A multi-granularity locally optimal prototype-based approach for classification

Gu, Xiaowei; Li, Miqing

DOI:
10.1016/j.ins.2021.04.039

Document Version
Peer reviewed version

Citation for published version (Harvard):
Gu, X & Li, M 2021, 'A multi-granularity locally optimal prototype-based approach for classification', *Information Sciences*, vol. 569, pp. 157-183. https://doi.org/10.1016/j.ins.2021.04.039

A Multi-Granularity Locally Optimal Prototype-Based Approach for Classification

Xiaowei Gu[1], Miqing Li[2]

[1]Department of Computer Science, Institute of Mathematics, Physics and Computer Science, Aberystwyth University, Aberystwyth, SY23 3DB, UK

[2]School of Computer Science, University of Birmingham, Birmingham, B15 2TT, UK

Emails: xig4@aber.ac.uk; m.li.8@cs.bham.ac.uk

Abstract: Prototype-based approaches generally provide better explainability and are widely used for classification. However, the majority of them suffer from system obesity and lack transparency on complex problems. In this paper, a novel classification approach with a multi-layered system structure self-organized from data is proposed. This approach is able to identify local peaks of multi-modal density derived from static data and filter out more representative ones at multiple levels of granularity acting as prototypes. These prototypes are then optimized to their locally optimal positions in the data space and arranged in layers with meaningful dense links in-between to form pyramidal hierarchies based on the respective levels of granularity accordingly. After being primed offline, the constructed classification model is capable of self-developing continuously from streaming data to self-expend its knowledge base. The proposed approach offers higher transparency and is convenient for visualization thanks to the hierarchical nested architecture. Its system identification process is objective, data-driven and free from prior assumptions on data generation model with user- and problem- specific parameters. Its decision-making process follows the "nearest prototype" principle, and is highly explainable and traceable. Numerical examples on a wide range of benchmark problems demonstrate its high performance.

Keywords: local optimality; multi-granularity; prototype-based; pyramidal hierarchy.

1. Introduction

Classification is a hot topic centred in the machine learning and statistic domains. Many classification techniques have been developed and successfully applied in various disciplines of science and technology.

In recent years, deep neural networks (DNNs) have gained enormous popularity among researchers as well as the general public thanks to the state-of-the-art performance they demonstrated on many practical applications [25]. Despite their success, research communities and industries start calling for explainable artificial intelligence [10] due to the increasing concerns on the issues of understandability and trustability of intelligent systems. Compared with DNNs and other mainstream classification algorithms such as decision tree (DT) [27] and random forests [4], prototype-based approaches (e.g., support vector machines (SVMs) [7], k-nearest neighbour (KNN) [8], learning vector quantization (LVQ) [23],[35] and evolving intelligent systems (EISs) [1],[34]) are more popular in the application scenarios where the model interpretability plays an important role. Nonetheless, it is also observed that prototype-based systems learned from high-dimensional, large-scale, complex problems can be over-sized and extremely difficult to interpret [10].

To further enhance the interpretability and explainability of prototype-based systems, a feasible way is to organize the identified prototypes in layers according to their descriptive abilities [19]. In addition, one may need to consider both the objectiveness of the prototype identification process and the local optimality of the learned solutions from data in system design because they both determine the effectiveness and validity of prototype-based approaches. In other words, the learning model should objectively disclose the underlying data patterns while providing users with the currently best-fitted solutions from empirically observed data.

In this paper, a novel multi-granularity locally optimal prototype-based (MLOP) approach with such characteristics is proposed for classification. The proposed approach is capable of building a multi-layered recognition model with locally optimal prototypes representing local peaks of multi-modal density. In the proposed approach, prototypes are firstly identified from data as the most representative samples at multiple levels of specificity. Then, they are optimized iteratively to the locally optimal positions to guarantee both the effectiveness and validity of the learned solutions. After this, these locally optimal prototypes are arranged naturally in a pyramidally hierarchical form according to the respective levels of granularity. In contrast with the

classification approach presented in [19], the MLOP classifier is an approach designed for learning locally optimal prototypes from static data with the capability of self-determining its model structure. The proposed model builds much denser connections between locally optimal prototypes of successive layers based on their spatial scattering, resulting in more robust performance across a wide variety of different problems. Despite of being an offline learning model, the proposed MLOP classifier is also capable of continuously self-learning from streaming data in a recursive manner after being primed with static data. This further allows the proposed approach to quickly self-adapt to new data patterns and makes it suitable for online application scenarios.

Instead of being a "black box" model, the learning process of the proposed MLOP approach is driven by data without involving any prior assumption on data generation model. Any decisions made during the learning and decision-making processes are directly based on empirically observed training data, and thus, rationales behind the decisions can be explained clearly to humans. The learned prototypes can be visualized in a human-understandable form objectively reflecting the underlying data patterns.

Key features of the proposed algorithm are: (1) self-organization of a multi-layered recognition model for classification in a fully autonomous, data-driven manner; (2) self-determination of the system structure based on the ensemble properties and mutual distribution of data; (3) maximization of the information mined from data by iteratively optimizing the obtained solutions; (4) perception of complex problems with multiple levels of specificity simultaneously.

The reminder of this paper is organized as follows. Section 2 provides a review of related works. The algorithmic procedure of the MLOP classifier is described in detail in Section 3. Computational complexity of the proposed approach is analysed in Section 4. Numerical examples are provided in Section 5 as the proof of the concept. Section 6 concludes this paper and points out directions for future work.

2. Related Works

Currently, there have been a wide variety of successful classification approaches developed. Due to the limited space of this paper, it is practically impossible to cover all of them. The review of related works in this paper is focused on mainstream approaches of the two most relevant categories, namely, (1) multi-layered and (2) prototype-based.

DNNs (or artificial neural networks, ANNs) are currently the best-known multi-layered approaches for classification. They have achieved great success in many complex recognition tasks involving visual and speech information [24],[28],[38], which leads to the recent hot wave of deep learning [25]. Although DNNs are very powerful, they suffer from several deficiencies as follows [17],[20],[47]. Firstly, it is well known that the training process of DNNs is data- and computational resource- hungry. Without a huge amount of labelled training data and powerful computational facilities, it would be very difficult for individuals to fully exploit the learning ability and build a well-performing model. Secondly, DNNs are highly complicated models with typically millions of hyper-parameters. Their performance depends heavily on careful tuning, and their training and decision-making processes lack transparency and are not human-interpretable because of too many interfering factors with almost infinite configurational combinations [49]. Thirdly, the performance of DNNs is fragile to new observations with unfamiliar patterns, and DNNs can be easily fooled to produce high confidence predictions for images that are unrecognizable to humans [29].

Recent researches have demonstrated that both transfer learning and semi-supervised learning can substantially reduce the amount of labelled data needed for DNN training. To be more specific, transfer learning [50] aims to train a DNN to solve new problems better and faster using limited labelled training data by utilizing the previously learned knowledge from different but related problems. Meanwhile, semi-supervised learning [3],[42] attempts to build a strong recognition model by involving a great amount of unlabelled data with a limited amount of labelled ones together. Both approaches have demonstrated great success in addressing the data-hunger issue, but other issues inherent in DNNs, such as lack of transparency and explainability, remain open.

There are some alternative multi-layered learning models introduced recently attempting to achieve high-level performance competitive to DNNs but with less aforementioned deficiencies. For example, a deep forest framework was proposed in [49] by constructing a multi-layered model using random forests as its building blocks. Essentially, this approach employs a cascade structure with each level formed by an ensemble of random forests. Input information is processed level-by-level in the deep forest model resembling DNNs. A similar multi-layered model with gradient boosting DTs as base units was presented in [9]. Nonetheless, both models are complicated,

and their transparency and explainability might still be limited depending on the nature of problem. In addition, they both are limited to numerical and simple image classification problems.

By integrating a zero-order EIS [1],[18] with a multi-layer image-processing architecture, deep rule-based (DRB) classifiers proposed in [17] serve as a strong alternative to DNNs for image classification problems by offering both human-level precision and high model transparency. Same as conventional zero-order EISs [1],[18], DRB classifiers are based on prototypes. Prototypes are the most representative samples in the data space. They play an instrumental role in prototype-based systems by summarizing the empirically observed data and preserving the data structure and class distribution [2],[6],[16],[37]. Compared with DNNs, prototype-based systems have the advantages of intuitive model understanding and sparse representation [12]. They are more popular in the application scenarios where model interpretability plays an important role. On the other hand, the performance, efficiency, system transparency and interpretability of prototype-based systems may vary a lot due to the differences in the computational processes for prototype identification.

KNN classifier is one of the most used and powerful prototype-based classifiers [8]. KNN treats all the training samples as prototypes and uses them to classify unlabelled samples by the "nearest neighbours" principle. However, this simple operating mechanism also has several weaknesses, such as higher storage requirement, lower tolerance to noise, lower computational efficiency for decision-making and lower system interpretability [12]. SVM is another most used prototype-based classifier. SVM performs classification based on support vectors (namely, prototypes) obtained from data by identifying the maximum-margin hyperplanes in the data space through an iterative computational process. Compared with KNN, SVM is far more sophisticated and is considered as a typical type of "black box" models. Learning vector quantization (LVQ) [23] and generalized learning vector quantization (GLVQ) [35] are also popular prototype-based systems for classification. Both algorithms iteratively update a predefined number of prototypes within the data space searching for locally optimal solutions based on the principle of competitive learning. However, LVQ and GLVQ are ANNs and their learning processes are opaque due to the iterative parameter optimization. In contrast, zero-order EISs are popular for streaming data classification thanks to their higher transparency, computational efficiency and human-interpretability [39], but they, including the aforementioned DRB classifiers, suffer from the problem of system obesity when applied to large-scale complex problems [20]. In such cases, the computational efficiency and system interpretability of zero-order EISs can be significantly reduced.

There are a few recently proposed prototype-based approaches worth mentioning. For example, a selective prototype-based learning (SPL) algorithm is proposed in [6] for nonstationary streaming data classification. SPL learns a set of highly representative samples from streaming data as prototypes for classification and simultaneously maintains a separate set of misclassified samples for concept drift detection. However, despite that SPL has better capability of handling uncertainties in data streams, its model size as well as a few other parameters have to be fixed a priori by users. As a result, the performance of SPL is very much depending on the externally controlled parameter setting. An ensemble prototype selection approach is presented in [5] for selecting a set of prototypes from training data to achieve the maximum classification accuracy rate by following the "nearest neighbours" principle. This approach considers not only the frequencies of data samples that are selected as prototypes by the ensemble prototype selectors and but also the relationships between these selected prototypes. Nevertheless, this approach would select a very large number of data samples as prototypes from large-scale, high-dimensional datasets, making the constructed classification model uninterpretable.

To address the system obesity problem of prototype-based systems and improve the model transparency, one possible solution is to further aggregate the identified prototypes into a smaller number of more descriptive ones and organize them into pyramidal hierarchies according to their descriptive abilities. An example of this is the two-level approach for streaming data classification, named SyncStream [37]. SyncStream dynamically maintains a two-level data structure with the first level storing raw prototypes representing the current data pattern and the second level storing highly summarized prototypes representing historical data patterns. Raw prototypes at the first level of SyncStream are directly extracted from data and are updated all the time to better capture the current data pattern. Once a new data pattern is detected, these raw prototypes that represent the previous data pattern will be clustered into more descriptive ones and inserted into the second level, and the first level will then be occupied by new raw prototypes. The main issue with SyncStream is that its model size is not self-adjustable from data but has to be predetermined by users based on the prior knowledge. If the scale of prototype exceeds the pre-set maximum numbers, parts of them have to be either discarded or merged together to save spaces for new prototypes. This places a restriction on its applicability in real world applications concerning large-scale data streams with

complex structure. In contrast, the hierarchical prototype-based (HP) classifier proposed in [19] is very suitable for solving such problems with very high prediction precision and computational efficiency. The HP classifier is capable of self-organizing a pyramidical structure composed of meaningful prototypes identified at multiple levels of granularity from streaming data and continuously self-evolving to capture the new data patterns. Nevertheless, prototypes identified by the HP classifier lacks optimality due to its "one pass" learning mechanism, which may adversely influence its prediction precision. Another issue with the HP approach is that its model depth in terms of layer number has to be predetermined by users, which may have a great impact on both the precision and computational efficiency of the learning model. In addition, the HP classifier will require a full re-training if extra layers are added into the system structure.

3. Proposed Approach

In this section, the general architecture and computational process of the proposed approach are presented in detail.

Table 1. A summary of key notations and the respective definitions

Notations	Definitions
$\mathbf{R}^N$	Real metric space
N	Dimensionality of $\mathbf{R}^N$
$\{\boldsymbol{x}\}_K$	Dataset
K	Cardinality of $\{\boldsymbol{x}\}_K$
$\boldsymbol{x}_k$	The data sample observed at the kth time instance
C	Number of classes
$\{\boldsymbol{x}\}^i_{K^i}$	Subset of $\{\boldsymbol{x}\}_K$ belonging to the ith class
K^i	Cardinality of $\{\boldsymbol{x}\}^i_{K^i}$
$\boldsymbol{\mu}^i_{K^i}$	Mean of $\{\boldsymbol{x}\}^i_{K^i}$
$X^i_{K^i}$	Mean of $\{\|\boldsymbol{x}\|^2\}^i_{K^i}$
$\{\boldsymbol{u}\}^i_{L^i}$	Set of unique data samples of the ith class
$\{f\}^i_{L^i}$	Occurrence frequencies of $\{\boldsymbol{u}\}^i_{L^i}$
L^i	Cardinality of $\{\boldsymbol{u}\}^i_{L^i}$
$\boldsymbol{u}^i_k$	The kth unique data sample of the ith class
f^i_k	Occurrence frequency of $\boldsymbol{u}^i_k$
$D^{MM}(\boldsymbol{x})$	Multimodal density of $\boldsymbol{x}$
H^i	Layer number of the ith prototype-based hierarchy
M^i_h	Number of prototypes at the hth layer of the ith prototype-based hierarchy
$\{\boldsymbol{u}\}^{i*}$	Collection of local maxima of the ith class
$\boldsymbol{u}^{i*}_k$	The kth local maximum of the ith class
$\{\mathbf{C}\}^i$	Clusters formed around local maxima of the ith class
$\mathbf{C}^i_k$	Cluster formed around $\boldsymbol{u}^{i*}_k$
S^i_k	Cardinality of $\mathbf{C}^i_k$
$\boldsymbol{q}^i_k$	Centre of $\mathbf{C}^i_k$
$\mathbf{Q}^i_{h,k}$	Collection of neighbouring cluster centres around $\boldsymbol{q}^i_k$ at the hth level of granularity
γ^i_h	Average radius of area of influence around each prototype at the hth layer of the ith prototype-based hierarchy
$\mathbf{P}^i_h$	Collection of prototypes at the hth layer of the ith prototype-based hierarchy
$\boldsymbol{p}^i_{h,k}$	The kth prototypes at the hth layer of the ith prototype-based hierarchy
$S^i_{h,k}$	Number of data samples associated with $\boldsymbol{p}^i_{h,k}$
$\boldsymbol{\mathcal{L}}^i_{h,k}$	Collection of subordinates of $\boldsymbol{p}^i_{h,k}$
$\lambda^i(\boldsymbol{x})$	Score of confidence on $\boldsymbol{x}$ given by the ith prototype-based hierarchy

First of all, let $\{\boldsymbol{x}\}_K = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_k, \ldots, \boldsymbol{x}_K\}$ ($\boldsymbol{x}_k = \left[x_{k,1}, x_{k,2}, \ldots, x_{k,N}\right]^T \in \mathbf{R}^N$) be a particular dataset in a real metric space, $\mathbf{R}^N$ with the dimensionality of N; K is the cardinality of $\{\boldsymbol{x}\}_K$; the subscript k indicates the time instance at which $\boldsymbol{x}_k$ is observed. It is assumed that $\{\boldsymbol{x}\}_K$ is composed of data samples of C different classes. Thus, $\{\boldsymbol{x}\}_K$ can be divided into C subsets, denoted by $\{\boldsymbol{x}\}_{K^i}^i = \{\boldsymbol{x}_1^i, \boldsymbol{x}_2^i, \ldots, \boldsymbol{x}_{K^i}^i\}$ ($i = 1,2,\ldots,C$), based on the corresponding class labels, where the superscript i denotes the ith class and there is $\sum_{i=1}^{C} K^i = K$. For each subset $\{\boldsymbol{x}\}_{K^i}^i$, some samples may share the same values, for example, $\boldsymbol{x}_m^i = \boldsymbol{x}_n^i$ and $m \neq n$. The set of unique data samples of the ith class is denoted as $\{\boldsymbol{u}\}_{L^i}^i = \{\boldsymbol{u}_1^i, \boldsymbol{u}_2^i, \ldots, \boldsymbol{u}_{L^i}^i\}$ ($\{\boldsymbol{u}\}_{L^i}^i \subseteq \{\boldsymbol{x}\}_{K^i}^i$), and the corresponding occurrence frequencies are denoted as $\{f\}_{L^i}^i = \{f_1^i, f_2^i, \ldots, f_{L^i}^i\}$, where f_k^i is the occurrence frequency of $\boldsymbol{u}_k^i$; L^i is the cardinality of $\{\boldsymbol{u}\}_{L^i}^i$; $\sum_{k=1}^{L^i} f_k^i = K^i$. Without loss of generality, in this paper, Euclidean distance is used for derivation by default. For clarity, key notations and the respective definitions used in this paper are summarized in Table 1.

