Research Article

Multi-label classification of line chart images using convolutional neural networks

Cem Kosemen[1] · Derya Birant[2]

Received: 15 February 2020 / Accepted: 11 June 2020 / Published online: 19 June 2020

Abstract
In this paper, we propose a new convolutional neural network (CNN) architecture to build a multi-label classifier that categorizes line chart images according to their characteristics. The class labels are organized in the form of trend property (increasing or decreasing) and functional property (linear or exponential). In the proposed method, the Canny edge detection technique is applied as a data preprocessing step to increase both the classification accuracy and training speed. In addition, two different multi-label solution approaches are compared: label powerset (LP) and binary relevance (BR) methods. The experimental studies show that the proposed LP-CNN model achieves 93.75% accuracy, while the BR-CNN model reaches 92.97% accuracy on the test set, which contains real-world line chart images. The aim of this study is to build an efficient classifier that can be used for many purposes, such as automatically captioning the chart images, providing recommendations, redesigning charts, organizing a collection of chart images and developing better search engines.

Keywords Line charts · Image classification · Multi-label classification · Convolutional neural networks · Deep learning · Machine learning

1 Introduction

Line charts are popular and preferable tools in practice to represent useful numerical data in documents due to their many advantages over textual representations such as better representing ideas, perceivable within a short time and staying for a long time in memory. As just like the old saying "a picture is worth a thousand words", they are extremely rich and valuable sources of information. Line charts provide an efficient way to monitor the progress, such that data can be examined in terms of descriptive scales (high, medium, low), fluctuation (the variation of the data points) and trend (increasing or decreasing). These charts have been used in a wide range of areas to visualize mathematical functions in a coordinate system, to overview statistics and to give a quick understanding of changes in the variables. Hence, line charts are frequently embedded objects in many different types of digital sources such as web pages, books, articles, reports, research papers, newspapers, magazines, blog posts, and presentation slides.

Most of the line charts are currently created to be human-understandable and they are not originally machine-readable since they are generally in raster image format. However, during the past decades, there is an increasing need in many applications for the machine's ability to classify and interpret insights presented in line chart images automatically. When the underlying information is not available as textual, it is necessary to extract the knowledge from a line chart image to utilize this knowledge for further process. The main challenge when dealing with classifying line charts automatically is the variant

✉ Derya Birant, derya@cs.deu.edu.tr; Cem Kosemen, cem.kosemen@bakircay.edu.tr | [1]Department of Computer Engineering, Izmir Bakircay University, Izmir, Turkey. [2]Department of Computer Engineering, Dokuz Eylul University, Izmir, Turkey.

visual appearance and structure of the images. Therefore, it is significantly difficult for machines to classify the line chart according to their characteristics automatically and understand the encoded information accurately. The field of machine learning addresses this problem by utilizing deep learning methods to extract knowledge from graphic images. Therefore, in this study, we used a *convolutional neural network* (CNN) method as a *deep learning* technique to classify line charts to provide valuable knowledge. This knowledge then will help in automatically captioning/tagging the chart images, semantic description of the line chart, providing recommendations, redesigning charts, organizing a collection of chart images and developing better search engines.

Since line charts have various different types, they can be classified from different points of view such as according to their periods, shapes or growth rates. In this study, we focus on the classification of line charts according to their trend properties (increasing or decreasing) and functional properties (linear or exponential). Since more than one class label will be assigned to a single line chart at the same time, this task is a multi-label classification problem. The case in which one category is assigned to each instance is called *classification*, while *multi-label classification* (MLC) is the case when many categories are simultaneously assigned to the same instance. In our MLC task, a CNN model is first built from labeled line chart images in the training set and then an unseen line chart image can be correctly categorized according to the model that best fits it.

The novelties and main contributions of this paper can be listed as follows. (1) It is the first study that classifies line chart images based on their trend (increasing or decreasing) and functional (linear or exponential) properties using the CNN method. (2) Previous researches on chart classification are primarily based on single-label classification, while multi-label classification on line charts is not well studied. (3) As a data preprocessing step, this paper proposes the application of the Canny edge detection method to increase the classification performance. (4) We also compared two different multi-label classification approaches, called *label powerset* (LP) and *binary relevance* (BR), with CNN on the task of line chart classification for the first time. The experimental studies showed that the proposed LP-CNN model achieved 93.75% accuracy on the test set, which contains real-world line chart images.

The rest of the paper is organized as follows. Section 2 briefly states the motivation behind this work, in other words, the benefits of this study. Section 3 explains the previous studies about chart classification. Section 4 describes the technical details about the proposed CNN model, data preprocessing method and multi-label classification approaches that were compared in this study. Section 5 describes the dataset and presents the performance results of the proposed CNN model for each multi-label classification method, named LP-CNN and BR-CNN. Finally, the conclusions and possible future works are discussed in Sect. 6.

2 Motivation

This study focuses on building an intelligent model that can learn visual and graphical features directly from line chart images and is able to automatically predict multi-label classes of line charts. The motivation behind this work (the benefits of this study) can be briefly stated as follows:

- The line chart classification model can be used to automatically captioning and tagging the chart images.
- Line chart classification can be an assistant to a recommendation engine. To provide a recommendation for users, it is first important to understand the pattern in the line chart image. Recommendation strategies can be effectively given by considering the information extracted from the chart image. Patterns recognized from charts can assist the recommendation engine in presenting an appropriate choice for users.
- Line charts provide a visual assessment of the relationship between two or more variables. However, they cannot be noticed by visually impaired users since they are in visual picture form. Hence, all people with visual impairments have difficulties in information access. They can use screen and document readers for the textual documents; however, these readers don't interpret the charts, only read the caption commonly given under the chart itself. The text below the line chart may not enough for the visually impaired users to imagine what the visualization actually represents. Our method can help people with visual impairments to understand the information given in the line chart. Hereafter, line chart images will be understood by the visually impaired.
- Nowadays, major search engines involve images in their search results. However, indexed content for line charts principally relies on the textual metadata, rather than the content of the chart image. The metadata generally doesn't include enough information about the chart it represents, hence the search engines may not find many useful results from a user query. Search engines may overlook many useful query results if they don't consider the actual information represented in these charts during the query process. Thence, enriching the indexing content for line chart images provides an additional dimension to search improvement.

- Information extracted from the line charts by our method can help in chart redesigning such as ignoring some unnecessary primitives (i.e. grid lines). Hence it can assist to improve the chart for more accurate perception.
- This study provides a solution for line charts that need to be readable and interpretable by computers. For example, it is possible to interpret a line graph even if for the year for which the data was not available.
- There is an increasing demand for intelligent document understanding, where chart interpretation is an important issue since line charts are frequently embedded objects in the documents. Line charts may contain significant information that is not mentioned in the text. This study focuses on predicting the structure of line charts, which is an essential stage of chart interpretation.
- The accuracy of text mining algorithms can be improved by associating graphical and textual information. For this purpose, understanding of a line chart image can be achieved by classifying its graphical information on both semantic and logical levels. Our approach can help to automatically comprehend the knowledge within a text document for text mining studies.
- Chart recognition is an area of research and as important as text recognition to understand the information within the document automatically. Our study provides transforming its visual contents into computer understandable values. It allows us to capture the meaning carried by the line chart image in a suitable way.

In this study, an intelligent model was developed to be able to automatically classify and interpret insights presented in line chart images. Briefly, such a classification would be useful for many purposes, such as automatically captioning/tagging the chart images, semantic description of the line chart, providing recommendations, redesigning charts, organizing a collection of chart images and developing better search engines.

3 Related works

Although textual information is still the main source of data, there has been a growing trend of introducing chart figures to provide information. Charts are widely used to present a huge amount of data, emphasize key points presented in the text, and illustrate trends or changes. An average business computer user generates tens of charts and plots each week [1], it means that a very huge amount of potentially useful chart images are available on information sources. Though it is possible to interpret information from a chart representation manually by humans, it may become impractical since this rapidly increasing availability of chart images. Therefore, automated methods based on computer devices (computer-based processing) are required to extract the information present in a chart image.

The previous researches on chart classification primarily focus on identifying different types of charts (i.e. bar, pie, line, radar). Mishchenko and Vassilieva [1] compared many different machine learning techniques for the classification of images by chart type, including naive Bayes (NB), J48 decision tree (DT), support vector machine (SVM), random forest (RF) and neural network. Prasad et al. [2] used SVM and image processing techniques for classifying chart images based on the spatial relationships and shapes of their primitives. They considered 5 chart categories: bar-chart, curve-plot, pie-chart, surface-plot, and scatter-plot. They tested their approach on 653 images and achieved 84.23% classification accuracy. Savva et al. [3] also used SVM to classify 10 different types of chart images: area graph, bar chart, curve plot, map, pareto chart, pie chart, radar plot, scatter plot, tables, and Venn diagram. Their work, named ReVision, achieved 80% accuracy on average for multi-class classification on a 2601 image corpus. Instead of SVM, Jung et al. [4] developed a system, named ChartSense, which used a deep learning technique to improve the accuracy rate of ReVision when classifying 10 different types of chart types.

