung University, Tainan, Taiwan and his main research interests in artificial intelligences, machine learning and spatio-temporal databases.

Yi-Chung Chen received the B.S. and M.S. degrees in Electrical Engineering from National Cheng Kung University, Tainan, Taiwan, in 2007 and 2008, and the Ph.D. degrees in Department of Computer Science and Information Engineering from National Cheng Kung University, Tainan, Taiwan, in 2014. He is currently an Assistant Professor in the Department of Industrial Engineering and Management, National Yunlin University of Science and Technology. His research interests include spatio-temporal databases and artificial intelligences.

Chiang Lee received the BS degree from National Cheng-Kung University, Taiwan, in 1980 and the ME and PhD degrees in electrical engineering from the University of Florida, Gainesville, in 1986 and 1989, respectively. He joined IBM Mid-Hudson Laboratories, Kingston, New York, in 1989. He joined the faculty of National Cheng-Kung University in 1990 and is currently a professor of the Department of Computer Science and Information Engineering. He has published many papers in major journals and conferences, and has been invited as an author of a chapter for several technical books.