3.1. General architecture

The general architecture of the MLOP classifier is given in Fig. 1, where one can see that the proposed approach consists of C different prototype-based hierarchies (one hierarchy per class). In Fig. 1, $\boldsymbol{p}_{h,k}^i$ denotes the kth prototype at the hth layer of the ith hierarchy; $h = 1,2,\ldots,H^i$; $k = 1,2,\ldots,M_h^i$; H^i is the layer number, which would be different for each hierarchy and is determined by data; M_h^i is the total number of prototypes at the hth layer of the ith hierarchy; $i = 1,2,\ldots,C$.

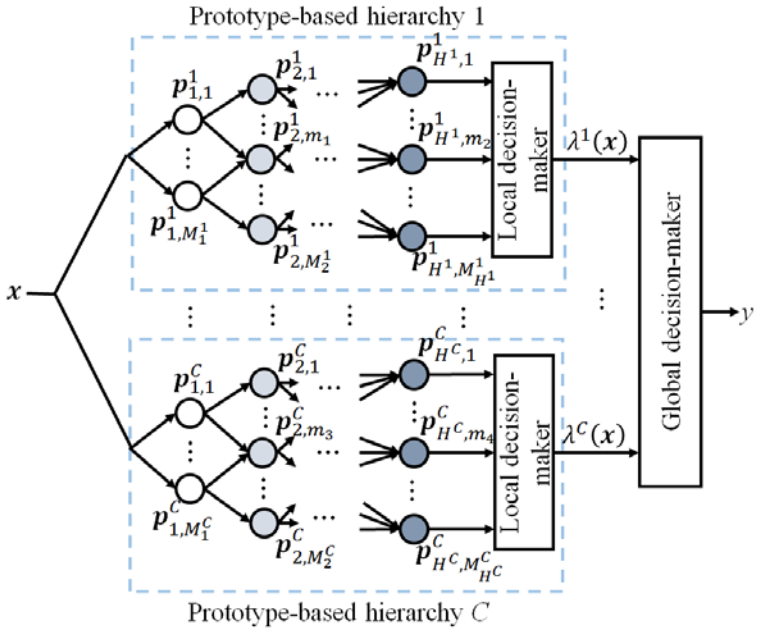


Fig. 1. General architecture of the proposed approach ($1 \leq m_1 \leq M_2^1$; $1 \leq m_2 \leq M_{H^1}^1$; $1 \leq m_3 \leq M_2^C$; and $1 \leq m_4 \leq M_{H^C}^C$)

Each hierarchy is composed of meaningful prototypes identified from data samples of the corresponding class at multiple levels of granularity from low to high. Each prototype (except for leaf prototypes at the bottom layer) is connected to one or multiple subordinate prototypes at the next layer. At the same time, each prototype (except for apex prototypes at the top layer) is linked with one or more superior prototypes at the layer above. However, unlike the artificial neurons of adjacent layers in neural networks that are fully connected, links between prototypes at the successive layers of the hierarchies within the MLOP classifier are established only when they are physically neighbouring in the data space. Note that there is no connection between prototypes at the same layer or prototypes of different hierarchies.

Very importantly, the prototype-based hierarchies can be easily visualized in an easy-to-interpret form, allowing users to perceive a problem at multiple levels of granularity. The top layers of the hierarchies usually have only a

very small amount of highly descriptive prototypes representing global patterns of data, which can help users to quickly capture the big picture. Meanwhile, the lower layers may have a larger number of prototypes, which are closer to the raw data samples. These prototypes represent the local patterns of data and can provide lots of fine details, but users may have to spend more times to interpret them. In addition, the links between prototypes of successive layers provide users very important information regarding the relationships between these global and local patterns, which can help users to gain a better understanding of the problem.

In the next three subsections, the learning and decision-making processes of the MLOP classifier are described in detail.

3.2. Learning process from static data

In this subsection, the algorithmic procedure for the proposed approach to self-organize a hierarchical structure from static data is described in detail. Since the learning process of the proposed approach is performed class-wise, the *i*th hierarchy is used as an example for illustration ($i = 1,2,\dots,C$). The same principles can be applied to all other hierarchies within the MLOP classifier.

Stage 1. Voronoi tessellation formation [18]

In this stage, the multimodal density value at each unique data sample, $\boldsymbol{u}_k^i$ ($\boldsymbol{u}_k^i \in \{\boldsymbol{u}\}_{L^i}^i$) of the *i*th class is firstly calculated using equation (1) [2]:

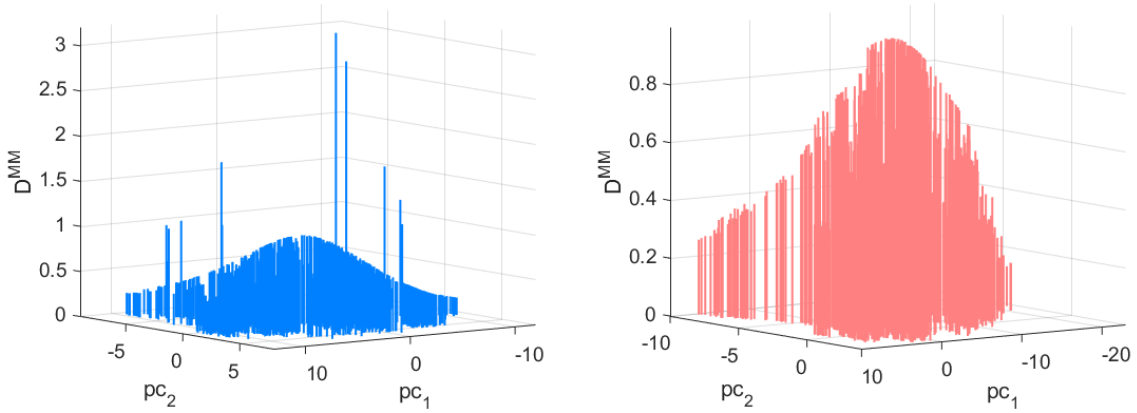
$$D^{MM}(\boldsymbol{u}_k^i) = f_k^i \frac{1}{1+\frac{\left\|\boldsymbol{u}_k^i - \boldsymbol{\mu}_{K^i}^i\right\|^2}{X_{K^i}^i - \left\|\boldsymbol{\mu}_{K^i}^i\right\|^2}} \tag{1}$$

where $\boldsymbol{\mu}_{K^i}^i$ and $X_{K^i}^i$ are the respective means of $\{\boldsymbol{x}\}_{K^i}^i$ and $\{\|\boldsymbol{x}\|^2\}_{K^i}^i$, which can be calculated by following expression:

$$\boldsymbol{\mu}_{K^i}^i = \frac{1}{K^i}\sum_{k=1}^{K^i} \boldsymbol{x}_k^i; \quad X_{K^i}^i = \frac{1}{K^i}\sum_{k=1}^{K^i} \left\|\boldsymbol{x}_k^i\right\|^2 \tag{2}$$

and $\|\boldsymbol{x}_k\| = \sqrt{\sum_{j=1}^{N} x_{k,j}^2}$ denotes the Euclidean norm of $\boldsymbol{x}_k$.

Multimodal density behaves like the multimodal discrete probability density function by considering both occurrence frequencies and mutual distances of data. It also has multiple local peaks representing the local models of data distribution [2],[16]. Data samples with locally maximum multimodal density values can better represent the local models of data distribution and thus, are used as prototypes for classification.



(a) Class 1 (b) Class 2

Fig. 2. Multimodal density obtained from the BA dataset

To better deliver the concept, the banknote authentication (BA) dataset[1] is employed for visual illustration. BA dataset is a binary classification problem very suitable for visualization thanks to its smaller scale and simpler structure (2 classes with 1372 data samples in total; each sample has 4 attributes). Here, principal component analysis (PCA) is applied to further reduce the dimensionality of data to two for visual clarity. The multimodal density values calculated at data samples of classes 1 and 2 using equation (1) are depicted in Figs. 2(a) and 2(b), respectively.

To identify the local peaks of multimodal density, firstly, all the unique data samples, $\{\boldsymbol{u}\}_{L^i}^i$ are arranged in an indexing list, denoted by $\{\boldsymbol{r}\}^i$, with regard to their mutual distances and ensemble properties. The unique data sample with the highest multimodal density value is selected as the first element, $\boldsymbol{r}_1^i$ of $\{\boldsymbol{r}\}^i$ [16]:

$$\boldsymbol{r}_1^i = argmax_{\boldsymbol{u}_k^i \in \{\boldsymbol{u}\}_{L^i}^i}\left(D^{MM}(\boldsymbol{u}_k^i)\right); \quad \{\boldsymbol{u}\}_{L^i}^i \leftarrow \{\boldsymbol{u}\}_{L^i}^i / \{\boldsymbol{r}_1^i\} \tag{3}$$

Then, remaining elements of $\{\boldsymbol{r}\}^i$ are identified one-by-one based on the following rule ($k = 2,3,\dots,L^i$) [16]:

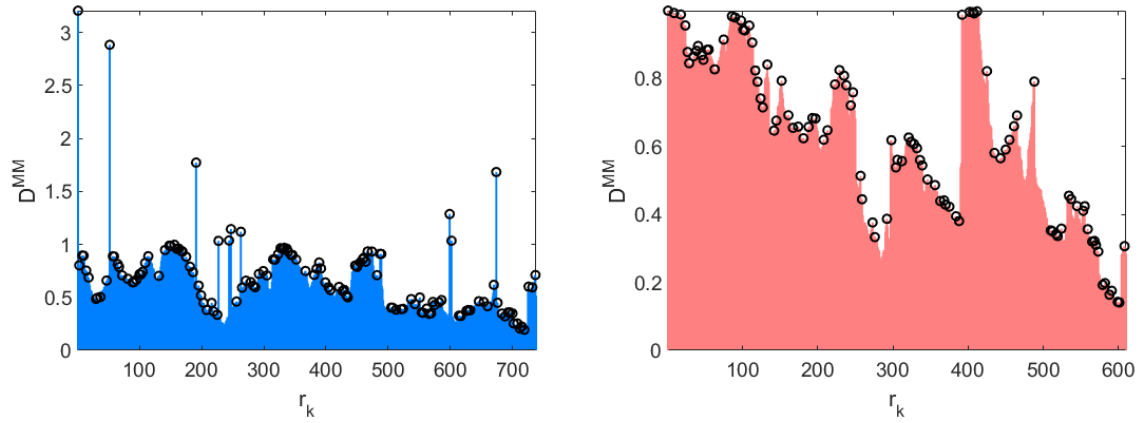
$$\boldsymbol{r}_k^i = argmin_{\boldsymbol{u} \in \{\boldsymbol{u}\}_{L^i}^i}\left(\|\boldsymbol{r}_{k-1}^i - \boldsymbol{u}\|\right); \quad \{\boldsymbol{u}\}_{L^i}^i \leftarrow \{\boldsymbol{u}\}_{L^i}^i / \{\boldsymbol{r}_k^i\} \tag{4}$$

Once the full indexing list is built, local maxima, denoted by $\{\boldsymbol{u}\}^{i*}(\{\boldsymbol{u}\}^{i*} \leftarrow \{\boldsymbol{r}_k^1\})$, of multimodal density can be identified by **Condition 1** ($k = 2,3,\dots,L^i$-1) [16],[18]:

$$\textbf{\textit{Condition 1}}: \quad \begin{matrix} If \ \left(sgn\left(D^{MM}(\boldsymbol{r}_k^i) - D^{MM}(\boldsymbol{r}_{k-1}^i)\right) = sgn\left(D^{MM}(\boldsymbol{r}_k^i) - D^{MM}(\boldsymbol{r}_{k+1}^i)\right) = 1\right) \\ Then \ \left(\{\boldsymbol{u}\}^{i*} \leftarrow \{\boldsymbol{u}\}^{i*} \cup \{\boldsymbol{r}_k^i\}\right) \end{matrix} \tag{5}$$

where $sgn(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$ is the sign function; the cardinality of $\{\boldsymbol{u}\}^{i*}$ is L^{i*}.

Continuing the example in Fig. 2, the ranked multimodal density values in regard to the indexing list obtained by equations (3) and (4) are given in Fig. 3, where the identified local maxima by **Condition 1** are marked by black circles, "○". In addition, the positions of local maxima in the data space are given in Fig. 4, where dots "•" represent data samples.



(a) Class 1 (b) Class 2

Fig. 3. Ranked multimodal density and local maxima identified using **Condition 1** (local maxima are marked by black circles, "○")

[1] Available at: https://archive.ics.uci.edu/ml/datasets/banknote+authentication.