Recently, the CNN technique to classify chart images has attracted increasing attention from researchers [5–9]. Amara et al. [5] presented a CNN architecture for classifying 11 different chart types and achieved 89.5% accuracy over 3377 images. Bajic et al. [6] used the VGG (Visual Geometry Group) model, which is one of the well-known CNN architectures, and achieved 81.67% accuracy for 10 chart categories on 541 test images. Another work [7] used the CNN technique to predict what type of chart a given image is representing (i.e. area, bar, line, pareto, pie or radar). Chagas et al. [8] also classified charts by their types using different CNN architectures (Resnet-50, VGG-19, and Inception-V3). According to their results, Resnet-50 performed the best result with an accuracy of 77.76% on the test dataset, which has 2683 chart images collected from Google. It showed that CNN outperformed the conventional methods (k-nearest neighbors, NB, RF, and SVM) when classifying chart images in terms of their types. Tang et al. [9] proposed an approach to classify charts by combining deep belief networks and deep convolutional networks. Their proposed approach achieved the accuracy of 75.4% on a 5-class chart dataset (bar, flow, line, scatter, and pie chart). Based on these recent previous studies, in our study, we also used CNN as a deep learning method, since

it has been proved to be a successful method for recognizing and classifying chart and graph images.

As shown in Table 1, our work differs from the previous works in many respects. First, while many studies [1–9] have been focused on the classification of chart images by type (i.e. area, bar, line, pie), our study was conducted to classify line charts according to their trend (increasing or decreasing) and functional (linear or exponential) properties. Second, in the literature, many studies [1–9] have been focused on *multi-class classification*, while a study [10] conducted the chart type classification as a *multi-instance classification* problem. Unlike these previous works, our paper presents an experimental study of *multi-label classification* (MLC) methods (LP and BR) and gives suggestions for MLC that are effective for automatic chart image interpretation applications.

In the literature, while a wide range of studies [1–11] on chart classification cover various types of charts (i.e. area, bar, map, line, pie), several studies only cover a single chart type such as only pie chart [12], line chart [13] or radar chart [14]. In the study [12], a region-based convolutional neural network was used to automatically determine the type of pie chart (2D or 3D pie chart) depicted in a given image. Takagi and Chen [13] focused on classifying broken lines in the charts as dotted lines or chain lines. Liu et al. [14] proposed a classification technique that expresses multi-dimensional data with radar chart.

Classifying chart patterns is a crucial task; hence it has been required to be used in many different areas, including manufacturing [15], industrial engineering [16], finance [17] and civil engineering [18]. Lesanya et al. [16] used the neural network technique to automatically classify control chart patterns as Downward trend, Upward trend, Downward shift, Upward shift, Cycle, and Systematic. Wan and Si [17] proposed a rule-based method to classify chart patterns in financial time series, such as "Triple Tops", "Cup with Handle", and "Head-and-Shoulders". In [18], the charts representing the variation of Q index was used to classify the shale from "as good" to "very good" category.

The problem of understanding and interpreting charts has been addressed in various studies [19, 20]. However, instead of the classification task, they only used image processing (i.e. edge detection, segmentation, and feature extraction) and text recognition techniques [i.e. optical character recognition (OCR)]. Other similar studies [3, 21] combined both textual information and classification output to capture the semantic meaning of the charts. Text components such as caption, axis title, legend, and data value are firstly located in the chart image and then recognized using OCR. Unlike these studies, we didn't use textual information when interpreting a line chart, since the underlying information is not generally available as textual in many applications and domains. In this study,

Table 1 Comparison of our study with the previous studies

Refs.	Years	# Class	Chart type										Algorithms			Multi-class	Multi-instance	Multi-label	Accuracy (%)
			Area	Bar	Line	Map	Pareto	Pie	Radar	Scatter	Table	Venn	Others	CNN	SVM	Others			
[6]	2019	10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓			✓		81.67
[8]	2018	10	✓	✓	✓			✓		✓			✓	✓	✓	✓	✓		77.76
[4]	2017	10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓			✓		91.30
[5]	2017	11	✓	✓	✓	✓				✓			✓	✓			✓		89.50
[7]	2017	10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓		70.00
[9]	2016	5		✓	✓			✓		✓			✓	✓	✓	✓	✓		75.40
[1]	2011	5	✓	✓	✓			✓					✓		✓	✓	✓		90.00
[3]	2011	10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓		80.00
[2]	2007	5		✓				✓		✓			✓		✓		✓		84.23
[10]	2007	5		✓	✓			✓					✓			✓		✓	76.76
Our study		4			✓									✓				✓	93.75

we used machine learning techniques which have been proven to be useful in many fields ranging from industrial applications [16] to localization problems [22].

4 Materials and methods

4.1 Multi-label classification

Standard *single-label classification* is the task of learning from a collection of instances that are assigned with exactly one label. If there are only two class labels, the learning problem is called as a *binary classification* problem, whereas, if there are more than two class labels, it is then called as a *multi-class classification* problem. Conversely, *multi-label classification* is a concept of learning from a collection of instances where each instance is associated with several labels, meaning that an instance can belong simultaneously to one or multiple classes.

Let $X = R^d$ be the d-dimensional input feature vector and $Y = \{0, 1\}^{|L|}$ be the target output vector with $|L|$ possible labels such that $L = \{l_1, l_2, \ldots, l_s\}$. Giving a training dataset $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ that contains n instances, where each instance $x_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$ is a d-dimensional vector and $y_i = [y_{i1}, y_{i2}, \ldots, y_{i|L|}]$ is the label vector of x_i, where y_{ij} is 1 if x_i has the j-th label and 0 otherwise. The goal is to build a multi-label classifier function $f: X \rightarrow Y$ that optimizes evaluation metric(s) and can predict the label vectors for unseen instances.

In the literature, many successful multi-label methods used the problem transformation approaches. Problem transformation approaches once convert the multi-label classification task into a multi-class classification task or several binary classification tasks; after that, apply conventional classification algorithms to train them. Two of the most common methods for solving a multi-label classification problem are label powerset and binary relevance methods. The *label powerset* method transforms a multi-label dataset into a multi-class dataset, whereas *binary relevance* decomposes a multi-label dataset into several binary datasets (one for each label). In this study, we compared these two methods to determine which method works better for line chart classification.

4.1.1 Label powerset method

The *label powerset* (LP) approach transforms a multi-label dataset into a multi-class dataset by considering each combination of labels in the dataset as if it were a new single label [23]. In other words, the set of labels for each instance is combined as a single label, which is the concatenation of all the labels associated with this instance. After that, a multi-class classifier is constructed, and then an unseen observation is assigned to one of those combined labels.

Assume that L^+ is label combinations, including each possible combination of multiple labels in the training dataset as a new label, $|L^+| \leq 2^{|L|}$. For the ith instance x_i, we transform the original label vector $y_i = [y_{i1}, y_{i2}, \ldots, y_{i|L|}]$ into an L-dimensional vector $\hat{y}_i$. If the original label belongs to the class y_k, the kth component of the new label combination is assigned to 1, otherwise 0. Each (x, y) pair in the multi-label training set is transformed into $(x, \hat{y})$, and so, the corresponding multi-class training dataset is represented as $\{(x_1, \hat{y}_1), \ldots, (x_i, \hat{y}_i), \ldots, (x_n, \hat{y}_n)\}$.

The number of combined labels in LP is upper bounded by $2^{|L|}$, where $|L|$ is the number of labels. When the number of labels in the training dataset is high, and the data size is large, this may lead to computational complexity. Moreover, some combined labels can have a few training samples, so the resultant dataset can become imbalanced and that may negatively affect the classification performance. One possible solution for this problem is to prune the infrequent combined labels; although this process may improve the accuracy, the dataset may lose some of its multi-label structure. Furthermore, the LP method is sensitive to the label combinations in the dataset; this means that it only learns the label combinations that are present in the dataset, which is a kind of over-fitting problem. Therefore, if the new observation, which has a new label combination not present in the training set, will be classified; the model will never be able to predict this new combination. These possible problems of LP are not a critical issue for our study since we have only two labels, the training set is balanced and has all possible label combinations.

4.1.2 Binary relevance method

Binary relevance (BR) is one of the commonly used multi-label classification approaches [23]. In the BR method, classifiers are trained based on one-against-all strategy; the multi-label problem is transformed into multiple binary classification problems equivalent to the number of labels. In other words, it deals with the multi-label classification problem by constructing one classifier for each class. After that, in the training phase, a single-label binary classification method is utilized to solve each subtask. In the prediction phase, each binary classifier estimates whether its class is relevant for a new unlabeled input observation or not, resulting in a set of relevant labels. Although the BR method is a straightforward and practical method, a common drawback of it is that it ignores the relationship between labels since each label is trained independently; however, it may not have importance in some multi-label learning problems.

In the BR method, the L-label task is decomposed into L independent binary sub-tasks, where the k-th sub-task is expressed as, $\{(x_1, y_{1k}), \ldots, (x_i, y_{ik}), \ldots, (x_n, y_{nk})\}$. Hence, this method individually trains $|L|$ binary classifiers denoted by $C_1, C_2, \ldots, C_{|L|}$. Each classifier C_i is responsible for predicting the relevance of its corresponding label $l_i \in L$ by a 0/1 association such that $C_i: X \rightarrow \{0, 1\}$, where $i = 1, \ldots, |L|$. If an instance contains L_k, it is regarded as a positive instance "1", otherwise as a negative instance"0". Given an unseen observation, the binary predictions are combined to form a multi-label target. Hence, an unseen observation x_p is assigned the prediction $(C_1(x_p), C_2(x_p), \ldots, C_{|L|}(x_p))^T$. The computational complexity of BR is linearly dependent on $|L|$.