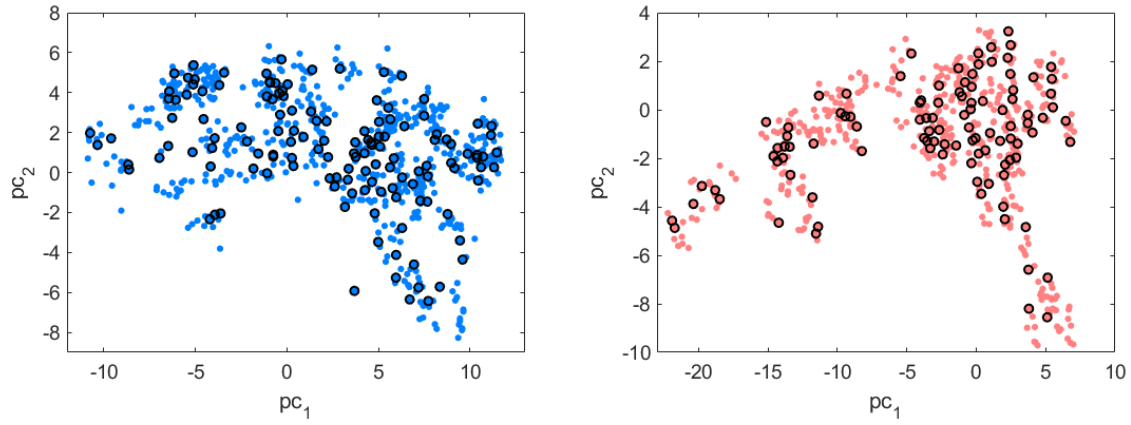
After the local maxima, $\{\boldsymbol{u}\}^{i*}$ have been identified, Voronoi tessellations are formed in the data space by using them as raw prototypes to attract nearby data samples creating a number of clusters, denoted by $\{\mathbf{C}\}^i$:

$$\mathbf{C}_{n_*}^i \leftarrow \mathbf{C}_{n_*}^i \cup \{\boldsymbol{x}_k^i\}; \quad n_* \leftarrow argmin_{j=1,2,\dots,L^{i*}}\left(\left\|\boldsymbol{u}_j^{i*} - \boldsymbol{x}_k^i\right\|\right) \tag{6}$$

where $k = 1,2,\dots,K^i$. Then, multimodal density values at the centres of the clusters, $\{\mathbf{C}\}^i$ are calculated by the following equation ($k = 1,2,\dots,L^{i*}$) [2]:

$$D^{MM}\left(\boldsymbol{q}_k^i\right) = S_k^i \frac{1}{1+\frac{\left\|q_k^i - \mu_{K^i}^i\right\|^2}{x_{K^i}^i - \left\|\mu_{K^i}^i\right\|^2}} \tag{7}$$

where S_k^i is the cardinality of $\mathbf{C}_k^i$; $\boldsymbol{q}_k^i$ is the centre of $\mathbf{C}_k^i$ and there is $\boldsymbol{q}_k^i = \frac{1}{S_k^i}\sum_{\boldsymbol{x}\in\mathbf{C}_k^i}\boldsymbol{x}$. After this, the learning algorithm enters the next stage.



(a) Class 1 (b) Class 2

Fig. 4. Identified local maxima in the data space (dots "•" represent data samples; local maxima are marked by black circles, "○")

Stage 2. Multi-granularity prototype identification

As the local maxima identified by **Condition 1** may contain some less representative ones, in this stage, these local maxima are filtered based on their multimodal density values and mutual distances to extract the most representative ones as prototypes at different levels of granularity [2],[16],[18]. The identification process starts from the first level, namely, $h = 1$.

To extract prototypes at the *h*th level of granularity, the neighbouring clusters of each cluster need to be identified in the first place using the following condition based on the mutual distances of cluster centres ($k, j = 1,2,\dots,L^{i*}; k \neq j$) [18]:

$$\textbf{Condition 2}: \quad \begin{array}{c} If\ \left(\left\|\boldsymbol{q}_k^i - \boldsymbol{q}_j^i\right\|^2 \leq \gamma_h^i\right) \\ Then\ \left(\mathbf{Q}_{h,k}^i \leftarrow \mathbf{Q}_{h,k}^i \cup \{\boldsymbol{q}_j^i\}\right) \end{array} \tag{8}$$

where $\mathbf{Q}_{h,k}^i$ denotes the collection of neighbouring cluster centres surrounding $\boldsymbol{q}_k^i$ at the *h*th level of granularity; γ_h^i is the corresponding average radius of area of influence around each prototype and is derived by equation (9) ($h = 1,2,3,\dots$) [18]:

$$\gamma_h^i = \frac{1}{N_h^i}\sum_{\boldsymbol{x},\boldsymbol{y}\in\{\boldsymbol{x}\}_{K^i}^i;\boldsymbol{x}\neq\boldsymbol{y};\|\boldsymbol{x}-\boldsymbol{y}\|^2\leq\gamma_{h-1}^i}\|\boldsymbol{x}-\boldsymbol{y}\|^2 \tag{9}$$

here $\gamma_0^i = 2\left(X_{K^i}^i - \left\|\boldsymbol{\mu}_{K^i}^i\right\|^2\right)$, which is the average distance between any two data samples of the *i*th class; N_h^i is the number of pairs of data samples within $\{\boldsymbol{x}\}_{K^i}^i$ between which the distance is smaller than γ_{h-1}^i. γ_h^i provides an intuitive estimation of the average distance between any two strongly connected prototypes at the *h*th level of granularity by condensing the mutual distribution information extracted from data. Thus, γ_h^i is guaranteed to be valid all the time. Thus, **Condition 2** and equation (9) together define the concept of closeness at multiple levels of granularity in a meaningful, data-driven way.

Then, the following principle is used for identifying prototypes as the most representative local maxima ($k = 1,2,\ldots,L^{i*}$) [18]:

$$\textbf{Condition 3}: \quad \begin{array}{c} If\ \left(D^{MM}(\boldsymbol{q}_k^i) > max_{\boldsymbol{q}\in\mathbf{Q}_{h,k}^i}\left(D^{MM}(\boldsymbol{q})\right)\right) \\ Then\ \left(\mathbf{P}_h^i \leftarrow \mathbf{P}_h^i \cup \{\boldsymbol{q}_k^i\}\right) \end{array} \tag{10}$$

where $\mathbf{P}_h^i$ denotes the collection of prototypes of the *i*th class identified at the *h*th level of granularity; M_h^i is the cardinality of $\mathbf{P}_h^i$. After the prototypes have been extracted, the algorithm enters the next stage to optimize them to their locally optimal positions in the data space.

Stage 3. Prototype optimization

The local optimality of the obtained prototypes plays a critical role in determining the overall performance of the proposed classifier due to its prototype-based nature [18]. Therefore, in this stage, the algorithm optimizes the solution obtained in **Stage 2**, namely, $\mathbf{P}_h^i$ by iteratively minimizing the following objective function [36]:

$$J_1\left(\mathbf{P}_h^i\right) = \frac{\sum_{k=1}^{K^i}\sum_{j=1}^{M_h^i} w_{j,k}\left\|\boldsymbol{p}_{h,j}^i - \boldsymbol{x}_k^i\right\|^2}{K^i\gamma_0^i} \tag{11}$$

$$\text{where } w_{j,k} = \begin{cases} 1, & if\ \left\|\boldsymbol{p}_{h,j}^i - \boldsymbol{x}_k^i\right\|^2 = min_{t=1,2,\ldots,M_h^i}\left(\left\|\boldsymbol{p}_{h,t}^i - \boldsymbol{x}_k^i\right\|^2\right) \\ 0, & else \end{cases} \tag{12}$$

Essentially, equation (11) calculates the intra-cluster variance of the partitioning results. To minimize $J_1\left(\mathbf{P}_h^i\right)$, the following two steps are repeated until $J_1\left(\mathbf{P}_h^i\right)$ converges to the (locally) minimum value [18],[40]:

Step 1. Create Voronoi tessellations in the data space by using $\mathbf{P}_h^i$ to attract nearby data samples and form new clusters (namely, equation (6)).

Step 2. Update $\mathbf{P}_h^i$ as the centres of the newly formed clusters and recalculate the objective function, $J_1\left(\mathbf{P}_h^i\right)$ (namely, equation (11)).

Once $J_1\left(\mathbf{P}_h^i\right)$ has converges to the (locally) minimum value, the optimization process is completed and $\mathbf{P}_h^i$ have been adjusted to the locally optimal positions. Then, the algorithm enters the next stage.

Stage 4. Stopping criterion inspection

In this stage, the learning algorithm uses the following objective function (equation (13)) to assess whether the prototypes identified at the *h*th level of granularity, namely, $\mathbf{P}_h^i$ have sufficiently partitioned the data space and disclosed fine details of the underlying data patterns to build a well-performing recognition model altogether with $\mathbf{P}_1^i,\ \mathbf{P}_2^i,\ldots,\ \mathbf{P}_{h-1}^i$.

$$J_2\left(\mathbf{P}_h^i\right) = J_1\left(\mathbf{P}_h^i\right) + \rho\frac{M_h^i}{K^i} = \frac{\sum_{k=1}^{K^i}\sum_{j=1}^{M_h^i} w_{j,k}\left\|\boldsymbol{p}_{h,j}^i - \boldsymbol{x}_k^i\right\|^2}{K^i\gamma_0^i} + \rho\frac{M_h^i}{K^i} \tag{13}$$

where ρ is the regularization parameter ($\rho \geq 0$); $\frac{M_h^i}{K^i}$ is the penalty in terms of number of prototypes. ρ controls the trade-off between the intra-cluster variance and the number of prototypes (layers).

Condition 4 is used for the MLOP classifier to automatically self-determine the most appropriate level of granularity:

$$\textbf{Condition 4}: \quad \begin{aligned} &If\ \left(sgn\left(J_2(\mathbf{P}_h^i) - J_2(\mathbf{P}_{h-1}^i)\right) = sgn\left(J_2(\mathbf{P}_h^i) - J_2(\mathbf{P}_{h+1}^i)\right) = 1\right) \\ &Or\ \left(sgn\left(J_2(\mathbf{P}_h^i) - J_2(\mathbf{P}_{h+1}^i)\right) = 0\right) \\ &Then\ (The\ hth\ level\ of\ granularity\ is\ sufficient\ for\ recognition) \end{aligned} \tag{14}$$

Condition 4 is based on the elbow method [22], which is the oldest method for determining the number of clusters in the dataset. The elbow method treats the intra-cluster variance of the partitioning results as a function of the number of clusters. The appropriate number of clusters is determined when adding extra clusters does not significantly reduce the intra-cluster variance of the results. Nonetheless, the elbow method requires visual inspection by human experts and is sometimes ambiguous. **Condition 4**, on the other hand, replaces the human inspection process by introducing a penalty term based on the number of prototypes at the corresponding level of the hierarchical prototype-based structure.

If **Condition 4** is satisfied, the algorithm proceeds to **Stage 5** to form the hierarchical prototype-based structure with the identified prototypes $\mathbf{P}_1^i$, $\mathbf{P}_2^i$,…, $\mathbf{P}_{h-1}^i$ and $\mathbf{P}_h^i$. Otherwise, the algorithm goes back to **Stage 2** to extract prototypes from data at a higher level of granularity ($h \leftarrow h + 1$).

An illustrative example is provided in Fig. 5 showing how the value of ρ determines the appropriate level of granularity for the proposed MLOP classifier by **Condition 4**, where the blue curve is the relationship between the value of the objective function $J_2(\mathbf{P}_h^i)$ and the layer number/level of granularity, h; black diamonds "◊" are the knee points on the curves where **Condition 4** is satisfied given a specific ρ. In general, it can be observed from Fig. 5, the MLOP classifier tends to self-organize hierarchies with more layers given a smaller ρ.

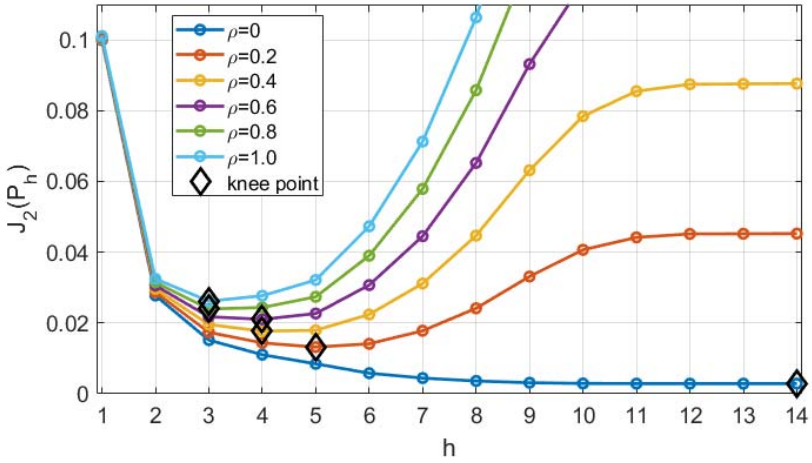
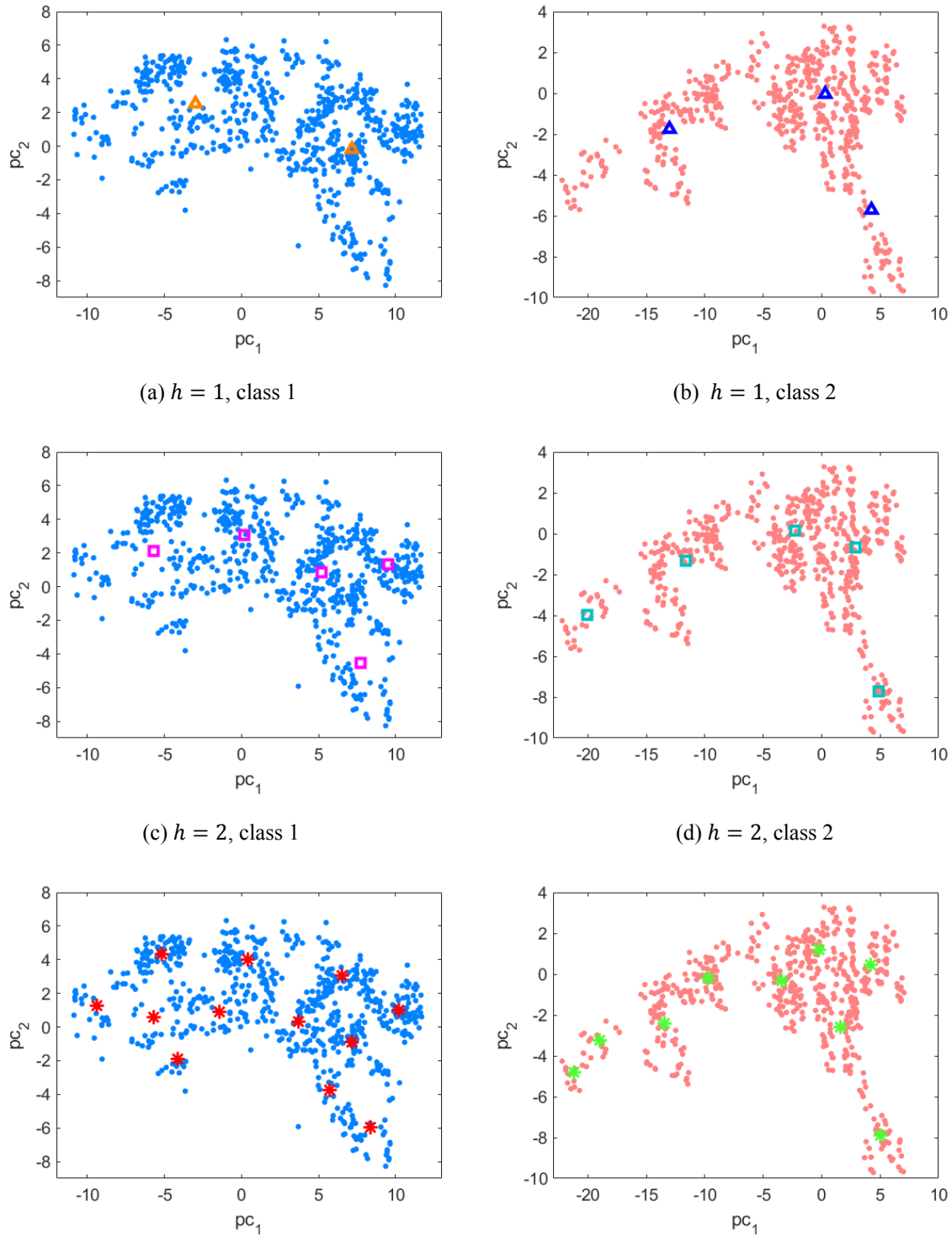


Fig. 5. Appropriate layer number, h determined by **Condition 4** given different values of ρ

Following the visual examples given by Figs. 2-4, **Condition 4** is satisfied for data samples of both classes with $h = 3$ given $\rho = 0.9$. In total, there are two, five and 12 prototypes identified from data samples of class 1 at the first, second and third levels of granularity by **Conditions 2 and 3**, respectively. Meanwhile, there are three, five and nine prototypes identified from data samples of class 2 correspondingly at the respective three different levels of granularity. The obtained locally optimal prototypes by the algorithm at the three levels of granularity from the BA dataset are given in Fig. 6, where asterisks "*" represent the prototypes identified at the first level of granularity; squares "□" represent the prototypes of the second level; triangles "Δ" represent prototypes of the third level.



(a) $h = 1$, class 1

(b) $h = 1$, class 2

(c) $h = 2$, class 1

(d) $h = 2$, class 2

(e) $h = 3$, class 1

(b) $h = 3$, class 2

Fig. 6. Locally optimal prototypes obtained by **Conditions 2 and 3** at different levels of granularity (asterisks "*" represent the prototypes identified at the first level of granularity; squares "□" represent the prototypes at the second level of granularity; triangles "Δ" represent prototypes at the third level of granularity)

Stage 5. Prototype-based hierarchy assembly

The final stage of the algorithm starts by aggregating prototypes in a multi-layered hierarchical structure based on their corresponding levels of granularity. Assuming that **Condition 4** is satisfied by prototypes identified at the H^ith level of granularity, the algorithm will build a H^i-layer hierarchy with $\mathbf{P}_1^i$ as the first layer prototypes, $\mathbf{P}_2^i$ as the second layer prototypes, $\mathbf{P}_3^i$ as the third layer prototypes, etc.

Then, the links (superior-subordinate relationships) between prototypes at successive layers are established based on **Conditions 5a** and **5b**:

$$\textbf{\textit{Condition}}\ \textbf{5a}: \quad \begin{array}{c} If\ \left(\left\|\boldsymbol{p}_{h,j}^i - \boldsymbol{p}_{h-1,k}^i\right\|^2 = min_{t=1,2,\dots M_{h-1}^i}\left(\left\|\boldsymbol{p}_{h,j}^i - \boldsymbol{p}_{h-1,t}^i\right\|^2\right)\right) \\ Then\left(\boldsymbol{\mathcal{L}}_{h-1,k}^i \leftarrow \boldsymbol{\mathcal{L}}_{h-1,k}^i \cup \{\boldsymbol{p}_{h,j}^i\}\right) \end{array} \tag{15}$$

$$\textbf{\textit{Condition}}\ \textbf{5b}: \quad \begin{array}{c} If\ \left(\left\|\boldsymbol{p}_{h,t}^i - \boldsymbol{p}_{h,j}^i\right\|^2 < \alpha_o \cdot \gamma_h^i\right) \\ Then\left(\boldsymbol{\mathcal{L}}_{h-1,k}^i \leftarrow \boldsymbol{\mathcal{L}}_{h-1,k}^i \cup \{\boldsymbol{p}_{h,t}^i\}\right) \end{array} \tag{16}$$

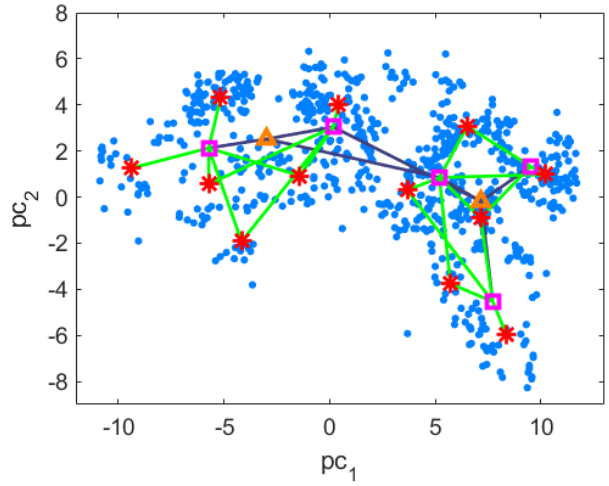
where $\boldsymbol{\mathcal{L}}_{h-1,k}^i$ denotes the collection of subordinates of $\boldsymbol{p}_{h-1,k}^i$; $h = 2,3,\dots,H^i$; $k = 1,2,\dots,M_{h-1}^i$; $j,t = 1,2,\dots,M_h^i, j \neq t$; $\alpha_o = 4$. **Condition 5a** indicates that $\boldsymbol{p}_{h,j}^i$ is recognized as one of the subordinates of $\boldsymbol{p}_{h-1,k}^i$ if its distance to $\boldsymbol{p}_{h-1,k}^i$ is smaller than its distances to other prototypes at the upper layer. **Condition 5b** further adds neighbouring prototypes of $\boldsymbol{p}_{h,j}^i$ at the same layer into $\boldsymbol{\mathcal{L}}_{h-1,k}^i$. Prototypes that satisfy **Condition 5b** are highly likely to be associated with $\boldsymbol{p}_{h-1,k}^i$ because they are spatially close to $\boldsymbol{p}_{h-1,k}^i$. By establishing links between these prototypes and $\boldsymbol{p}_{h-1,k}^i$, the robustness of the decision-making process, which will be detailed in the next subsection, will be significantly enhanced at the price of very little extra computation. Note that all prototypes at the bottom layer have no subordinates and they are the leaf prototypes of the hierarchy, namely, $\boldsymbol{\mathcal{L}}_{H^i,k}^i = \emptyset, k = 1,2,\dots,M_{H^i}^i$.

Once the links between all prototypes at adjacent layers have been built, the learning process of the *i*th hierarchy is completed. The system is ready for classifying unlabelled testing samples after all prototype-based hierarchies have been constructed.

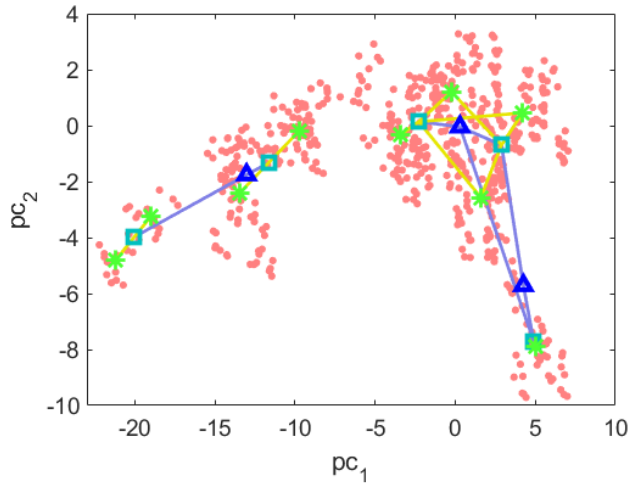
The final three-layer prototype-based hierarchies built by the proposed algorithm from the BA dataset are visualized in Fig. 7 following the illustrative example given by Fig. 6, where lines in different colours stand for the links between prototypes of successive layers.

The main procedure of the prototype-based hierarchy identification process is summarized in the following pseudo code.

Input: $\{\boldsymbol{x}\}_{K^i}^i$ ***Algorithm begins***
i. Calculate D^{MM} at $\{\boldsymbol{u}\}_{L^i}^i$ by (1) and (2); *ii.* Reorder $\{\boldsymbol{u}\}_{L^i}^i$ into $\{\boldsymbol{r}\}^i$ by (3) and (4); *iii.* Identify $\{\boldsymbol{u}\}^{i*}$ by **Condition 1**; *iv.* Form Voronoi tessellation and obtain $\{\mathbf{C}\}^i$ by (6); *v.* Calculate D^{MM} at centres of $\{\mathbf{C}\}^i$ by (7); *vi.* $h \leftarrow 0$; *vii.* **While (Condition 4 is not satisfied)** *a.* $h \leftarrow h + 1$; *b.* Identify $\mathbf{P}_h^i$ by **Conditions 2** and **3**; *c.* Optimize $\mathbf{P}_h^i$ by iteratively minimizing $J(\mathbf{P}_h^i)$; *viii.* **End while** *ix.* Initialize the multi-layered structure with $\mathbf{P}_1^i$, $\mathbf{P}_2^i$,…, $\mathbf{P}_{h-1}^i$ and $\mathbf{P}_h^i$; *x.* Build links between prototypes by **Conditions 5a** and **5b**;
Algorithm ends ***Output: the ith prototype-based hierarchy***



(a) Class 1



(b) Class 2

Fig. 7. Prototype-based hierarchies obtained from the BA dataset using the proposed algorithm (asterisks "*" represent the prototypes identified at the first level of granularity; squares "□" represent the prototypes at the second level of granularity; triangles "Δ" represent prototypes at the third level of granularity; lines in different colours stand for the links between prototypes of successive layers)

3.3. Learning process from streaming data

After the MLOP classifier has self-organized its system structure and meta-parameters from static training data, it can be expected that more training samples become available in the form of data streams. In this subsection, an online learning extension is introduced to the proposed approach, which allows the learning model to continuously self-develop from streaming data on a sample-by-sample basis.

The algorithmic procedure of the online learning process of the MLOP classifier is detailed as follows. Note that, the online learning process is also performed class-wise. During this stage, the classifier will not add new layers, and prototypes may not be at their optimal positions anymore due to the "one pass" updating process.

For each newly available data sample of the *i*th class denoted by $\boldsymbol{x}^{i}_{K^{i}+1}$, the model updating process is performed in a top-down layer-by-layer manner. The average radii of area of influence of prototypes at all H^{i} layers are firstly updated using equation (17) [18]:

$$\gamma_h^i \leftarrow \frac{X_{K^i+1}^i - \left\|\boldsymbol{\mu}_{K^i+1}^i\right\|^2}{X_{K^i}^i - \left\|\boldsymbol{\mu}_{K^i}^i\right\|^2}\gamma_h^i \tag{17}$$

where $h = 1,2,\dots,H^i$; $\boldsymbol{\mu}_{K^i+1}^i = \boldsymbol{\mu}_{K^i}^i + \frac{\boldsymbol{x}_{K^i+1}^i - \boldsymbol{\mu}_{K^i}^i}{K^i+1}$ and $X_{K^i+1}^i = X_{K^i}^i + \frac{\left\|\boldsymbol{x}_{K^i+1}^i\right\|^2 - X_{K^i}^i}{K^i+1}$.

Then, $\boldsymbol{x}_{K^i+1}^i$ is compared with the nearest prototype at the hth layer (starting with $h = 1$), namely, $\boldsymbol{p}_{h,n*}^i$ by **Condition 6** to see whether $\boldsymbol{x}_{K^i+1}^i$ has the potential to become a new prototype at this layer [19]:

$$\textbf{\textit{Condition 6}}: \quad \begin{matrix} If\ \left(\left\|\boldsymbol{p}_{h,n*}^i - \boldsymbol{x}_{K^i+1}^i\right\|^2 > \gamma_h^i\right) \\ Then\ \left(\boldsymbol{x}_{K^i+1}^i\ becomes\ a\ new\ prototype\right) \end{matrix} \tag{18}$$

$$\text{where } \boldsymbol{p}_{h,n_h^*}^i = \begin{cases} argmin_{\boldsymbol{p}\in\mathbf{P}_1^i}(\|\boldsymbol{p}-\boldsymbol{x}\|) & if\ h=1 \\ argmin_{\boldsymbol{p}\in\mathcal{L}_{h-1,n_{h-1}^*}^i}(\|\boldsymbol{p}-\boldsymbol{x}\|) & else \end{cases} \tag{19}$$

The key idea of equation (19) is that, instead of directly searching the nearest one from all leaf prototypes, which can be highly computationally expensive, the classifier searches the nearest prototype layer-by-layer in a top-down manner by only checking the subordinates of the nearest prototype at the next layer. This significantly improves the computational efficiency of the nearest prototype searching process because the searching range is limited to a small group of subordinate prototypes instead of the entire data space. Such searching strategy effectively avoids the waste of computational resources since the majority of prototypes, especially at lower layers, are actually far away from $\boldsymbol{x}$ in the data space and should not be considered for nearest neighbour searching. On the other hand, compared with the similar approach used in [19], the searching strategy proposed in this paper is more robust thanks to the denser connections between prototypes built by **Condition 5b**.

If **Condition 6** is not satisfied, $\boldsymbol{x}_{K^i+1}^i$ is used for updating $\boldsymbol{p}_{h,n*}^i$ [1]:

$$S_{h,n*}^i \leftarrow S_{h,n*}^i + 1; \quad \boldsymbol{p}_{h,n*}^i \leftarrow \boldsymbol{p}_{h,n*}^i + \frac{\boldsymbol{x}_{K^i+1}^i - \boldsymbol{p}_{h,n*}^i}{S_{h,n*}^i} \tag{20}$$

where $S_{h,n*}^i$ is the number of data samples associated with $\boldsymbol{p}_{h,n*}^i$. After this, $\boldsymbol{x}_{K^i+1}^i$ is passed to the next layer ($h \leftarrow h+1$) and compared with the subordinates of $\boldsymbol{p}_{h,n*}^i$ to see whether $\boldsymbol{x}_{K^i+1}^i$ can be a new prototype at the next layer. The same process will be repeated until **Condition 6** is satisfied or $\boldsymbol{x}_{K^i+1}^i$ reaches the bottom layer (namely, $h = H^i$).

If $\boldsymbol{x}_{K^i+1}^i$ meets **Condition 6**, it initializes a new prototype at the hth layer and all the layers below using equation (21) ($k = h, h+1, \dots, H^i$):

$$M_k^i \leftarrow M_k^i + 1; \quad \boldsymbol{p}_{k,M_k^i}^i \leftarrow \boldsymbol{x}_{K^i+1}^i; \quad S_{k,M_k^i}^i \leftarrow 1; \quad \mathbf{P}_k^i \leftarrow \mathbf{P}_k^i \cup \left\{\boldsymbol{p}_{k,M_k^i}^i\right\} \tag{21}$$

After the existing prototypes have been updated and/or new prototypes have been added, links between these prototypes and other prototypes within this hierarchy are updated and/or established using **Condition 5a** and **5b**. The current updating cycle is then completed, and the classifier will move on to learn from the next available data sample following the same algorithmic procedure ($K^i \leftarrow K^i + 1$).

The online prototype-based hierarchy updating process is summarized in the following pseudo code.

Input: $\boldsymbol{x}_{K^i+1}^i, \boldsymbol{x}_{K^i+2}^i, \boldsymbol{x}_{K^i+3}^i, \dots$ ***Algorithm begins***
While ($\boldsymbol{x}_{K^i+1}^i$ is available) *i.* **For** $k = 1$ **to** H^i**:** *a.* Update γ_h^i using (17); *ii.* **End for** *iii.* **For** $h = 1$ **to** H^i**:** *a.* Identify $\boldsymbol{p}_{h,n*}^i$ using (19);

b. **If (Condition 6 is satisfied)**
1. **For** $k = h$ **to** H^i**:**
- $M_k^i \leftarrow M_k^i + 1$;
- Initialize $\boldsymbol{p}_{k,M_k^i}^i$ and $S_{k,M_k^i}^i$, expand $\mathbf{P}_k^i$ using (21);
- Build links of $\boldsymbol{p}_{k,M_k^i}^i$ by **Conditions 5a** and **5b**;
2. **End for**
3. **Break for loop**;

c. **Else**
1. Update $\boldsymbol{p}_{h,n*}^i$ and $S_{h,n*}^i$ using (20);
2. Update links of $\boldsymbol{p}_{h,n*}^i$ by **Conditions 5a** and **5b**;

d. **End if**

iv. **End for**

v. $K^i \leftarrow K^i + 1$;

End while

Algorithm ends

Output: the ith prototype-based hierarchy

3.4. Decision-making process

In this subsection, the algorithmic procedure for decision-making is presented. For an unlabelled sample $\boldsymbol{x}$, the local decision-maker of each prototype-based hierarchy will produce a score of confidence based on the distance between $\boldsymbol{x}$ and the nearest leaf prototype. The score of confidence produced by the ith prototype-based hierarchy is calculated by equation (22):

$$\lambda^i(\boldsymbol{x}) = e^{-\left\| \boldsymbol{p}_{H^i,n_{H^i}^*}^i - \boldsymbol{x} \right\|^2} \tag{22}$$

where $\boldsymbol{p}_{H^i,n_{H^i}^*}^i$ is the nearest leaf prototype to $\boldsymbol{x}$ identified by equation (19) in a top-down, layer-by-layer manner. A visual example of the decision-making process is given by Fig. 8 for better illustration.