In this study, two multi-label problem transformation methods (LP and BR) were used to determine the better one for line chart classification. As shown in Table 2a, the class labels of line charts are organized in the form of trend (increasing or decreasing) and functional (linear or exponential) properties. Hence, each instance (line chart) is associated with multiple labels. Table 2b shows the tabular representation of the multi-class dataset transformed by the LP method. An instance that is assigned with class labels l_1 and l_2 would receive a single combined label l^{12}. For instance, if any data instance (line chart in this case) has both linear and increasing labels, it is considered as "linearly increasing". The combination of two labels would receive four unique combined labels: $L^+ = \{$linearly increasing (l^{11}), linearly decreasing (l^{01}), exponentially increasing (l^{10}), and exponentially decreasing (l^{00})$\}$. Table 2c shows the datasets generated by the BR method, one for trend label (increasing or not) and the other one for function label (linear or not).

4.2 Image preprocessing

The main challenge in line chart classification is to deal with the wild variety of chart styles in terms of structure, context, and visual appearance of the charts. *Structural variability* of line charts may be illustrated by "single" and "multiple" series charts or 2D and 3D-line charts, where they differ dramatically by their structure, but they are generally sensed as line charts by the human eye. *Context variability* includes variability of line chart surroundings, such as axes, legends, text regions, and grids. *Appearance variability* refers to the variability of colors, fill effects, and shadings used for the line chart. For the diverse appearance of line chart images, the following observations are generally true: there are text regions in the image; the background is sometimes complex; the line has lower contrast compared to other components in the image. Different line chart images have different and specific features, which are not common for natural scenes. One solution is to determine some constraints on acceptable chart images; however, it is not the desired solution for many domains. In addition, the existing images often have low

Table 2 (a) A tabular representation of a multi-label dataset with d features, n instances, and two labels. (b) The dataset transformed by the LP method, each combination of labels is translated into a new class. (c) The datasets generated by the BR method, one for each label

Input features				Labels		Input features				Label
F_1	F_2	…	F_d	L_1 (Increasing)	l_2 (Linear)	F_1	F_2	…	F_d	
(a) Multi-label dataset						(b) Multi-class dataset[a]				
x_{11}	x_{12}	…	x_{1d}	1	0	x_{11}	x_{12}	…	x_{1d}	l^{10}
x_{21}	x_{22}	…	x_{2d}	0	1	x_{21}	x_{22}	…	x_{2d}	l^{01}
x_{31}	x_{32}	…	x_{3d}	0	0	x_{31}	x_{32}	…	x_{3d}	l^{00}
…	…	…	…	…	…	…	…	…	…	…
x_{n1}	x_{n2}	…	x_{nd}	1	1	x_{n1}	x_{n2}	…	x_{nd}	l^{11}

Input features				Label l_1 (Increasing)	Input features				Label l_2 (Linear)
F_1	F_2	…	F_d		F_1	F_2	…	F_d	
(c) Binary datasets[b]									
x_{11}	x_{12}	…	x_{1d}	1	x_{11}	x_{12}	…	x_{1d}	0
x_{21}	x_{22}	…	x_{2d}	0	x_{21}	x_{22}	…	x_{2d}	1
x_{31}	x_{32}	…	x_{3d}	0	x_{31}	x_{32}	…	x_{3d}	0
…	…	…	…	…	…	…	…	…	…
x_{n1}	x_{n2}	…	x_{nd}	1	x_{n1}	x_{n2}	…	x_{nd}	1

[a]Dataset transformed by the LP method

[b]Datasets generated by the BR method

resolution and degraded by noise and blur. Thus the problem of line chart image classification requires an advanced approach. To overcome these challenges, we used several preprocessing techniques as the first step of our approach.

In our approach, first, line chart images are converted to greyscale because the colors of lines do not affect its shape. Colored images carry redundant data and increase training time. In a greyscale image, each pixel of the image is stored as a single value instead of three, which is the case in RGB images.

In real life, line chart images can be of any size. However, the images in the dataset should be in a fixed size for training them on CNNs. Because of that, images in the training and test sets are resized to 120×120 pixels while preserving their aspect ratio. We determined this size value based on trial and error tests. Images with sizes larger than 120×120 pixels did not increase test accuracy and caused much longer training times since the input size was increased. Smaller sizes than that also reduced classification accuracies.

After resizing images in a fixed size, the Canny edge detection method is applied to each of them for extracting characteristic features [24]. Canny edge detector algorithm is used to retrieve useful information from images. It also reduces the size of image data by discarding redundant information. Hence, CNN's training process becomes faster and better because it gets fewer data to work on and process. In this study, we used the OpenCV library [25] for Canny edge detection. Figures 1 and 2 show example line charts and their structures before and after applying Canny edge detection.

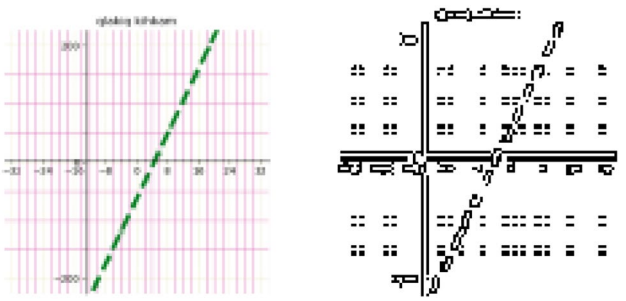


Fig. 1 Canny edge detection applied on a linearly increasing line chart

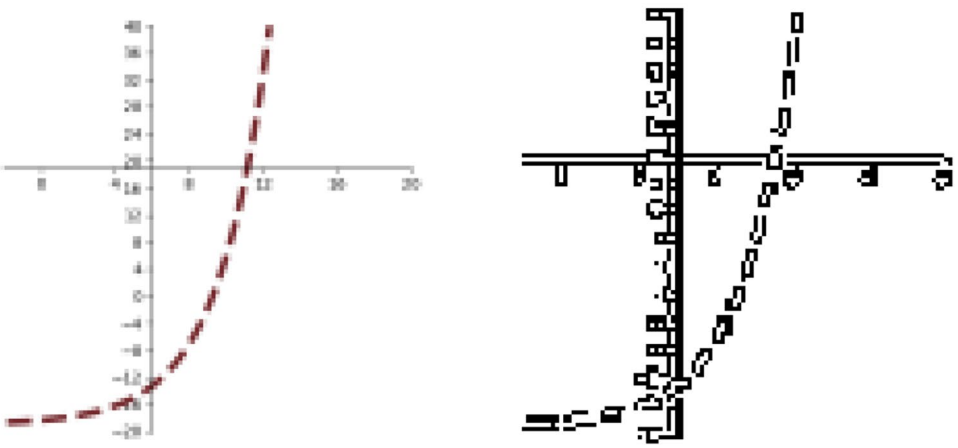


Fig. 2 Canny edge detection applied on an exponentially increasing line chart

4.3 Convolutional neural network

Convolutional Neural Networks (CNNs) are a special type of deep neural networks and image classification is one of the most common applications of this method [26]. The CNNs have drawn attention both in the high classification performance and in extracting information from the image.

Since images have large data size, giving them as an input to a deep neural network without extracting their distinctive features is not efficient. Convolution operations help to solve this problem by only feeding the neural network with useful features of an image [27]. Convolutions provide filter operations that are used in image processing methods like edge detection or noise reduction. In CNNs, distinctive features of images are extracted through convolution operations and pooling layers. CNNs learn the relevant filter kernels (matrices) for extracting the most distinctive features of the given images. Since the line chart images may have some noise and distortions, we chose CNN as the best solution for the line chart classification problem.

After convolution operation extracts the high-level features of the image, an actual neural network is fed with this output. This layer also called as a fully-connected layer or dense layer, and it results in a classification prediction. Since CNN extracted the useful features from the images, the fully-connected layer works with fewer data and achieves better accuracy results.

4.4 The proposed approach

Figure 3 illustrates the general structure of the proposed approach. In the first step, raw line chart images are preprocessed by recoloring, resizing and Canny edge detection methods and then fed into CNN architecture. CNN selects only useful features from all available ones by the feature extraction process. After that, in the training phase, the learning algorithm builds a good model that map inputs to correct outputs. The training process may be repeated with different parameters until a desired classification accuracy level achieved. Once a model is constructed, then it is utilized to predict the labels of an unseen line chart image.

The proposed methodology yields to work well since it has various advantages. First, the Canny edge detection technique is used as a data preprocessing step. Canny edge detector algorithm retrieves useful information from images and reduces the size of the image with discarding redundant information (i.e. unnecessary lines and noises). Hence, the training process becomes faster and better, so it can be possible to get higher accuracies in earlier epochs. Second, unlike the simple single label classifiers, the proposed method deals with a multi-label classification problem. Hence, it provides us a unified framework to collaboratively make several predictions. Third, the proposed method is a CNN-based method; hence, it automatically extracts information from images and has generally better performance on image classification, compared to the traditional classification methods such as decision tree, support vector machine, naive Bayes and k-nearest neighbors.

Chart image classification that has relied on simple features often fails when addressing the data that could include many varieties and less common line chart types. Therefore, in this study, we used CNN technique not only because it has achieved high classification performance in many image classification problems but also because it can be able to learn representations of images without designing a specific feature extractor. We propose to use the CNN approach since it automates the feature extraction step as a first step. Motivated by advances in deep learning techniques, which have been designed to produce considerable results in the field of image classification, a new CNN architecture is proposed for line chart analysis. We designed a simpler version of previous successful CNN models like VGG Net [6], Resnet [8] and AlexNet [28]. Our model uses 2D convolution layers and max-pooling layers followed by dropouts in the feature extraction process. The layer structure of the proposed CNN model is given in Table 3.