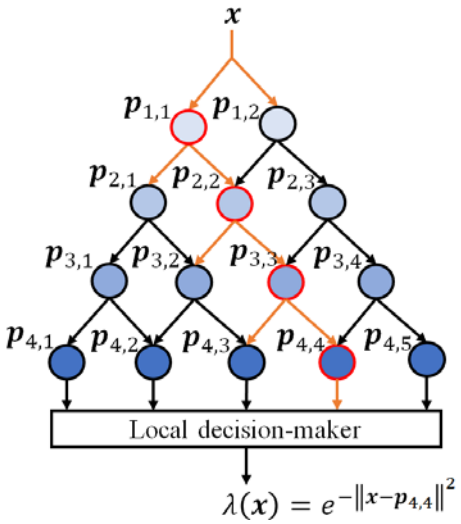


Fig. 8. Illustrative example of the decision-making process (the orange arrows are the exploited paths during the searching process; the nearest prototype at each layer is marked by red circle)

Based on the C scores of confidence (one per class), the global decision-maker of the MLOP classifier determines the class label of $\boldsymbol{x}$ following the "winner takes all" principle:

$$label(\boldsymbol{x}) \leftarrow class\ i^* \quad i^* = argmax_{i=1,2,\ldots,C}\left(\lambda^i(\boldsymbol{x})\right) \tag{23}$$

It is worth noting that the decision-making process of the proposed algorithm follows the "nearest prototype" principle, and scores of confidence are directly calculated from the dissimilarities between data samples and the most similar leaf prototypes. Therefore, there is no randomness existing during this process and one can easily trace back any decisions by examining the exploited paths (see Fig. 8 as example). This allows users to gain a straightforward understanding about the rationales behind the decisions made by the MLOP classifier.

In the next section, computational complexity of the proposed approach will be analysed.

4. Computational Complexity Analysis

4.1. Learning process from statistic data

Since the learning process of the MLOP classifier is conducted class-wise, the *i*th hierarchy is used as an example. In **Stage 1** of the learning process, the complexity of calculating multimodal density values at all unique data samples is $O(NL^i)$; the complexity of ranking unique data samples based on their mutual distances is $O(N(L^i)^2)$; and the complexity of forming Voronoi tessellation and calculating multimodal density values at local maxima are $O(NK^iL^{i*})$ and $O(NL^{i*})$, respectively. During **Stage 2**, the overall computational complexity for estimating the average radius of area of influence and identifying prototypes is $O(N(K^i)^2)$. The computational complexity of the prototype optimization process in **Stage 3** is $O\left(NK^iT_h^iM_h^i\right)$, where T_h^i is the number of iterations for the prototypes converge to the locally optimal positions in the data space; the subscript h stands for the current level of granularity. Computational complexity of **Stage 4** is negligible comparing with other stages. Assuming that **Stages 2-4** are repeated for H^i times (the level of granularity increases from 1 to H^i) until **Condition 4** is satisfied finally, the overall computational complexity of this process is $O\left(N\left(K^i\sum_{h=1}^{H^i}T_h^iM_h^i + H^i(K^i)^2\right)\right)$. **Stage 5** is for building connections between prototypes at adjacent layers and the computational complexity for this process is $O\left(N\left(\sum_{h=1}^{H^i-1}M_h^iM_{h+1}^i + \sum_{h=2}^{H^i}\left(M_h^i\right)^2\right)\right)$. Therefore, the overall computational complex of the entire learning process to build a recognition model from data by the proposed MLOP approach is $O\left(N\sum_{i=1}^{C}\left(K^i\sum_{h=1}^{H^i}T_h^iM_h^i + H^i(K^i)^2 + \sum_{h=1}^{H^i-1}M_h^iM_{h+1}^i + \sum_{h=2}^{H^i}\left(M_h^i\right)^2\right)\right)$.

4.2. Learning process from streaming data

The online updating process of the classifier is also conducted class-wise but on a sample-by-sample basis, therefore, the complexity analysis is performed on a particular updating cycle of the *i*th hierarchy. During the updating cycle, the classifier firstly updates the average radii of area of influence of prototypes at all H^i layers with the newly available data sample, $\boldsymbol{x}_{K^i+1}^i$, and the computational complexity is $O(NH^i)$. However, depending on the mutual distances between $\boldsymbol{x}_{K^i+1}^i$ and prototypes at different layers, the minimum computational complexity of the system updating process is reached if $\boldsymbol{x}_{K^i+1}^i$ meets **Condition 6** at the top layer and is added to the prototype-based hierarchy as a new prototype at every layer. In this case, the computational complexity is $O\left(N\left(M_1^i\right)\right)$. The maximum computational complexity of the updating process is reached if $\boldsymbol{x}_{K^i+1}^i$ fails to satisfy **Condition 6** and is used for updating the nearest prototypes layer-by-layer. In this situation, the computational complexity is $O\left(N\left(M_1^i + \sum_{h=1}^{H^i-1}\left|\boldsymbol{\mathcal{L}}_{h,n_h^*}^i\right|\right)\right)$, where $\left|\boldsymbol{\mathcal{L}}_{h,n_h^*}^i\right|$ denotes the cardinality of $\boldsymbol{\mathcal{L}}_{h,n_h^*}^i$. In both cases, the computational complexity for updating/initializing the links of these prototypes with other prototypes within the hierarchy is $O\left(N\left(\sum_{h=1}^{H^i-1}M_h^i + \sum_{h=2}^{H^i}M_h^i\right)\right)$. Therefore, the computational complexity of a particular online updating cycle is between $O\left(N\left(2\sum_{h=1}^{H^i-1}M_h^i + M_{H^i}^i\right)\right)$ and $O\left(N\left(2\sum_{h=1}^{H^i-1}M_h^i + M_{H^i}^i + \sum_{h=1}^{H^i-1}\left|\boldsymbol{\mathcal{L}}_{h,n_h^*}^i\right|\right)\right)$.

4.3. Decision-making process

During the decision-making process, the complexity of the computational process for each hierarchy to produce a score of confidence on an unlabelled data sample is $O\left(N\left(M_1^i + \sum_{h=1}^{H^i-1}\left|\boldsymbol{\mathcal{L}}_{h,n_h^*}^i\right|\right)\right)$, where $i = 1,2,\dots,C$. Therefore, the overall computational complexity of the decision-making process for a particular data sample is $O\left(N\sum_{i=1}^{C}\left(M_1^i + \sum_{h=1}^{H^i-1}\left|\boldsymbol{\mathcal{L}}_{h,n_h^*}^i\right|\right)\right)$.

5. Experimental Investigation

In this section, numerical examples are presented to justify the effectiveness and validity of the proposed approach. The performance of the MLOP classifier is evaluated on a variety of widely used benchmark datasets and compared with a number of state-of-the-art approaches. The algorithms were developed on MATLAB2018a platform, and the performance was evaluated on a desktop with dual core i7 processer $3.60GHz \times 2$ and $16GB$ RAM. In the numerical examples presented in this section, by default, the MLOP classifier is trained with static data in offline scenarios, and the reported numerical results are obtained after 10 Monte Carlo experiments unless specifically declared otherwise.

5.1. Performance demonstration on numerical datasets

The following eight benchmark datasets are involved in numerical experiments for demonstrating the performance of the proposed approach.

(1) Wilt (WI) dataset[2];

(2) Semeion handwritten digit (SH) dataset[3];

(3) Occupancy detection (OD) dataset[4];

(4) Letter recognition (LR) dataset[5];

(5) Optical recognition of handwritten digits (OR) dataset[6];

(6) Pen-based recognition of handwritten digits (PR) dataset[7];

(7) Phishing websites (PW) dataset[8]; and,

(8) Epileptic seizure recognition (ES) dataset[9].

Details of the eight datasets are summarized in Table 2.

Table 2. Details of benchmark numerical datasets for demonstration

Dataset		# Samples	# Attributes	# Classes
WI	Training set	4339	5 + 1 label	2
	Testing set	500		
SH		1593	256 + 1 label	10
OD [a]	Training	8143	5 + 1 label	2
	Testing set 1	2664		
	Testing set 2	9752		
LR		20000	16 + 1 label	26
OR		5620	62 + 1 label	10
PR		10996	16 + 1 label	10
PW		10550	30 + 1 label	2
ES		11500	178 + 1 label	2

[a] Time stamps of the original dataset have been removed.

In the first numerical example, the influence of ρ on performance and computational complexity of the MLOP classifier is investigated. Four benchmark datasets, namely, WI, SH, OD and LR are involved for investigation.

[2] Available at: http://archive.ics.uci.edu/ml/datasets/wilt.
[3] Available at: https://archive.ics.uci.edu/ml/datasets/semeion+handwritten+digit.
[4] Available at: https://archive.ics.uci.edu/ml/datasets/Occupancy+Detection+.
[5] Available at: https://archive.ics.uci.edu/ml/datasets/letter+recognition.
[6] Available at: https://archive.ics.uci.edu/ml/datasets/optical+recognition+of+handwritten+digits.
[7] Available at: https://archive.ics.uci.edu/ml/datasets/Pen-Based+Recognition+of+Handwritten+Digits.
[8] Available at: https://archive.ics.uci.edu/ml/datasets/phishing+websites.
[9] Available at: https://archive.ics.uci.edu/ml/datasets/Epileptic+Seizure+Recognition.

For SH and LR datasets, 50% of data samples are randomly selected out to form the training sets and the remaining samples are used to form the testing sets. For WI and OD datasets, the original split is adopted, and the order of the training samples is randomly scrambled for each experiment. During the experiment, the value of ρ varies from 0 to 0.3. Numerical results including the average layer number (# Layer), classification accuracy, training and testing time consumptions (in seconds) are reported in Table 3 in the form of $mean \pm standard\ deviation$. To gain a better picture regarding the system complexity, numbers of prototypes at different layers of the two hierarchies identified from the training set of the OD dataset (with $\rho = 0.05$) during a particular experiment are reported in Fig. 9.

As one can see from Table 3, generally, the performance of the proposed approach in terms of classification accuracy is stronger with a smaller value of ρ. However, this will inevitably increase the system complexity because there will be more layers and more prototypes (see Fig. 9) in the system structure. On the other hand, Table 3 also shows that the computational complexity of the training and testing processes is only slightly increased with the system structure going deeper. Therefore, in the following numerical examples presented in this paper, $\rho = 0.05$ is used by default to pursue higher classification precision.

Table 3. Influence of different values of ρ on the performance of the MLOP classifier

Dataset	ρ	# Layer	Accuracy	Training time, s	Testing time, s
WI	0.00	9.0	0.7440±0.0000	3.42±0.17	0.60±0.08
	0.05	5.5	0.7760±0.0000	1.86±0.11	0.32±0.06
	0.10	5.0	0.7880±0.0000	1.73±0.06	0.28±0.03
	0.15	4.0	0.7860±0.0000	1.47±0.03	0.20±0.01
	0.20	4.0	0.7860±0.0000	1.44±0.01	0.21±0.04
	0.25	4.0	0.7860±0.0000	1.47±0.04	0.20±0.02
	0.30	4.0	0.7860±0.0000	1.44±0.01	0.20±0.01
SH	0.00	4.3	0.8601±0.0101	0.08±0.05	1.85±0.54
	0.05	4.3	0.8601±0.0101	0.06±0.01	1.83±0.54
	0.10	4.3	0.8601±0.0101	0.06±0.01	1.79±0.51
	0.15	4.3	0.8601±0.0101	0.06±0.01	1.79±0.53
	0.20	4.2	0.8600±0.0101	0.06±0.01	1.94±0.63
	0.25	4.2	0.8599±0.0101	0.06±0.02	1.89±0.61
	0.30	4.2	0.8596±0.0099	0.06±0.01	1.80±0.47
OD	0.00	14.0	0.9435±0.0000	2.10±0.12	16.17±0.12
	0.05	5.0	0.9441±0.0000	2.01±0.02	5.57±0.07
	0.10	4.5	0.9300±0.0000	2.01±0.04	4.95±0.02
	0.15	4.0	0.9290±0.0000	2.04±0.04	4.42±0.05
	0.20	3.5	0.9079±0.0000	2.61±0.04	3.81±0.02
	0.25	3.5	0.9079±0.0000	2.62±0.07	3.81±0.02
	0.30	3.0	0.9075±0.0000	2.59±0.08	3.28±0.20
LR	0.00	8.7	0.9283±0.0035	0.81±0.09	122.13±2.14
	0.05	7.6	0.9281±0.0034	0.74±0.03	107.67±1.92
	0.10	6.1	0.9273±0.0035	0.67±0.04	102.03±4.36
	0.15	5.4	0.9253±0.0036	0.63±0.03	93.22±1.75
	0.20	5.0	0.9234±0.0051	0.56±0.01	77.55±2.11
	0.25	4.7	0.9187±0.0034	0.53±0.01	71.44±1.48
	0.30	4.4	0.9144±0.0036	0.50±0.02	65.91±2.42

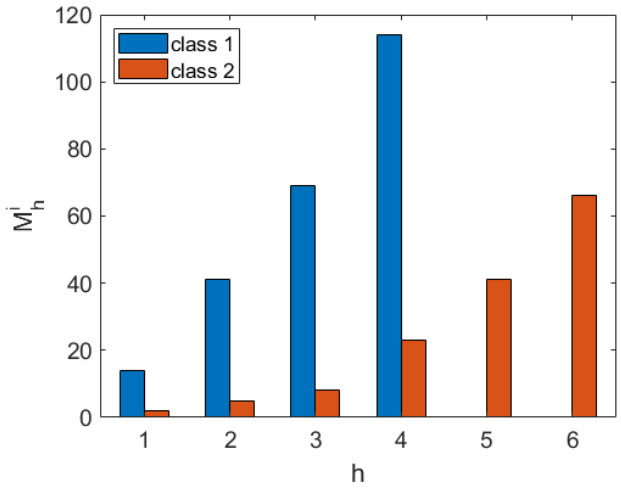


Fig. 9. Number of prototypes at different hierarchies identified from training data with $\rho = 0.05$

In the following numerical example, the online learning ability of the proposed MLOP classifier is demonstrated based on eight benchmark datasets as listed in Table 1, namely, WI, SH, OD, LR, OR, PR, FW and ES. In particular, for SH, LR, FW and ES datasets, 50% of data samples are randomly selected out to form the training sets and the remaining samples are used to form the testing sets. For WI, OD, PR and OR datasets, the original splits are kept but the orders of training samples are randomly scrambled for each experiment. During the experiments, the MLOP classifier is firstly trained with half of the training set in an offline scenario and, then, the classifier continues to self-update based on the remaining 10%, 20%, 30%, 40% and 50% training data on a sample-by-sample basis online. The classification accuracy rates of the classifier on the testing sets are reported in Table 4. The performance of the MLOP classifier trained with the entire training set offline is given in the same table as baseline. From Table 4 one can conclude that the MLOP classifier can effectively learn from new streaming data samples online after being primed with static data and maintain very high classification precision.