Padding is used to maintain the input and output dimensions. Our model applies padding in the first convolution layer to reduce data size from (120, 120, 32) to (118, 118, 32). Also, in other 2D convolution pairs, the first one reduces the data size, while the second 2D convolution layer does not change the image size. The same

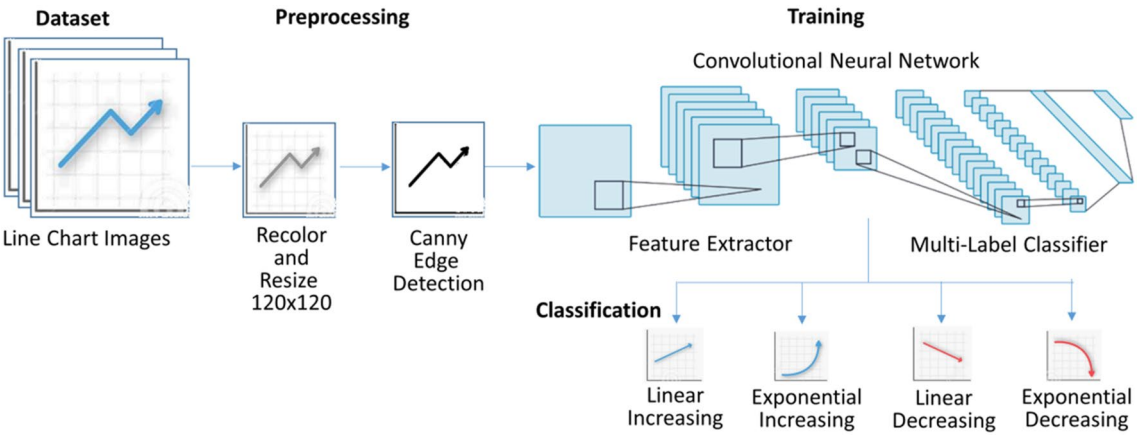


Fig. 3 The general structure of the proposed approach

padding fills the borders of images with zero values to prevent data loss when convolution operation applies.

Pooling layers in the convolution process are used for reducing the size of the images while keeping its distinctive features. The pooling process helps to avoid overfitting in CNNs by down-sampling the image data. If overfitting occurs, the resulting CNN works on training data well, but on test data poorly. We used the max-pooling method in our model as seen in Table 3.

The *dropout* method, which randomly drops some neurons in the network, prevents overfitting of CNNs and helps it to achieve higher test accuracies [29]. We used dropouts both after convolution operations and dense neuron layer. While dropout rates after max-pooling layers were defined as 25%, the dropout rate of 50% was applied after the dense layer.

The *loss* method basically compares the correct class and predicted class to return a score. As a loss function, a sparse categorical cross-entropy method was used in this study. This method is very popular when performing a classification task.

The *optimization* method is also needed in CNN models, which minimizes loss value, and it is important for the actual learning task. In this study, we selected Adam optimizer, which is a type of gradient descent algorithm [30].

The *rectified linear unit* (*ReLU*) activation function is commonly used in CNN models to achieve successful results [28]. In this study, ReLU was used with both convolution and dense layers. While, for negative inputs, ReLU outputs zero; for positive inputs, it returns the same input value as an output. ReLU's mathematical function is given in Eq. 1.

$$f(x) = max(0, x) \tag{1}$$

The *dense* (*fully-connected*) layer works as a multi-layer perceptron neural network. The output of the convolutional layer is flattened and then it becomes the input of the dense layer. Flattening operation converts a multi-dimensional data structure to a single-dimensional.

After the dense layer, *softmax* function [31] was used in this study, which is a useful and popular probability method and commonly used in classification tasks. Softmax function takes a real number value set and turns a probability for each value. The output shape of the softmax layer should be the same as the number of classes, which is 4 in this study. Each class has a probability value between 0 and 1, and they all sum up to 1. The mathematical formulation of the softmax function is given in Eq. 2

$$softmax\left(z_i\right) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{2}$$

where the function takes a vector of K real numbers as input and applies the exponential function to each element z_i of the input vector z and normalizes them by dividing by the sum of all these exponentials.

The pseudo-code of the proposed approach is presented in Algorithm 1. First, training and test sets are taken from the data repository. After that, two data preprocessing techniques (resizing and Canny edge detection) are applied to these datasets. The rest of the algorithm is mainly divided into two main parts: the implementation of the label powerset method and the binary relevance method. Each part individually contains data building, training, and prediction phases.

Table 3 Proposed CNN model

Layer type	Output shape	Trainable parameters
Conv2D	(120, 120, 32)	320
Conv2D	(118, 118, 32)	9248
MaxPooling2D	(59, 59, 32)	0
Dropout (25%)	(59, 59, 32)	0
Conv2D	(59, 59, 64)	18,496
Conv2D	(57, 57, 64)	36,928
MaxPooling2D	(28, 28, 64)	0
Dropout (25%)	(28, 28, 64)	0
Conv2D	(28, 28, 64)	36,928
Conv2D	(26, 26, 64)	36,928
MaxPooling2D	(13, 13, 64)	0
Dropout (25%)	(13, 13, 64)	0
Flatten	(10,816)	0
Dense	(512)	5,538,304
Dropout (50%)	(512)	0
Dense (Softmax)	(4)	2052

Algorithm 1 Multi-Label Classification of Images

Inputs:

D: the multi-labeled training image dataset $D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$
x_i: each instance $x_i = [x_{i1}, x_{i2}, ..., x_{id}]$ is a d-dimensional vector
y_i: each $y_i = [y_{i1}, y_{i2}, ..., y_{i|L|}]$ is the label vector
n: the number of images in the training set D
L: labels such that $L = \{l_1, l_2, ..., l_s\}$ and $|L|$ is the number of labels

Output:

Y: class predictions of un-labeled images in the test set

Begin:

```
D = LoadTrainingImages()      /* multi-labeled image dataset for training */
T = LoadTestImages()          /* un-labeled image dataset for testing*/
foreach image x in D and T    /* data preprocessing */
   ResizeImage(x)
   CannyEdgeDetection(x)
end foreach
Case: LabelPowerset           /* label powerset method */
   D_LP = {}
   for i = 1 to n do          /* build a new dataset containing label combination ŷ */
      for k=1 to |L| do
         ŷ_i = ŷ_i U y_ik     /* concatenation of all the labels associated with the current instance x_i */
      end for
      D_LP = D_LP U (x_i, ŷ_i)   /* D_LP = {(x_1, ŷ_1),...,(x_i, ŷ_i),...,(x_n, ŷ_n)} */
   end for
   C_LP = CNN(D_LP)           /* train classifier C_LP with D_LP using CNN */
   Y = {}
   foreach element t in T do  /* class prediction by the label powerset method */
      Y = Y U (y ← C_Lp (t))
   end foreach
   return Y
Case: BinaryRelevance         /* binary relevance method */
   for k=1 to |L| do
      D_k = {}
      for i = 1 to n do       /* build a new dataset considering only label y_k of D */
         D_k = D_k U (x_i, y_ik)   /* D_k = {(x_1, y_1k),...,(x_i, y_ik),...,(x_n, y_nk)} */
      end for
      C_k = CNN(D_k)          /* train classifier C_k with D_k using CNN */
   end for
   Y = {}
   foreach element t in T do  /* class prediction by the binary relevance method */
      for j = 1 to |L| do
         Y_t = Y_t U (y ← C_j (t))
      end for
      Y = Y U Y_t
   end foreach
   return Y
end case
```

End Algorithm

5 Experimental studies

In this study, different ways of addressing multi-label line chart classification problem are compared: label-powerset (LP) and binary relevance (BR).

5.1 Dataset description

In this study, the proposed model was used to discriminate between two labeled classes (trend and function), each of which has two features "increasing" and "decreasing", and "linear" and "exponential", respectively. Whereas, in the BR

method, separate datasets were generated for each label; in the LP method, the labels were transformed as; "linearly increasing", "linearly decreasing", "exponentially increasing" and "exponentially decreasing". The basic shapes of these line chart types are shown in Fig. 4.

A *linear chart* has a straight-line shape in its graphical representation. It increases or decreases with a constant rate of change. A linear function forms a linear chart if visualized, and the function has a mathematical form as indicated in Eq. 3. If the constant a is positive, the function is named as "linearly increasing"; whenever a has a negative value, the function is called as "linearly decreasing".

$$f(x) = ax + b \tag{3}$$

An *exponential chart* is structured as nonlinear and has curved lines. It has a mathematical form as indicated in Eq. 4. If the constant b is greater than 1, the function is labeled as "exponentially increasing"; whenever b has a value between 0 and 1, the function is called as "exponentially decreasing".

$$f(x) = ab^x \tag{4}$$

The training dataset should have a wide variety of line charts with different visual features, so the trained model can be able to make successful predictions on real-life images. Since a special image corpus is not available for this specific purpose, a computer-generated training dataset was used in this study. Here, we used Matplotlib plotting library in Python in order to generate the images in the training set. The markers and line styles were determined randomly by using a large number of different visualization methods available in Matplotlib. The widths of the lines were chosen randomly within a specified range. The other graphic features such as legend, grid style, value intervals in x and y axes were also chosen randomly.

With our plot generator Python script, we generated 400 line chart images for each class which makes 1600 training image samples in total. Since it contains an equal number of samples from each class, the training dataset is balanced. Several sample images from the generated training dataset are shown in Fig. 5.