Table 4. Classification accuracy rates of the MLOP classifier trained in a hybrid manner with different amounts of data for online training

Dataset	Offline	Online Training					Baseline
	50%	10%	20%	30%	40%	50%	
WI	0.7482	0.7602	0.7734	0.7798	0.7856	0.7884	0.7760
SH	0.8318	0.8367	0.8389	0.8449	0.8477	0.8511	0.8601
OD	0.9455	0.9368	0.9359	0.9336	0.9320	0.9288	0.9441
LR	0.8843	0.9001	0.9123	0.9202	0.9269	0.9326	0.9281
OR	0.9716	0.9735	0.9743	0.9762	0.9765	0.9778	0.9850
PR	0.9673	0.9692	0.9709	0.9732	0.9734	0.9744	0.9757
PW	0.9155	0.9177	0.9216	0.9259	0.9295	0.9334	0.9371
ES	0.9008	0.9079	0.9079	0.9198	0.9237	0.9268	0.9204

Furthermore, the impact of different amounts of offline training data on the overall performance of the MLOP classifier trained in a hybrid manner (namely, offline training first followed by online training) is further investigated. In this example, the MLOP classifier is firstly trained with 30%, 40%, 50%, 60%, 70% and 80% of the training data offline, and then continues to learn from the remaining data on a sample-by-sample basis online. Classification accuracy rates of the classifier obtained on the testing sets are presented in Table 5. The performance of the MLOP classifier trained with the entire training set offline is also given here as baseline. It can be seen from Table 5 that, in general, the overall performance of the MLOP classifier trained in a hybrid manner is less influenced by the available amount of offline training data. This further demonstrates the effectiveness of the online learning mechanism of the proposed approach.

Table 5. Classification accuracy rates of the MLOP classifier trained in a hybrid manner with different amounts of offline training data

Dataset	Offline Training						Baseline
	30%	40%	50%	60%	70%	80%	
WI	0.7476	0.7652	0.7884	0.7882	0.7914	0.7806	0.7760
SH	0.8576	0.8525	0.8511	0.8545	0.8551	0.8599	0.8601
OD	0.9270	0.9304	0.9288	0.9288	0.9295	0.9296	0.9441
LR	0.9329	0.9335	0.9326	0.9312	0.9309	0.9291	0.9281
OR	0.9720	0.9737	0.9778	0.9767	0.9774	0.9786	0.9850
PR	0.9746	0.9749	0.9744	0.9752	0.9742	0.9730	0.9757
PW	0.9355	0.9357	0.9334	0.9356	0.9364	0.9339	0.9371
ES	0.9260	0.9266	0.9268	0.9246	0.9240	0.9225	0.9204

For better demonstration, statistical performance (classification accuracy and training time consumption, in seconds) of the MLOP classifier on the eight benchmark problems, namely, WI, SH, OD, LR, OR, PR, PW and ES, in offline scenarios is reported in Table 6 in the form of $mean \pm standard\ deviation$. During the experiments, the same training-testing splits of the eight datasets used in the previous examples are considered. In addition, statistical performance of the MLOP classifier trained in a hybrid manner (denoted by MLOP-H) is also reported in this table. During the experiments, the MLOP-H classifier is firstly primed with 50% of training samples offline and continuously trained with the remaining 50% training samples online.

The performance of the proposed approach is further compared with the following algorithms:

(1) SVM classifier with linear kernel [7];

(2) DT classifier [27];

(3) KNN classifier with $k = 5$ [8];

(4) Sequence-dictionary-based KNN (SDKNN) classifier with $k = 5$ [32];

(5) Sequence classifier (SC) [32];

(6) Extreme learning machine (ELM) classifier with maximum 200 neurons [21];

(7) Multi-layer perceptron (MLP) network with three hidden layers and 20 neurons in each hidden layer;

(8) GLVQ with 25 reference vectors per class and the gain factor set as $\alpha = 0.005$ [35];

(9) Extended sequential adaptive fuzzy inference system (ESAFIS) [34];

(10) Zero-order autonomous learning multiple-model system (ALMMo0) [1];

(11) Self-organising neuro-fuzzy inference system (SONFIS) with the level of granularity set as $G = 5$ [18];

(12) Hierarchical prototype-based (HP) classifier with six layers ($H = 6$) [19].

SVM, DT and KNN are the most used generic approaches for classification. MLP uses the well-known resilient back propagation algorithm for parameter optimization. GLVQ is a prototype-based ANN popular for multi-class classification. ELM is a single-layer feedforward neural network for classification. SC and SDKNN are two recently introduced dictionary-based approaches for classification. ESAFIS is a first-order EIS designed for regression and classification. SONFIS and ALMMo0 are both zero-order EISs with a prototype-based nature. HP classifier is prototype-based approach with a multi-layered structure, which is of the same type as the proposed MLOP classifier. Nonetheless, the layer number of the HP classifier is predefined by users instead of being determined by data and it employs cosine dissimilarity as the distance measure. During the experiments, the externally controlled parameters for the involved classification approaches are determined by the commonly used experimental protocols and are fixed across different problems due to insufficient prior knowledge. The numerical results obtained by the 12 comparative approaches on the eight benchmark datasets are also reported in Table 6

following the same experimental protocols, where the training time consumptions of KNN are not reported because it literally requires no training.

For better demonstration, the comparative algorithms are ranked from best to worst on each dataset in terms of classification accuracy, and the ranks are given in Table 7. The overall ranks of the algorithms across the eight datasets are given in the same table as well. One can see from Table 7 that the overall performance of the proposed MLOP classifier is in the second place among the involved classification methods for comparison while the HP classifier ranks the top, but the performance of the proposed MLOP approach is relatively more stable.

Table 6. Performance comparison between the 14 classification approaches in terms of classification accuracy and training time consumption on the eight benchmark datasets (the best accuracy rates are bolded)

Dataset	WI		SH	
Algorithm	Accuracy	Training time, s	Accuracy	Training time, s
MLOP	0.7760±0.0000	1.86±0.11	0.8601±0.0101	0.06±0.01
MLOP-H	0.7884±0.0346	4.03±0.09	0.8511±0.0091	0.69±0.03
SVM	0.7150±0.0017	54.58±1.73	**0.9174±0.0096**	0.49±0.29
DT	0.8140±0.0000	0.04 ± 0.06	0.7103±0.0227	0.12±0.21
KNN	0.7260±0.0000		0.8767±0.0139	
SDKNN	0.3980±0.0000	4.52±0.26	0.8242±0.0255	2.05±0.10
SC	0.6720±0.0000	5.27±0.27	0.8927±0.0109	3.45±0.13
ELM	**0.8574±0.0044**	0.06±0.03	0.3923±0.1212	0.04±0.03
MLP	0.7008±0.0874	0.73±0.25	0.5933±0.0455	0.40±0.22
GLVQ	0.6260±0.0000	19.70±0.06	0.8313±0.0142	28.07±1.29
ESAFIS	0.6258±0.0018	5.53±3.09	0.6736±0.0427	87.58±5.08
ALMMo0	0.7728±0.0129	0.39±0.08	0.8934±0.0090	0.09±0.03
SONFIS	0.7960±0.0000	1.12 ± 0.04	0.8680±0.0113	0.04±0.05
HP	0.8046±0.0166	2.85±0.17	0.8951±0.0097	0.35±0.03
Dataset	OD		LR	
Algorithm	Accuracy	Training time, s	Accuracy	Training time, s
MLOP	0.9441±0.0000	2.01±0.02	0.9281±0.0034	0.74±0.03
MLOP-H	0.9288±0.0045	4.19±0.27	0.9326±0.0018	4.83±0.11
SVM	0.6787±0.2118	164.24±10.28	0.8540±0.0030	15.77±0.54
DT	0.9314±0.0000	0.04±0.05	0.8235±0.0068	0.12±0.05
KNN	0.9580±0.0000		0.9325±0.0020	
SDKNN	0.7576±0.0000	7.83±0.22	0.8339±0.0050	12.02±0.49
SC	0.8984±0.0000	22.00±0.73	0.8561±0.0028	27.71±1.09
ELM	**0.9895±0.0002**	0.13±0.03	0.5001±0.0585	0.16±0.04
MLP	0.9152±0.0268	1.05±0.58	0.5160±0.0306	2.24±0.32
GLVQ	0.9314±0.0000	25.86±1.37	0.7606±0.0089	56.30±0.94
ESAFIS	0.9586±0.0237	15.26±6.71	0.8251±0.0077	66.85±2.31
ALMMo0	0.9404±0.0040	0.63±0.11	0.9179±0.0029	0.76±0.19
SONFIS	0.9382±0.0000	2.51±0.07	0.9223±0.0052	0.27±0.15
HP	0.7873±0.0116	3.91±0.70	**0.9414±0.0011**	5.31±0.26
Dataset	OR		PR	
Algorithm	Accuracy	Training time, s	Accuracy	Training time, s
MLOP	**0.9850±0.0000**	0.52±0.02	0.9757±0.0000	0.93±0.03
MLOP-H	0.9786±0.0023	2.62±0.05	0.9744±0.0026	5.07±0.24
SVM	0.9627±0.0000	0.75±0.31	0.9551±0.0000	58.65±0.90
DT	0.8525±0.0000	0.06±0.03	0.9122±0.0003	0.04±0.03
KNN	0.9794±0.0000		0.9760±0.0000	
SDKNN	0.9627±0.0000	5.08±0.45	0.9531±0.0000	8.33±0.38
SC	0.9549±0.0000	8.41±0.69	0.9511±0.0000	12.50±0.45
ELM	0.9053±0.0096	0.07±0.03	0.9086±0.1622	0.10±0.03
MLP	0.9304±0.0077	0.92±0.14	0.9459±0.0078	1.13±0.29
GLVQ	0.9199±0.0000	34.25±1.01	0.8453±0.0000	33.56±0.62

ESAFIS	0.9538±0.0067	32.24±10.36	0.9197±0.0134	38.80±5.93
ALMMo0	0.9789±0.0000	0.33±0.09	0.9752±0.0023	0.56±0.05
SONFIS	0.9766±0.0000	0.09±0.06	0.9763±0.0000	0.39±0.03
HP	0.9772±0.0000	1.56±0.21	**0.9782±0.0001**	3.87±0.07
Dataset	PW		ES	
Algorithm	Accuracy	Training time, s	Accuracy	Training time, s
MLOP	0.9371±0.0047	1.85±0.08	0.9204±0.0071	41.77±3.43
MLOP-H	0.9334±0.0052	6.17±0.43	0.9268±0.0119	72.22±16.21
SVM	0.9262±0.0027	4.76±1.46	0.1921±0.0319	96.19±2.41
DT	**0.9466±0.0030**	0.09±0.16	0.9351±0.0036	0.57±0.08
KNN	0.9314±0.0045		0.9146±0.0072	
SDKNN	0.9107±0.0048	5.64±0.40	0.9198±0.0050	15.04±0.52
SC	0.9456±0.0029	13.71±0.59	**0.9452±0.0028**	36.29±1.47
ELM	0.9100±0.0064	0.09±0.03	0.2719±0.2317	0.12±0.03
MLP	0.9386±0.0066	0.57±0.34	0.9408±0.0493	1.05±0.40
GLVQ	0.9099±0.0050	24.65±0.90	0.6983±0.2589	35.53±0.94
ESAFIS	0.9413±0.0027	244.20±69.50	0.9044±0.0123	909.64±89.66
ALMMo0	0.9436±0.0044	0.93±0.08	0.8937±0.0025	25.77±3.19
SONFIS	0.9138±0.0038	0.66±0.09	0.8884±0.0140	2.24±0.07
HP	0.9450±0.0050	1.92±0.39	0.8941±0.0026	3.23±0.37

Table 7. Overall classification accuracy ranks of comparative algorithms

Algorithm	Rank								
	WI	SH	OD	LR	OR	PR	PW	ES	Overall
MLOP	*6*	*7*	*4*	*4*	*1*	*4*	*7*	*5*	4.8±2.0
MLOP-H	*5*	*8*	*9*	*2*	*4*	*6*	*8*	*4*	5.8±2.4
SVM	*9*	*1*	*14*	*8*	*7*	*7*	*10*	*14*	8.8±4.2
DT	*2*	*11*	*7*	*11*	*14*	*12*	*1*	*3*	7.6±5.1
KNN	*8*	*5*	*3*	*3*	*2*	*3*	*9*	*7*	5.0±2.7
SDKNN	*14*	*10*	*13*	*9*	*8*	*8*	*12*	*6*	10.0±2.8
SC	*11*	*4*	*11*	*7*	*9*	*9*	*2*	*1*	6.8±4.0
ELM	*1*	*14*	*1*	*14*	*13*	*13*	*13*	*13*	10.3±5.7
MLP	*10*	*13*	*10*	*13*	*11*	*10*	*6*	*2*	9.4±3.7
GLVQ	*12*	*9*	*8*	*12*	*12*	*14*	*14*	*12*	11.6±2.1
ESAFIS	*13*	*12*	*2*	*10*	*10*	*11*	*5*	*8*	8.9±3.7
ALMMo0	*7*	*3*	*5*	*6*	*3*	*5*	*4*	*10*	5.4±2.3
SONFIS	*4*	*6*	*6*	*5*	*6*	*2*	*11*	*11*	6.4±3.2
HP	*3*	*2*	*12*	*1*	*5*	*1*	*3*	*9*	4.5±4.0

In addition, confusion matrices obtained by the MLOP classifier and the 12 comparative algorithms on the four binary classification problems, namely, WI, OD, PW and ES are reported in Table 8 for better comparison.

Table 8. Confusion matrices obtained by the MLOP and 12 comparative classifiers on the four binary classification problems (the best results are bolded)

Dataset	Algorithm	True positive	False negative	False positive	True negative
WI	MLOP	79.0	108.0	4.0	309.0
	SVM	66.2	120.8	21.7	291.3
	DT	107.0	80.0	13.0	300.0
	KNN	51.0	136.0	1.0	312.0
	SDKNN	38.0	149.0	15.0	298.0
	SC	187.0	0.0	301.0	12.0
	ELM	**125.5**	**61.5**	**9.8**	**303.2**

	MLP	44.5	142.5	7.1	305.9
	GLVQ	0.0	187.0	0.0	313.0
	ESAFIS	0.2	186.8	0.3	312.7
	ALMMo0	83.1	103.9	9.7	303.3
	SONFIS	111.0	76.0	26.0	287.0
	HP	117.6	69.4	28.3	284.7
OD	MLOP	9098.0	298.0	396.0	2625.0
	SVM	7189.9	2206.1	1784.1	1236.9
	DT	9048.0	348.0	504.0	2517.0
	KNN	9030.0	366.0	156.0	2865.0
	SDKNN	8755.0	641.0	620.0	2401.0
	SC	9394.0	2.0	3008.0	13.0
	ELM	**9281.8**	**114.2**	**16.3**	**3004.7**
	MLP	8697.7	698.3	354.2	2666.8
	GLVQ	8724.0	672.0	180.0	2841.0
	ESAFIS	9073.6	322.4	191.7	2829.3
	ALMMo0	9100.5	295.5	444.8	2576.2
	SONFIS	9207.0	189.0	578.0	2443.0
	HP	7647.3	1748.7	893.0	2128.0
PW	MLOP	2893.6	184.9	162.9	2287.1
	SVM	2913.4	165.1	242.9	2207.1
	DT	**2939.0**	**139.5**	**155.8**	**2294.2**
	KNN	2898.6	179.9	199.4	2250.6
	SDKNN	2968.4	110.1	190.8	2259.2
	SC	2831.0	247.5	246.3	2203.7
	ELM	2935.4	143.1	354.3	2095.7
	MLP	2925.4	153.1	186.2	2263.8
	GLVQ	2886.9	191.6	306.4	2143.6
	ESAFIS	2950.3	128.2	196.3	2253.7
	ALMMo0	2936.3	142.2	169.6	2280.4
	SONFIS	2894.5	184.0	292.5	2157.5
	HP	2982.9	95.6	208.4	2241.6
ES	MLOP	710.1	437.2	18.9	4583.8
	SVM	779.0	368.3	4277.2	325.5
	DT	955.7	191.6	181.7	4421.0
	KNN	662.5	484.8	6.0	4596.7
	SDKNN	843.1	304.2	11.0	4591.7
	SC	**692.9**	**454.4**	**6.6**	**4596.1**
	ELM	1131.1	16.2	4170.5	432.2
	MLP	878.9	268.4	72.1	4530.6
	GLVQ	323.5	823.8	911.0	3691.7
	ESAFIS	690.6	456.7	93.0	4509.7
	ALMMo0	795.8	351.5	259.9	4342.8
	SONFIS	604.9	542.4	99.5	4503.2
	HP	795.5	351.8	257.3	4345.4

5.2. Performance demonstration on benchmark image sets

In this subsection, the following four challenging image recognition problems are involved for evaluating the performance of the proposed approach on large-scale, high-dimensional, complex problems.

(1) MNIST dataset[10];

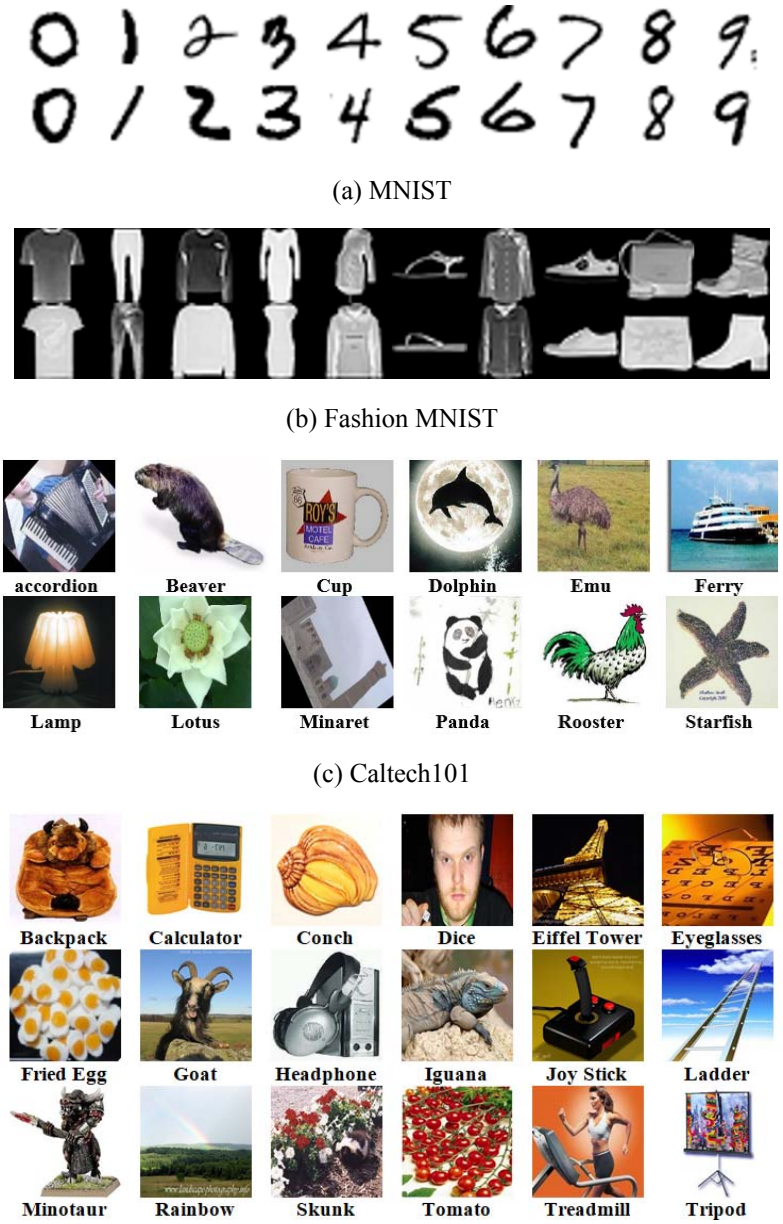
(2) Fashion MNIST dataset[11];

[10] Available at http://yann.lecun.com/exdb/mnist/.
[11] Available at https://github.com/zalandoresearch/fashion-mnist.

(3) Caltech101 dataset[12], and;

(4) Caltech256 dataset[13];

Example images of the four datasets are given in Fig. 10.



(a) MNIST

(b) Fashion MNIST

(c) Caltech101

(d) Caltech256

Fig. 10. Example images of the four benchmark image sets for performance evaluation.

Firstly, the performance of the MLOP classifier for image classification is evaluated on MNIST and Fashion MNIST datasets. In this example, for both datasets, all the training and testing images are converted into 784×1 dimensional vectors and, then, directly used to train and test the proposed algorithm. The classification accuracy rates of the proposed approach on testing images of the two problems are reported in Table 9. For better evaluation,

[12] Available at: http://www.vision.caltech.edu/Image_Datasets/Caltech101/.
[13] Available at: http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

10 comparative algorithms used in previous experiments are involved in this numerical example under the same experimental protocol, and their performances in terms of classification accuracy rates are reported in Table 9 for visual clarity. Note that, in this example, a MLP with three hidden layers, 200 neurons in each (in total, 600 neurons) is used instead due to the much larger problem size.

Table 9. Classification accuracy comparison between the MLOP classifier and comparative algorithms on MNIST and Fashion MNIST datasets (the best accuracy rates are bolded)

Algorithm	MNIST		Fashion MNIST	
	Original	Gist	Original	Gist
MLOP	**0.9696**	0.9835	0.8496	0.8886
SVM	0.9438	0.9857	0.8497	0.8871
DT	0.8779	0.9010	0.7934	0.8142
KNN	0.9684	**0.9875**	0.8554	**0.8957**
SDKNN	0.9555	0.9802	0.8660	0.8876
SC	0.9646	0.9762	**0.8738**	0.8817
ELM	0.1453	0.9122	0.3557	0.8045
MLP	0.8781	0.9577	0.7637	0.8869
ALMMo0	0.9683	0.9864	0.8589	0.8862
SONFIS	0.9681	0.9865	0.8610	0.8876
HP	0.9652	0.9864	0.8557	0.8845

Furthermore, the Gist feature descriptor [30] is used to extract 512×1 dimensional feature vectors from the original training/testing images of the two problems. Experiments are repeated by training and testing the algorithms with the Gist features of the images. The obtained results by the 11 approaches are reported in Table 9 as well. The average layer numbers of the prototype-based hierarchies that the MLOP classifier self-organized from the training images during the experiments are presented in Fig. 11.

Finally, the performance of the MLOP classifier is evaluated on Caltech101 and Caltech256 datasets. A ensemble feature descriptor formed by pretrained AlexNet [24] and VGG-VD-16 [38] models is employed for feature extraction, which extracts a 9192×1 dimensional discriminative representation, denoted by $\boldsymbol{x}$ from each image, $\mathbf{I}$:

$$\boldsymbol{x} = F(\mathbf{I}) = \left[\frac{AN(\boldsymbol{x})}{\|AN(\boldsymbol{x})\|}, \frac{VN(\boldsymbol{x})}{\|VN(\boldsymbol{x})\|}\right]^T \tag{24}$$

where $F(\mathbf{I})$ is the 9192×1 dimensional discriminative representation of $\mathbf{I}$ obtained by the ensemble feature descriptor; $AN(\boldsymbol{x})$ and $VN(\boldsymbol{x})$ are the 1×4096 dimensional activations extracted from the first fully connected layer of the AlexNet and VGG-VD-16 models, respectively.

Following the common practice [41], for Caltech101 image set, 15 and 30 images are randomly selected out from each category for training, respectively, and the rest of the dataset are used for testing. For Caltech256 image set, 15, 30, 45 and 60 images are randomly selected from each category for training, respectively, and the rest images are used for testing. The classification accuracy rates achieved by the MLOP classifier and four comparative approaches, namely, SVM, KNN, ALMMo0 and HP, on the testing sets of the two datasets are tabulated in Tables 10 and 11, respectively. In addition, selected state-of-the-art results reported by recent publications are given in the same tables for better evaluation. The average layer numbers of the prototype-based hierarchies that MLOP classifier self-organized from the training sets of Caltech101 and Caltech256 are presented in Fig. 11 as well.

Furthermore, three prototype-based hierarchies that the MLOP classifier identifies from training images of the "airplane", "llama" and "snoopy" classes (30 images per class) in the Caltech101 dataset during a particular experiment are visualized in Fig. 12. However, as both the learning and classification processes of the MLOP classifier are conducted based on the feature vectors instead of raw images during numerical experiments, the training images with feature vectors that are the most similar to the identified prototypes are used for visualization in the depicted hierarchies.

Table 10. Classification accuracy comparison between the MLOP classifier and comparative algorithms on Caltech101 dataset (the best accuracy rates are bolded)

Algorithm	Accuracy	
	15 Training Images	30 Training Images
MLOP	**0.9037±0.0047**	**0.9272±0.0043**
SVM	0.8729±0.0104	0.9027±0.0087
KNN	0.8721±0.0079	0.9100±0.0042
ALMMo0	0.8491±0.0062	0.8864±0.0047
HP	0.8863±0.0060	0.9224±0.0040
ICAC [44]	0.7148±0.0056	0.7663±0.0079
CASE-LLC-SVM [28]	0.6400±0.0040	0.7140±0.0120
DEFEATnet [11]	0.7128±0.0061	0.7760±0.0096
RNPCANet [33]	-	0.7227±0.0102
VLAD-LLC [26]	-	0.8923
LEFSI [31]	0.7721±0.0061	0.8578±0.0037
ESRO-BoW [13]	0.8052	0.8535
DEL-BoW [14]	0.8269	0.8691

Table 11. Classification accuracy comparison between the MLOP classifier and comparative algorithms on Caltech256 dataset (the best accuracy rates are bolded)

Algorithm	Accuracy			
	15 Training Images	30 Training Images	45 Training Images	60 Training Images
MLOP	**0.6854±0.0033**	**0.7144±0.0032**	**0.7265±0.0030**	**0.7376±0.0027**
SVM	Out of System Memory			
KNN	0.6251±0.0031	0.6805±0.0029	0.7091±0.0035	0.7310±0.0024
ALMMo0	0.6239±0.0033	0.6711±0.0026	0.6976±0.0034	0.7187±0.0029
HP	0.6353±0.0045	0.6908±0.0029	0.7172±0.0034	0.7347±0.0030
DEFEATnet [11]	0.3507±0.0038	0.4206±0.0025	0.4598±0.0026	0.4852±0.0032
VLAD-LLC [26]	-	-	-	0.7425
LEFSI [31]	0.3657±0.0056	0.4721±0.0069	0.5081 ± 0.0041	0.5290 ± 0.0048
DEL-BoW [14]	0.6122	0.6925	-	0.7257
LSC-LG [43]	0.4314±0.0063	0.5062±0.0053	0.5327±0.0056	0.5576±0.0048
OCB-FV [47]	0.4403±0.0046	0.5315±0.0044	0.5784±0.0040	0.5903±0.0045
SWSS-DeCAF [45]	0.6152±0.0039	0.6768±0.0065	0.6977±0.0053	0.7283±0.0044
SWSS-FV [45]	0.4246±0.0038	0.4985±0.0042	0.5466±0.0047	0.5652±0.0041
SC^2-CNN [46]	0.4758±0.0062	0.5542±0.0056	0.5912±0.0051	0.6174±0.0050
ResNet101-COMO-2 [48]	-	0.4466	-	0.6707

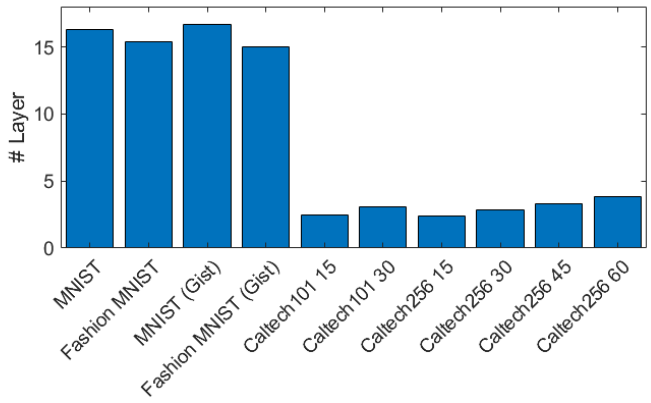
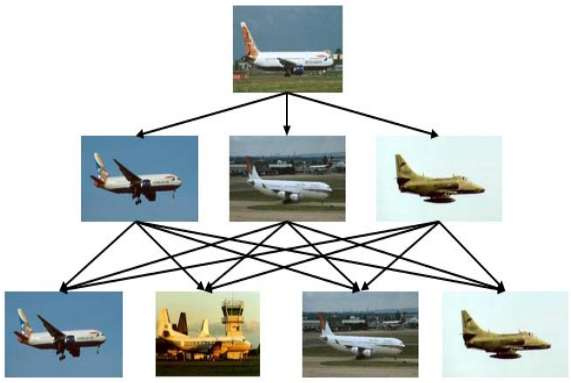
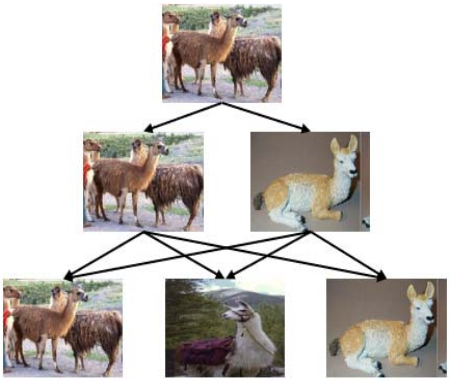


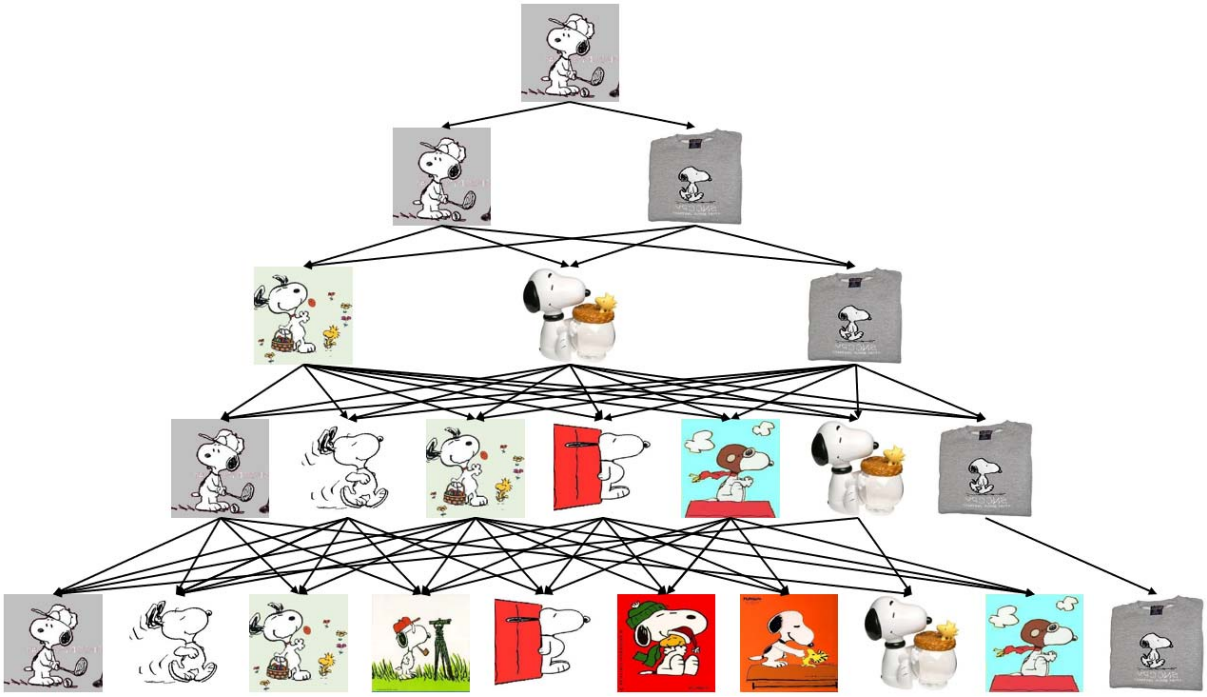
Fig. 11. Average layer numbers of the prototype-based hierarchies identified by the MLOP classifier during numerical experiments on benchmark image sets



(a)Airplane



(b) Llama



(c) Snoopy

Fig. 12. Three prototype-based hierarchies that the MLOP classifier identifies from the respective training images of three different classes of the Caltech101 dataset

5.3. Discussions

From Tables 6-11 one can see that the proposed approach demonstrates very high classification performance on all benchmark problems surpassing or, at least, on par with the state-of-the-art approaches. In addition, despite that the MLOP classifier learns from static data offline, Table 6 shows that in many cases, its computational efficiency is higher than the majority of alternative prototype-based approaches involved in the experiments. It is demonstrated by Tables 4, 5 and 6 that the MLOP classifier can continue to self-learn and self-update from streaming data online to incorporate newly observed data patterns. In addition, one interesting feature of the MLOP classifier is that its multi-layered system structure and meta-parameters are derived from data based on the ensemble properties and mutual distribution in a straightforward way. The user-controlled parameter, ρ only controls the trade-off between the intra-cluster variance and the number of prototypes in the constructed model without influencing the objectiveness of the prototype identification process. Therefore, the MLOP classifier keeps the advantage of objectiveness of data-driven approaches, and the effectiveness and validity of its learning results are always guaranteed for different problems.