Test data used in this work was collected from image search engines on the Internet, mainly from Google. In total, we gathered 320 test images, containing 80 images from each line chart class. The only constraint for selecting

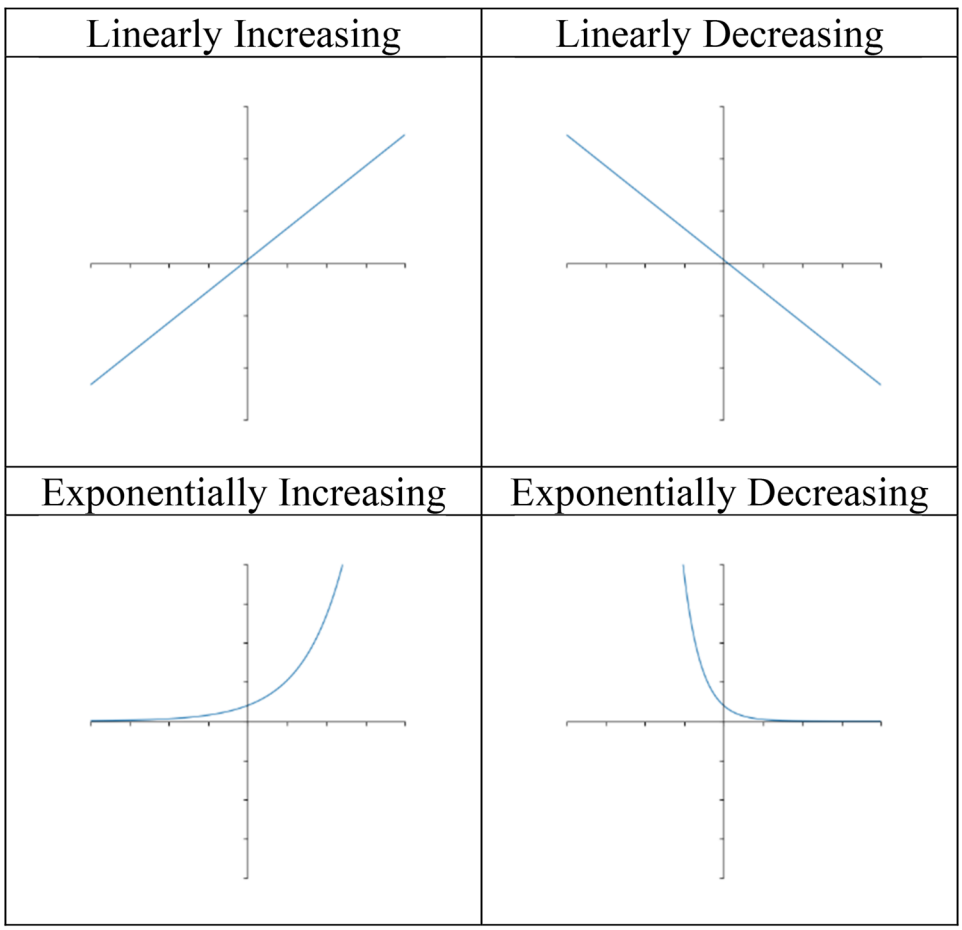


Fig. 4 Basic shapes of line charts

these test images is that the image should contain a single line series with a coordinate system. However, they can be in any size or they might have different line styles. Several sample images from the test dataset are given in Fig. 6. An additional validation set was not used on this work since such an approach would require a large dataset to be split further to generate a validation set. This would either result in a smaller test set or an even smaller training set.

5.2 Experimental settings

In this study, both LP and BR multi-label classification approaches were tested with our CNN model on the same dataset. From here onwards, the abbreviation of the multi-label method followed by the abbreviation of the learning technique is used to refer to the related approach. For example, LP-CNN refers to the LP method with the learning technique CNN. Each multi-label approach and learning technique is treated as an independent classifier. In other words, LP-CNN and BR-CNN are different multi-label classifiers.

The LP-CNN and BR-CNN methods were evaluated by means of line chart classification in terms of classification performance. These methods were explored and compared to each other. When evaluating the classification performance of the methods, we considered the most commonly used metrics such as train and test accuracies, precision, recall, and F1-Score. *Accuracy* means how well the model predicts given input data. While the *train accuracy* metric is the accuracy of the CNN model over the training set, the *test accuracy* metric is calculated on the test set. Train loss and test loss values are of loss function outputs over the training and test sets. *Precision* is the proportion of correct results in all the returned results. *Recall*, also called sensitivity, is the proportion of the correct predictions to the total number of correct results that could have been returned. *F1-Score* is the harmonic mean of precision and recall. All formulas of these metrics are given in the equations Eqs. 5–8, respectively.

$$Accuracy = \frac{true\ positives + true\ negatives}{total\ data} \tag{5}$$

$$Precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{6}$$

$$Recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{7}$$

$$F1\ Score = 2 \times \frac{precision \times recall}{precision + recall} \tag{8}$$

The proposed CNN model was implemented with Python programming language using Keras and TensorFlow frameworks. The Keras is a deep learning and neural network library that runs on top of the TensorFlow machine learning library. Keras also supports prototyping CNNs. In all experiments, a basic personal computer was used for training the CNNs and obtaining the performance results. This computer uses Windows 10 64-bit operating

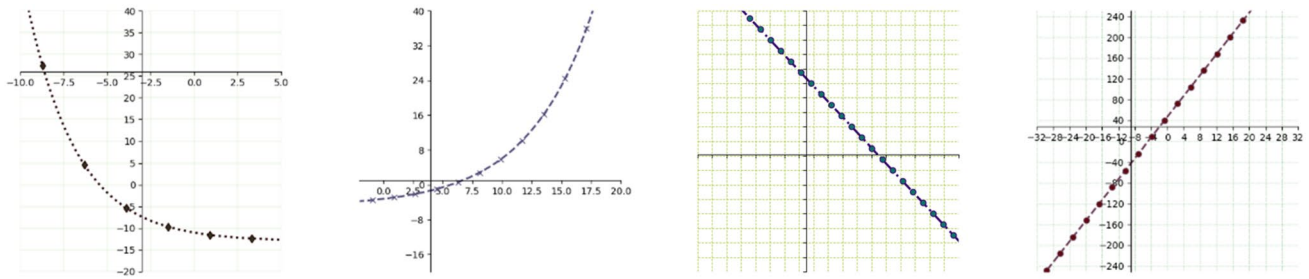


Fig. 5 Sample images from the training set

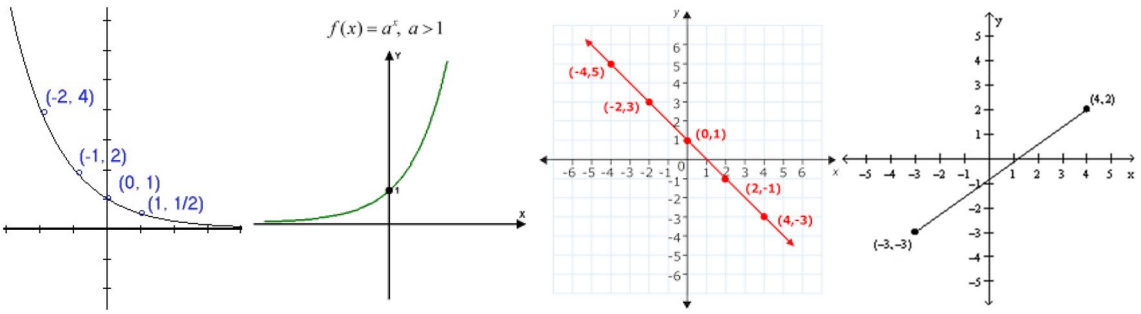


Fig. 6 Sample images from the test set

system with Intel Core i7-8750H 2.20 GHz processor and 16 GB of RAM.

Since the quality of a classification model critically depends on its hyperparameter configuration, we determined the optimum values for input parameters. The learning rate was set to 0.001 in our experiments. The batch size was the same as the number of training samples. Rather than limiting the training phase to a specific number of epochs, the training was set to expire when validation checks were reached. In order to prevent from overfitting problem, the validation checks were triggered according to the increases on the train and test accuracies and the decreases on the train and test error rates. Zero-padding was preferred over other alternative methods such as mirror-padding and linear-padding since these methods sometimes cause spurious effects in the matrices. Among alternative pooling techniques such as average or median pooling, we preferred max-pooling, since this improves convergence speed and also increases generalization due to position invariance over larger regions. While dropout rates after max-pooling layers were defined as 25%, the dropout rate of 50% was applied after the dense layer.

5.3 Experimental results

This paper introduces an extensive comparison of the multi-label classification approaches for line chart categorization.

5.3.1 Results of the LP-CNN method

The training process of the LP-CNN method across four combined labels reached its best test accuracy on the 45th epoch with a value of 93.75% as shown in Table 4. An *epoch* is defined as one training pass using all the training samples in the dataset. Thus, one epoch corresponds to all training samples being fed to the CNN. Training is performed on an epoch-by-epoch basis until the user-defined stopping criteria like the number of steps or targeted error minimization are satisfied. With respect to experimental results given in Table 4, it is possible to say that both test accuracy and train accuracy results reached very successful percentage values (> 93%).

Figure 7 shows the train and test accuracy and loss values obtained at the end of each epoch. Hence, the effect of epoch numbers is analyzed distinctly at each step. In this way, it can be easily determined the optimal epoch value by observing both train and test accuracies. The higher number of epochs usually results in high classification performance; when the accuracy reaches to its highest value, it remains the same or may drop due to overfitting.

Table 4 LP-CNN method classification performance results

Epoch	Train accuracy	Train loss	Test accuracy	Test loss
LP-CNN method				
45	0.998750	0.008740	0.937500	0.466359

The generalization performance of the model increases with increasing epoch until remains almost constant, as was generally noted in the previous studies. Train loss converges to zero as the number of epochs increases. After around 9 epochs, the training accuracy remains almost constant at about 99%, and training lost is very closely fixed to 0 value. It is observed from the experiments that the LP-CNN method is more easily stalled at flat regions during training. Train accuracy always shows better performance than the test accuracy, however, the two metrics usually follow each other closely. Initially, the test accuracy is continuously increased; after 5 epochs it achieves high values (about 90%) and then shows small fluctuations until it reaches the best accuracy of 93.75% at the 45th epoch. The key observation is the jump in the test losses, and a drop in the corresponding test accuracies. As the epoch progresses, the gap between the train loss and test loss sometimes increases, however, sometimes tends to be close. This instability is probably related to the variety of the structure, context, and visual appearance of the real-world line charts in the test set. For example, it may be relevant to the varieties of line chart surroundings such as legends, axes, text regions, and grids, or the variety of colors, fill effects, and shadings used for the line chart. The diversity of line chart images is sometimes observed when there are text regions in the image or background is complex or the line has a lower contrast compared to other components in the image.

The confusion matrix given in Table 5 shows the prediction performance of the LP-CNN method on the test set for each class individually. A *confusion matrix* has two-dimensions, the row dimension represents the actual classes of the objects, whereas the column dimension represents the classifier predicts. In the confusion matrix, the cell M_{ij} represents the number of examples actually belonging to class C_i, but that are classified as class C_j. It is observed from the confusion matrix that the model generally had no difficulty in identifying all classes. For instance, 76 of 80 "exponentially decreasing" line charts were predicted correctly; however, only 4 of them were labeled as "linearly increasing" by the constructed model, which are false predictions. According to the confusion matrix, three instances were classified incorrectly as an "exponentially increasing", instead of "linearly decreasing". It can be deduced from the confusion matrix that the best performance on the test set was achieved on "exponentially increasing" class detection

with accuracy 98.75%, where the classifier misclassified only one case for this class.

For assessing the LP-CNN method in more detail, Fig. 8 shows the precision, recall and F1-Score values obtained for each class on the test set. Higher numbers in this figure mean the LP-CNN method is more successful in the classification task for the corresponding class label. While the precision values are ranging from 0.8902 to 0.9863, the recall values are changing between 0.9 and 0.9875. Although both of them are very promising results with very high values; the recall scores are generally higher (so better) than the precision scores. For instance, the precision value for the "linearly increasing" class label is 0.8902 as a result of 73/82, where the value 73 is the correct prediction score for this class and 82 is the total count of predictions labeled as "linearly increasing". Likewise, the recall score for the same class is 0.9125 as a result of 73/80, where the values 73 and 80 are the number correct predictions and the number of instances for this class respectively. Among all class labels, LP-CNN achieved the best precision score (0.9863) on the "linearly decreasing" class label. When F1-Score value is close to 1, this means that the prediction result is close to ground truth. From this perspective, it is possible to say that the model built by LP-CNN has good generalization ability to variabilities in the input line charts, so it can be effectively used to predict them well.

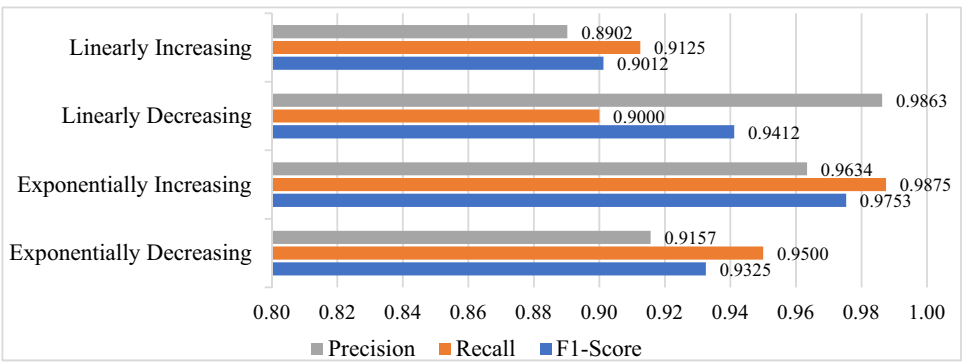
In addition to hold-out validation, the *fivefold cross-validation* technique was also performed to evaluate the performance of the LP-CNN method. In this technique, the dataset is divided into five disjoint subsets of almost equal

Fig. 7 LP-CNN learning process

Table 5 Confusion matrix of the LP-CNN model

Class	Predicted classes			
	Linearly increasing	Linearly decreasing	Exponentially increasing	Exponentially decreasing
Actual classes				
Linearly increasing	73 (91.25%)	0	0	7
Linearly decreasing	5	72 (90.00%)	3	0
Exponentially increasing	0	1	79 (98.75%)	0
Exponentially decreasing	4	0	0	76 (95.00%)

Fig. 8 Precision, recall and F1-Score values obtained from the LP-CNN model on the test set



size. In each fold, a subset is kept away, while the remaining subsets are used for training. The classifiers achieved the following accuracies at each fold respectively; 91.25% on 60th epoch, 90.62% on 90th epoch, 86.25% on 81st epoch, 90.62% on 60th epoch, and 88.44% on 49th epoch. The final result can be reported as 89.44%, which is the average of five runs.

5.3.2 Results of the BR-CNN method

In this experiment, the binary relevance multi-label classification approach was tested with the CNN model on the same test dataset. The BR-CNN method has two main steps: the first step determines whether the function type of the line chart is linear or exponential, and the second step predicts whether the values in the line chart are increasing or decreasing (trend). Hence, two classifiers, named as *function classifier* and *trend classifier*, are trained and then their outcomes are combined together to obtain the final prediction.

In the first part of the BR-CNN method, the function classifier was built by considering only one label attribute of the multi-label data as the target attribute. The linear line charts were labeled as 1, while others were assigned to 0. The constructed "function classifier" was tested on the real-world line chart images. As shown in Table 6, the trained model reached its best test accuracy (87.81%) on the test set at the 120th epoch.

In the second part of the BR-CNN method, the trend classifier was built by considering only related label attribute as the target. The line charts with increasing trend were labeled as 1, while others were assigned to 0. The constructed "trend classifier" was tested on the real-world line chart images. As shown in Table 7, the trained model reached its best test accuracy (98.13%) on the test set after 19 epochs.

The average test accuracy score of two classifiers (function classifier and trend classifier) was calculated as 92.97%, which is the final test accuracy output of the BR-CNN method.

Figures 9 and 10 show the train and test accuracy and loss values obtained at the end of each epoch during the learning processes of function and trend classifiers, respectively. Hence, the effect of epoch numbers is analyzed distinctly at each step. In this way, it can be easily determined the optimal epoch value by observing both train and test accuracies. Train loss converges to zero as the number of epochs increases. The generalization performances of the models improve with increasing epoch until remains almost constant. It is observed from the experiments that the BR-CNN method is more easily stalled at flat regions during the training of trend classifier. Train accuracy always shows a better performance than the test accuracy, however, the two metrics usually follow each other closely. Initially, the test accuracies are continuously increased; after a few epochs, they achieve high values and then show small fluctuations until they reach to their best accuracies (98.13% for the trend classifier and 87.81% for the function classifier). The key observation is the jump in the test losses. As the epoch progresses, the gap between the train loss and test loss sometimes increases; however, sometimes tends to be close. This instability is probably related to the variety of the structure, context, and visual appearance of the real-world line charts in the test set. For example, it may be relevant to the varieties of line chart surroundings such as legends, axes, text regions, and grids, or the variety of colors, fill effects, and shadings used for the line chart.

Since the BR-CNN method builds a separate classifier for each class label, Table 8 shows two confusion matrices, the first one for function classifier (linear or exponential) and the second one for trend classifier (increasing or decreasing). It is observed from the confusion matrices that the models generally had no difficulty in identifying all classes. For instance, 96.25% of increasing-trend line charts were predicted correctly; however, only 6 out of 160 charts were classified incorrectly. It can be deduced from the confusion matrices that the trend classifier has better performance on the test set, compared to the function classifier.