On the other hand, it needs to be kept in mind that a classifier might behave very differently depending on the nature of data. The proposed MLOP classifier categorizes unlabelled samples based on the "nearest prototype" principle. This decision-making mechanism is similar to the "nearest neighbour" principle used by the KNN classifier and would become less effective when data is not linearly separable. Therefore, one may notice from Table 6 that MLOP generally performs less well on nonlinear problems such as the WI, PW and ES dataset. The same problem can be observed from the KNN, ALMMo0 and HP classifiers as well. Meanwhile, DT, MLP, SC and ELM usually perform better on such types of problems because they are nonlinear classifiers. To further improve the ability of the MLOP classifier on nonlinear problems, one may use the kernel trick to transform the observed data samples into higher dimensions and make them linearly separable.

Furthermore, the MLOP classifier in this paper uses the Euclidean distance as the default distance measure. However, it has been widely recognized that different distance measures have different focuses on disclosing the ensemble properties of data. The differences can be even more significant in higher dimensional data spaces. For example, many commonly used distance metrics, including Euclidean distance, city block distance and Mahalanobis distance, suffer from the so-called "curse of dimensionality". In contrast, cosine dissimilarity is more frequently used for handling high-dimensional problems. Therefore, it is of great importance for a classifier to use a suitable distance measure for a particular problem. Otherwise, this might have an adverse impact on the performance of the system. As another direction for improvement, it would be valuable to introduce some modifications to the MLOP classifier, enabling it to work with various types of distance measures.

It also has to be admitted that the proposed MLOP classifier might not be suitable for extremely large-scale problems. Its operating mechanism requires a time-consuming iterative searching process to identify the locally optimal prototypes for classification. This would become a huge computation burden if there is a huge amount of training samples or a locally optimal solution cannot be easily found due to the very high dimensionality and/or complex structure of data. Therefore, it can be observed that the computation efficiency of the MLOP classifier is surpassed by a number of comparative approaches on WI, OD, PW and ES datasets. Although the MLOP classifier can continuously self-learn from new data sample-by-sample after being primed offline, the computational efficiency of the online learning process is lower than ALMMo0 and HP classifiers due to its multi-layered structure and denser links in-between. One feasible way to address this issue is to introduce some alternative online learning mechanism to the MLOP classifier.

Last but not least, it needs to be clarified that the model transparency and explainability offered by the hierarchical prototype-based structure and the traceable inference mechanism of the MLOP classifier are mostly for machine learning experts/specialists to perform model diagnostics rather than for end users. To further enhance the end-user explainability, one possible solution is to provide users with local explanation in terms of attribute importance based on the model topology. However, this is beyond the scope of this paper.

6. Conclusion and Future Work

This paper presents a novel approach named MLOP for classification. The proposed MLOP approach identifies locally optimal prototypes from data at multiple levels of granularity and self-organizes a multi-layered system structure by aggregating these prototypes in a pyramidally hierarchical form in regard to the respective levels of granularity. Unlike alternative mainstream algorithms, i.e., DNNs, the inner structure of the MLOP classifier is highly transparent and interpretable. In addition, it can continuously self-update with new data samples. Its decision-making process is fully explainable and traceable by following the "nearest prototype" principle, which is of great importance for financial and safety-critical applications. Most importantly, the learned knowledge by the proposed approach from data can be visualized in a human-interpretable form of prototype-based hierarchies. A much smaller number of highly abstract prototypes at the top layers of the hierarchies can provide users a general picture of the problem by summarizing key information. Meanwhile, a larger number of low-level prototypes can provide users fine details of the problem. Numerical examples demonstrated the efficacy of the proposed MLOP approach, showing its strong potential in real-world applications by offering both high predictive precision and explainablity.

There are several considerations for future work. Firstly, the MLOP classifier requires one user-controlled parameter to control the trade-off between the intra-cluster variance and the number of prototypes in the system (equation (13)). Although this parameter can be determined without prior knowledge of the problem, involving a new mechanism to automatically self-adjust its value based on the observed data can further enhance the autonomy and robustness of the proposed approach. Secondly, some modifications are needed to enable the MLOP classifier

to work with different types of distance measures, enhancing its ability to handle data of different natures. Thirdly, the online learning scheme of the proposed approach needs to be improved to speed up the overall computational process for handling large-scale problems. In addition, introducing a semi-supervised learning mechanism to the proposed MLOP approach can be very helpful considering the very small ratios between labelled and unlabelled samples in many real-world applications. Finally, as aforementioned, it will be very useful to further improve the end-user explainability of the proposed approach by explaining the importance of different attributes during decision-making, such that users can better understand the inner relationships between input attribute values and model outputs.

References

[1] P. P. Angelov and X. Gu, "Autonomous learning multi-model classifier of 0-order (ALMMo-0)," in IEEE International Conference on Evolving and Autonomous Intelligent Systems, 2017, pp. 1–7.

[2] P. P. Angelov, X. Gu, and J. Principe, "A generalized methodology for data analysis," IEEE Trans. Cybern., vol. 48, no. 10, pp. 2981–2993, 2018.

[3] D. Berthelot, N. Carlini, I. Goodfellow, A. Oliver, N. Papernot, and C. Raffel, "MixMatch: a holistic approach to semi-supervised learning," arXiv preprint arXiv:1911.09785, 2019.

[4] L. Breiman, "Random forests," Mach. Learn. Proc., vol. 45, no. 1, pp. 5–32, 2001.

[5] G. Cerruela-García, A. de Haro-García, J. P. P. Toledano, and N. García-Pedrajas, "Improving the combination of results in the ensembles of prototype selectors," Neural Networks, vol. 118, pp. 175–191, 2019.

[6] D. Chen, Q. Yang, J. Liu, and Z. Zeng, "Selective prototype-based learning on concept-drifting data streams," Inf. Sci. (Ny)., vol. 516, pp. 20–32, 2020.

[7] N. Cristianini and J. Shawe-Taylor, An introduction to support vector machines and other kernel-based learning methods. Cambridge: Cambridge University Press, 2000.

[8] P. Cunningham and S. J. Delany, "K-nearest neighbour classifiers," Mult. Classif. Syst., vol. 34, pp. 1–17, 2007.

[9] J. Feng, Y. Yu, and Z. H. Zhou, "Multi-layered gradient boosting decision trees," in Advances in Neural Information Processing Systems, 2018, pp. 3551–3561.

[10] A. Fernandez, F. Herrera, O. Cordon, M. Jose Del Jesus, and F. Marcelloni, "Evolutionary fuzzy systems for explainable artificial intelligence: why, when, what for, and where to?," IEEE Comput. Intell. Mag., vol. 14, no. 1, pp. 69–81, 2019.

[11] S. Gao, L. Duan, and I. W. Tsang, "DEFEATnet—a deep conventional image representation for image classification," IEEE Trans. Circuits Syst. Video Technol., vol. 26, no. 3, pp. 494–505, 2016.

[12] S. García, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy and empirical study," IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 3, pp. 417–435, 2012.

[13] I. F. J. Ghalyan, S. M. Chacko, and V. Kapila, "Simultaneous robustness against random initialization and optimal order selection in Bag-of-Words modeling," Pattern Recognit. Lett., vol. 116, pp. 135–142, 2018.

[14] I. F. J. Ghalyan, "Estimation of ergodicity limits of bag-of-words modeling for guaranteed stochastic convergence," Pattern Recognit., p. 107094, 2020.

[15] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6645–6649.

[16] X. Gu, P. P. Angelov, and J. C. Principe, "A method for autonomous data partitioning," Inf. Sci. (Ny)., vol. 460–461, pp. 65–82, 2018.

[17] X. Gu, P. P. Angelov, C. Zhang, and P. M. Atkinson, "A massively parallel deep rule-based ensemble classifier for remote sensing scenes," IEEE Geosci. Remote Sens. Lett., vol. 15, no. 3, pp. 345–349, 2018.

[18] X. Gu, P. Angelov, and H. J. Rong, "Local optimality of self-organising neuro-fuzzy inference systems," Inf. Sci. (Ny)., vol. 503, pp. 351–380, 2019.

[19] X. Gu and W. Ding, "A hierarchical prototype-based approach for classification," Inf. Sci. (Ny)., vol. 505, pp. 325–351, 2019.

[20] H. Hagras, "Toward human-understandable, explainable AI," Computer (Long. Beach. Calif)., vol. 51, no. 9, pp. 28–36, 2018.

[21] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," IEEE Trans. Syst. Man, Cybern. Part B Cybern., vol. 42, no. 2, pp. 513–529, 2012.

[22] T. M. Kodinariya and P. R. Makwana, "Review on determining number of cluster in K-means clustering," Int. J. Adv. Res. Comput. Sci. Manag. Stud., vol. 1, no. 6, pp. 2321–7782, 2013.

[23] T. Kohonen, “Learning vector quantization,” in Self-Organizing Maps, Berlin, Heidelberg: Springer, 1995, pp. 175–189.
[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Advances In Neural Information Processing Systems, 2012, pp. 1097–1105.
[25] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nat. Methods, vol. 13, no. 1, pp. 35–35, 2015.
[26] Q. Li, Q. Peng, and C. Yan, “Multiple VLAD encoding of CNNs for image classification,” Comput. Sci. Eng., vol. 20, no. 2, pp. 52–63, 2018.
[27] L. Lu, L. Di, and Y. Ye, “A decision-tree classifier for extracting transparent plastic-mulched Landcover from landsat-5 TM images,” IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 7, no. 11, pp. 4548–4558, 2014.
[28] W. Luo, J. Li, J. Yang, W. Xu, and J. Zhang, “Convolutional sparse autoencoders for image classification,” IEEE Trans. Neural Networks Learn. Syst., vol. 29, no. 7, pp. 3289–3294, 2018.
[29] A. Nguyen, J. Yosinski, and J. Clune, “Deep neural networks are easily fooled: high confidence predictions for unrecognizable images,” in IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 427–436.
[30] A. Oliva and A. Torralba, “Modeling the shape of the scene: a holistic representation of the spatial envelope,” Int. J. Comput. Vis., vol. 42, no. 3, pp. 145–175, 2001.
[31] Y. Pan, Y. Xia, Y. Song, and W. Cai, “Locality constrained encoding of frequency and spatial information for image classification,” Multimed. Tools Appl., vol. 77, no. 19, pp. 24891–24907, 2018.
[32] R. N. Patro, S. Subudhi, P. K. Biswal, and F. Dell’Acqua, “Dictionary-based classifiers for exploiting feature sequence information and their application to hyperspectral remotely sensed data,” Int. J. Remote Sens., vol. 40, no. 13, pp. 4996–5024, 2019.
[33] M. Qaraei, S. Abbaasi, and K. Ghiasi-Shirazi, “Randomized non-linear PCA networks,” Inf. Sci. (Ny)., vol. 545, pp. 241–253, 2021.
[34] H. J. Rong, N. Sundararajan, G. Bin Huang, and G. S. Zhao, “Extended sequential adaptive fuzzy inference system for classification problems,” Evol. Syst., vol. 2, no. 2, pp. 71–82, 2011.
[35] A. Sato and K. Yamada, “Generalized learning vector quantization,” in Advances in neural information processing systems, 1996, pp. 423–429.
[36] S. Z. Selim and M. A. Ismail, “K-means-type algorithms: a generalized convergence theorem and characterization of local optimality,” IEEE Trans. Pattern Anal. Mach. Intell., vol. PAMI-6, no. 1, pp. 81–87, 1984.
[37] J. Shao, F. Huang, Q. Yang, and G. Luo, “Robust prototype-based learning on data streams,” IEEE Trans. Knowl. Data Eng., vol. 30, no. 5, pp. 978–991, 2018.
[38] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in International Conference on Learning Representations, 2015, pp. 1–14.
[39] I. Škrjanc, J. Iglesias, A. Sanchis, D. Leite, E. Lughofer, and F. Gomide, “Evolving fuzzy and neuro-fuzzy approaches in clustering, regression, identification, and classification: a survey,” Inf. Sci. (Ny)., vol. 490, pp. 344–368, 2019.
[40] R. E. Wendell and A. P. Hurter Jr, “Minimization of a non-separable objective function subject to disjoint constraints,” Oper. Res., vol. 24, no. 4, pp. 643–657, 1976.
[41] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear spatial pyramid matching using sparse coding for image classification,” in IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1794–1801.
[42] X. Zhai, A. Oliver, A. Kolesnikov, and L. Beyer, “S4l: self-supervised semi-supervised learning,” in IEEE/CVF International Conference on Computer Vision, 2019, pp. 1476–1485.
[43] C. Zhang, J. Cheng, C. Li, and Q. Tian, “Image-Specific Classification with Local and Global Discriminations,” IEEE Trans. Neural Networks Learn. Syst., vol. 29, no. 9, pp. 4479–4486, 2018.
[44] C. Zhang, J. Cheng, and Q. Tian, “Incremental codebook adaptation for visual representation and categorization,” IEEE Trans. Cybern., vol. 48, no. 7, pp. 2012–2023, 2018.
[45] C. Zhang, J. Cheng, and Q. Tian, “Structured weak semantic space construction for visual categorization,” IEEE Trans. Neural Networks Learn. Syst., vol. 29, no. 8, pp. 3442–3451, 2018.
[46] C. Zhang, C. Li, D. Lu, J. Cheng, and Q. Tian, “Birds of a feather flock together: visual representation with scale and class consistency,” Inf. Sci. (Ny)., vol. 460–461, pp. 115–127, 2018.
[47] C. Zhang, G. Zhu, C. Liang, Y. Zhang, Q. Huang, and Q. Tian, “Image class prediction by joint object, context, and background modeling,” IEEE Trans. Circuits Syst. Video Technol., vol. 28, no. 2, pp. 428–438, 2018.
[48] B. Zhao, H. Xiong, J. Bian, Z. Guo, C. Z. Xu, and D. Dou, “COMO: widening deep neural networks with convolutional maxout,” IEEE Trans. Multimed., DOI: 0.1109/TMM.2020.3002614, 2020.

[49] Z. H. Zhou and J. Feng, “Deep forest: towards an alternative to deep neural networks,” in International Joint Conference on Artificial Intelligence, 2017, pp. 3553–3559.
[50] F. Zhuang et al., “A comprehensive survey on transfer learning,” Proc. IEEE, vol. 109, no. 1, pp. 43–76, 2020.