Figure 11 shows the precision, recall and F1-score values obtained by the BR-CNN method for each class on the test set. Higher numbers in this figure mean the BR-CNN method is more successful in the classification task for the corresponding class label. While the precision values are ranging from 0.9088 to 0.9530, the recall values are changing between 0.9031 and 0.9562. Although both of them are very promising results with very high values; the recall scores are higher (so better) than the precision scores for linear charts; while the opposite case is true for exponential charts. Among all class labels, the BR-CNN

Table 6 Function (linear or exponential) classifier performance results

Epoch	Train accuracy	Train loss	Test accuracy	Test loss
BR-CNN method				
120	1.000000	0.000015	0.878125	1.390342

Table 7 Trend (increasing or decreasing) classifier performance results

Epoch	Train accuracy	Train loss	Test accuracy	Test loss
BR-CNN method				
19	1.000000	0.000384	0.981250	0.874668

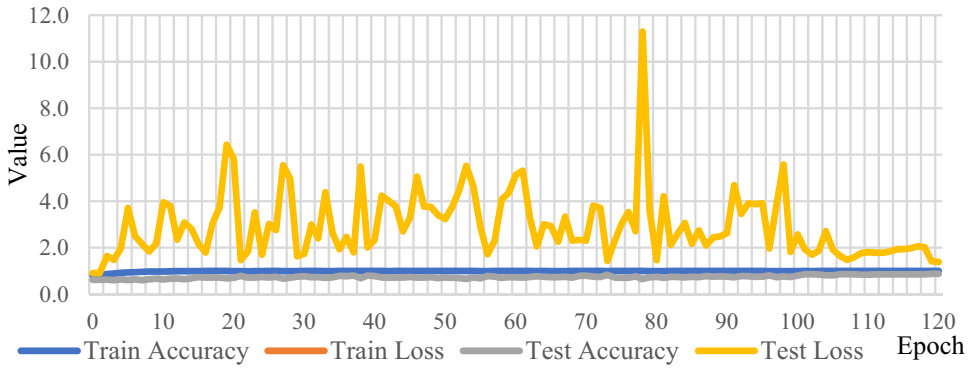


Fig. 9 BR-CNN learning process of function classifier (linear or exponential)

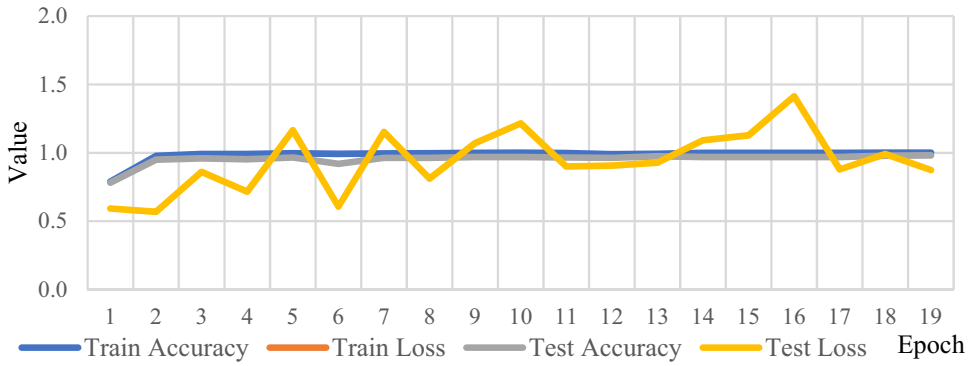


Fig. 10 BR-CNN learning process of trend classifier (increasing or decreasing)

Table 8 Confusion matrices of the BR-CNN models

Class	Predicted classes	
	Exponential	Linear
Actual classes		
Exponential	135 (84.38%)	25
Linear	14	146 (91.25%)
Class	Predicted classes	
	Increasing	Decreasing
Actual classes		
Increasing	154 (96.25%)	6
Decreasing	0	160 (100.00%)

method achieved the best score (0.9562) on the "exponentially increasing" class label. When F1-Score value is close to 1, this means that the prediction result is close to ground truth. From this perspective, it is possible to say that the model built by BR-CNN has good generalization ability (> 0.92) to variabilities in the input line charts, so it can be effectively used to predict them well.

5.3.3 Comparison of LP-CNN and BR-CNN Methods

Figure 12 shows the comparison of the LP-CNN and BR-CNN methods in terms of classification accuracy. Regarding the test accuracy results, LP-CNN (93.75%) slightly outperformed BR-CNN (92.97%) as illustrated in Fig. 12. This would suggest that the model built by LP-CNN has a better chance of being generalized beyond the training data. However, two methods can be alternatively used since the difference in the accuracy is small.

Furthermore, the LP-CNN and BR-CNN models were compared with the AlexNet [28] which is one of the most popular and efficient architectures that have been widely used to address problems in image classification. According to the results given in Fig. 12, it can be concluded that both LP-CNN and BR-CNN models outperformed

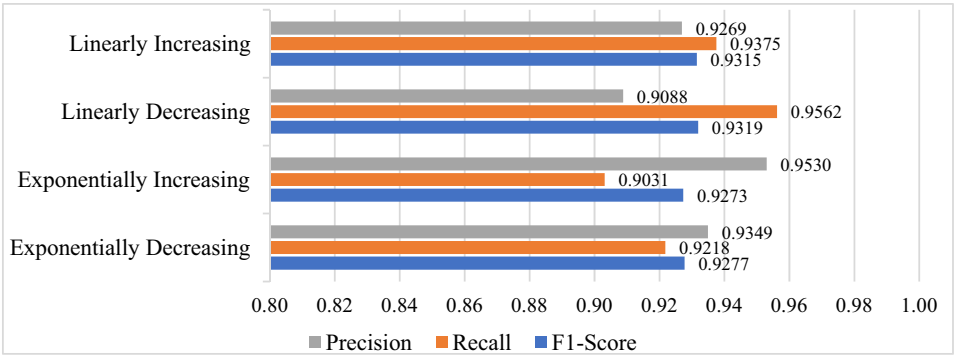


Fig. 11 Precision, recall and F1-Score values obtained from the BR-CNN model on the test set

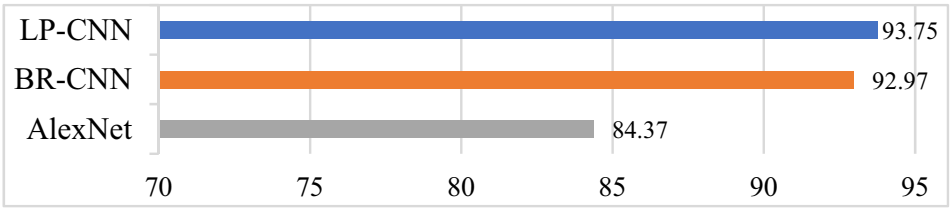


Fig. 12 Comparison of LP-CNN, BR-CNN, and AlexNet models in terms of classification accuracy

the AlexNet model in terms of classification accuracy. The results also show that the LP-CNN method approximately increased accuracy by 10% with respect to AlexNet in the experiment. While the LP-CNN method reached its best test accuracy (93.75%) on the 45th epoch, the AlexNet achieves its best performance (84.37%) on the 80th epoch. Thus, the experimental results indicate that the classifier built by the LP-CNN method can classify line chart images more effectively according to their trend and functional properties, compared to classifiers constructed by BR-CNN and AlexNet.

In addition to classification accuracy, we also compared the LP-CNN, BR-CNN, and AlexNet models in terms of F1-Score because of aggregating recall and precision into a single measure. Table 9 shows the F1-Scores obtained for each model and each multi-label class individually. In all cases, the LP-CNN and BR-CNN models provide better classification accuracy than the AlexNet model. On average, the LP-CNN model has the best performance with the 93.76% F1-Score value.

Although the LP-CNN method has higher F1-Scores, the obtained results should be validated by the statistical tests to ensure the significance of differences among the methods on the datasets. Here, the null hypothesis (H0) for the statistical test is that there are no performance differences among the methods on the datasets; otherwise, the alternative hypothesis (H1) is present when there are performance differences among the methods. The *p* value is defined as the probability under the null hypothesis of obtaining results and with a small *p* value (*p* value <= 0.05), we reject the null hypothesis (H0), so the relationship between the results is

Table 9 The F1-Scores of the CNN models

Dataset	F1-Scores (%)		
	LP-CNN	BR-CNN	AlexNet
Linearly increasing	90.12	93.15	73.85
Linearly decreasing	94.12	93.19	90.67
Exponentially increasing	97.53	92.73	81.28
Exponentially decreasing	93.25	92.77	90.17
Avg.	93.76	92.96	83.99

significantly different. For verification, we used three well-known non-parametric statistical tests by multiple group comparisons (all vs. all): Friedman Test [32], Friedman aligned ranks test [33] and Quade Test [34]. The *p* values obtained from these statistical tests are 0.00659, 0.04979 and 0.04930, respectively. Thereby, it is possible to say that the results are statistically significant since all *p* values are smaller than the significance level (0.05). Thus, it can be concluded that the differences between the performances of the examined methods are unlikely to occur by chance.

It should be noted that the classification accuracy is not the only issue in constructing a prediction model. The computational complexity should also be taken into consideration. The computational time of a multi-label problem can be evaluated in terms of the number of epochs. While the LP-CNN method reached the highest test accuracy value on the 45th epoch, the BR-CNN method achieved its best results after the 120 epochs. Increasing the epoch number will likely improve the classification accuracy of the model, but also increases the cost of time. Moreover, the consideration of

a large number of epochs usually involves the necessity to increase the required computing resources.

When the number of labels in the training set is low such as two, like in this study, the LP-CNN method can be usually run faster than the BR-CNN method. The reason behind this is that LP-CNN builds only one classifier, instead of individual classifiers for each class like BR-CNN. It deals with the multi-label problem as a multi-class problem, by concatenating the multiple labels that present in the dataset as one combined-label when constructing the classification model. Meanwhile, the BR-CNN method is slightly slower, since it should construct a collection of binary classifiers (one for each label) to deal with a multi-label problem. Increasing the number of training process increases the execution time of the method. However, this advantage of the LP-CNN method can be lost when the number of labels $|L|$ in the training set is high, since, in this case, the number of the combined labels is upper bounded by $2^{|L|}$ and this may lead to computational complexity.

It can be deduced from the experiments that higher classification rates can be achieved with the LP-CNN method for the line chart classification problem, rather than the BR-CNN method. The reason behind this is that the BR-CNN method ignores the relation between labels since each label is trained independently. However, the achievement of LP-CNN over BR-CNN can be changed depending on many factors, such as the number of labels, the application domain, and the structure underlying the training set.

6 Conclusion and future works

As an effective information transmitting way, line charts are widely used in many different sources (i.e. books, research papers, reports, newspapers) to visualize mathematical functions, to represent statistics datum and to gain a better understanding of changes. Line charts can provide additional information, may not be mentioned in the text, to allow readers to make their own inferences. However, the information given in line charts cannot automatically interpretable by the machines since they are generally in raster image format. Therefore, an intelligent model should be developed to be able to automatically classify and interpret insights presented in line chart images. Based on this motivation, in this study, we fundamentally focus on line chart classification which is formed as a multi-label classification task. Multi-label classification is concerned with a collection of training samples where each sample is assigned with several labels. In this study, the class labels are organized in the form of trend property (increasing or decreasing) and functional property (linear or exponential). We selected these properties because an essential stage in line chart image interpretation is to identify the trend and functional properties. However, classifying line chart images is a difficult task itself, due to the wild variety of chart styles.

In this paper, we propose a convolutional neural network (CNN) architecture to build a multi-label classifier that categorizes line chart images according to their characteristics. As a data preprocessing step, the Canny edge detection method was applied to decrease the training time and prevent the overfitting of the learning process. In addition, this paper presents the comparison of two multi-label classification approaches for line chart categorization: label powerset (LP) and binary relevance (BR) approaches.

In the experimental studies, the LP-CNN and BR-CNN methods were compared in terms of many different performance evaluation metrics such as accuracy, precision, recall, and F1-Score. While the training set consists of computer-generated line chart images, the benchmark test set is a collection of real-world line chart images, mainly obtained via Google image search. Both training and test sets are exactly balanced, so we have the same number of line chart images for each class. The experimental studies showed that the proposed LP-CNN model achieved 93.75% accuracy on the test set, which contains real-world line chart images. Hence, the constructed model can be effectively employed for classifying multi-labeled line chart images based on their characteristics with high accuracies.

In the future, more effort may be put to handle more complex line charts such as 3D charts and multiple series charts. In addition to linear and exponential shaped line charts, other different forms may be added, i.e. sinusoidal shaped charts. Line charts with changing trends, for example, a linear chart starting with an increase followed by a decrease, can also be inspected. In addition to LP and BR, other existing multi-label classification approaches such as classifier chains can be applied to extend the work. Furthermore, more attention can be given to the data preprocessing step, such as employing feature selection and feature weighting strategy.

Code availability Not applicable.

Compliance with ethical standards

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Availability of data and materials Not applicable.

References

1. Mishchenko A, Vassilieva N (2011) Model-based chart image classification. In: International symposium on visual computing (ISVC 2011), advances in visual computing. Lecture notes in computer science, vol 6939, pp 476–485. https://doi.org/10.1007/978-3-642-24031-7_48
2. Prasad VSN, Siddiquie B, Golbeck J, Davis LS (2007) Classifying computer generated charts. In: International workshop on content-based multimedia indexing (CBMI), pp 85–92. https://doi.org/10.1109/CBMI.2007.385396
3. Savva M, Kong N, Chhajta A, Li FF, Agrawala M, Heer J (2011) ReVision: automated classification, analysis and redesign of chart images. In: 24th annual ACM symposium on user interface software and technology (UIST'11), pp 393–402. https://doi.org/10.1145/2047196.2047247
4. Jung D et al (2017) ChartSense: interactive data extraction from chart images. Conf Hum Factors Comput Syst. https://doi.org/10.1145/3025453.3025957
5. Amara J, Kaur P, Owonibi M, Bouaziz B (2017) Convolutional neural network based chart image classification. In: 25th international conference in central europe on computer graphics, pp 83–88
6. Bajic F, Job J, Nenadic K (2019) Chart classification using simplified VGG model. In: Proceedings of the international conference on systems, signals and image processing (IWSSIP), 5–7 June 2019, Osijek, pp 229–233. https://doi.org/10.1109/IWSSIP.2019.8787299
7. Chagas P et al (2017) Architecture proposal for data extraction of chart images using convolutional neural network. In: Proceedings of the 21st international conference on information visualisation, 11–14 July 2017, London, pp 318–323. https://doi.org/10.1109/iV.2017.37
8. Chagas P et al (2018) Evaluation of convolutional neural network architectures for chart image classification. In: Proceedings of the international joint conference on neural networks (IJCNN), 8–13 July 2018, Rio de Janeiro, pp 1–8. https://doi.org/10.1109/IJCNN.2018.8489315
9. Tang B et al (2016) DeepChart: combining deep convolutional networks and deep belief networks in chart classification. Signal Process 124:156–161. https://doi.org/10.1016/j.sigpro.2015.09.027
10. Huang W, Zong S, Tan CL (2007) Chart image classification using multiple-instance learning. In: IEEE workshop on applications of computer vision (WACV'07), 21–22 Feb 2007, Austin. https://doi.org/10.1109/WACV.2007.17
11. Liu X, Klabjan D, Bless PN (2019) Data extraction from charts via single deep neural network. https://arxiv.org/abs/1906.11906
12. De P (2018) Automatic data extraction from 2D and 3D pie chart images. In: IEEE 8th international advance computing conference (IACC 2018), 14–15 Dec 2018, Greater Noida, pp 20–25. https://doi.org/10.1109/IADCC.2018.8692104
13. Takagi N, Chen J (2013) A broken line classification method of mathematical graphs for automating translation into scalable vector graphic. In: IEEE 43rd international symposium on multiple-valued logic, 22–24 May 2013, Toyama, pp 71–76. https://doi.org/10.1109/ISMVL.2013.3
14. Liu WY, Wang BW, Yu JX, Li F, Wang SX, Hong WX (2008) Visualization classification method of multi-dimensional data based on radar chart mapping. In: International conference on machine learning and cybernetics, July 2008, pp 857–862. https://doi.org/10.1109/ICMLC.2008.4620524
15. Hachicha W, Ghorbel A (2012) A survey of control-chart pattern-recognition literature (1991–2010) based on a new conceptual classification scheme. Comput Ind Eng 63:204–222. https://doi.org/10.1016/j.cie.2012.03.002
16. Lesany SA, Koochakzadeh A, Fatemi Ghomi SMT (2014) Recognition and classification of single and concurrent unnatural patterns in control charts via neural networks and fitted line of samples. Int J Prod Res 52:1771–1786. https://doi.org/10.1080/00207543.2013.848483
17. Wan Y, Si YW (2017) A formal approach to chart patterns classification in financial time series. Inf Sci 411:151–175. https://doi.org/10.1016/j.ins.2017.05.028
18. Kumar ND, Singh RR, Ali F, Efray'im (2017) Development of classification charts for Q Index of shale from the parameters. In: Advances in laboratory testing and modelling of soils and shales (ATMSS 2017), pp 281–287. https://doi.org/10.1007/978-3-319-52773-4_32
19. Yao JX, Agrawala M (2013) Linelens: automatic data extraction from line charts. In: Visualization, UC Berkeley CS, 2013, pp 1–10
20. Al-Zaidy RA, Giles CL (2015) Automatic extraction of data from bar charts. In: Proceedings of the 8th international conference on knowledge capture (K-CAP 2015), Oct 07–10, 2015, Palisades. https://doi.org/10.1145/2815833.2816956
21. Mishchenko A, Vassilieva N (2011) Chart image understanding and numerical data extraction. In: 6th International conference on digital information management, 26–28 Sept 2011, Melbourn, pp 115–210. https://doi.org/10.1109/ICDIM.2011.6093320
22. Nagy A, Bigler T, Treytl A, Sauter T (2019) A radio-map clustering algorithm for RSS based localization using directional antennas. In: 15th IEEE international workshop on factory communication systems, 27–29 May 2019, Sundsvall. https://doi.org/10.1109/WFCS.2019.8757938
23. Spolaôr N, Cherman EA, Monard MC, Lee HD (2013) A comparison of multi-label feature selection methods using the problem transformation approach. Electron Notes Theor Comput Sci 292:135–151. https://doi.org/10.1016/j.entcs.2013.02.010
24. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 6:679–698. https://doi.org/10.1109/TPAMI.1986.4767851
25. Bradski G (2000) The OpenCV Library. Dr Dobb's J Soft Tools 25:120–125
26. Basha SHS, Dubey SR, Pulabaigari V, Mukherjee S (2020) Impact of fully connected layers on performance of convolutional neural networks for image classification. Neurocomputing 378:112–119. https://doi.org/10.1016/j.neucom.2019.10.008
27. Rawat W, Wang Z (2017) Deep convolutional neural networks for image classification: a comprehensive review. Neural Comput 29:2352–2449. https://doi.org/10.1162/neco_a_00990
28. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90. https://doi.org/10.1145/3065386
29. Park S, Kwak N (2017) Analysis on the dropout effect in convolutional neural networks. Lecture Notes in Comput Sci 10112:189–204. https://doi.org/10.1007/978-3-319-54184-6_12
30. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: 3rd international conference on learning representations (ICLR 2015), pp 1–15. http://arxiv.org/abs/1412.6980
31. Bridle JS (1990) Probabilistic interpretation of feedforward classification network outputs with relationships to statistical pattern recognition. Neurocomputing 68:227–236. https://doi.org/10.1007/978-3-642-76153-9_28
32. Friedman M (1937) The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J Am Stat Assoc 32(200):675–701. https://doi.org/10.1080/01621459.1937.10503522

33. Hodges JL, Lehmann EL (1962) Ranks methods for combination of independent experiments in analysis of variance. Ann Math Stat 33:482–497. https://doi.org/10.1007/978-1-4614-1412-4_35
34. Quade D (1979) Using weighted rankings in the analysis of complete blocks with additive block effects. J Am Stat Assoc 74:680–683. https://doi.org/10.2307/2286991