Pattern Recognition 114 (2021) 107859

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Joint stroke classification and text line grouping in online handwritten documents with edge pooling attention networks

Jun-Yu Ye[a,b], Yan-Ming Zhang[a], Qing Yang[a], Cheng-Lin Liu[a,b,c,*]

[a] National Laboratory of Pattern Recognition, Institute of Automation of Chinese Academy of Sciences, Beijing 100190, China
[b] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China
[c] CAS Center for Exellence of Brain Science and Intelligence Technology, Beijing 100190, China

ARTICLE INFO

Article history:
Received 10 July 2020
Revised 28 November 2020
Accepted 22 January 2021
Available online 2 February 2021

Keywords:
Online handwritten documents
Stroke classification
Text line grouping
Graph neural networks
Edge pooling attention networks

ABSTRACT

Stroke classification and text line grouping are important tasks in online handwritten document segmentation. In the past, the two tasks were usually performed using different models which are trained independently and perform sequentially. This cannot optimize the integration of contextual information and the system may suffer from error accumulation in stroke classification. In this paper, we propose a method for joint text/non-text stroke classification and text line grouping in online handwritten documents using attention based graph neural network. In our framework, the stroke classification and text line grouping problems are formulated as node classification and node clustering problems in a relational graph, which is constructed based on the temporal and spatial relationship between strokes. We propose a new graph network architecture, called *edge pooling attention network (EPAT)* to efficiently aggregate information between the features of neighboring nodes and edges. The proposed model is trained by multi-task learning with cross entropy loss for node classification and distance metric loss for node clustering. In experiments on two online handwritten document datasets IAMOnDo and Kondate, the proposed method is demonstrated effective, yielding superior performance in both stroke classification and text line grouping.

1. Introduction

Pen-based interfaces such as smart phones, tablets and electronic whiteboards have enabled the generation of various heterogeneous contents such as text, drawing and table forms freely on a large writing area. Such freely handwritten ink documents bring new challenges to automatic analysis and recognition. A primary goal of document analysis is to group strokes into different structural objects (drawings, math formulas, text lines, etc.), among which text lines are the most salient structure and the carrier of semantic contents [1].

In previous works, the stroke classification problem and text line grouping problem in ink documents were usually considered as two separate tasks. For stroke classification, many structured prediction based methods were proposed to exploit temporal and spatial contextual information [2–6]. For text line grouping, optimization based methods [1,7–9] and distance metric learning based methods [10] were proposed. The interaction between stroke classification and text line grouping problems was only considered in Zhou et al. [1], where temporal and spatial merge procedures were proposed to correct the errors produced in stroke classification. However, the stroke classification model and text line grouping model used there were based on different methods and learned independently. In this way, the exploitation of contextual information in two tasks is not optimized, and the system may suffer from error accumulation in stroke classification.

In this paper, we propose a unified framework to address the stroke classification and text line grouping problems simultaneously, so as to couple the two tasks and minimize the errors by maximizing the usage of involved features. We consider each stroke in an ink document as a node in a relational graph and the relationship between strokes is modeled in edges. On the graph, the stroke classification problem and text line grouping problem can be formulated as node classification problem and node clustering problem, respectively, and can be modeled jointly and solved simultaneously. Specifically, we build a relational graph for each document based on the temporal and spatial relationship between strokes. Unary and pairwise geometric features are extracted as the raw node features and edge features, which are fed into the proposed *edge pooling attention network (EPAT)* model to obtain the distributed node and edge features. The probability of strokes

* Corresponding author at: National Laboratory of Pattern Recognition, Institute of Automation of Chinese Academy of Sciences, Beijing 100190, China.

E-mail addresses: junyu.ye@nlpr.ia.ac.cn (J.-Y. Ye), ymzhang@nlpr.ia.ac.cn (Y.-M. Zhang), qyang@nlpr.ia.ac.cn (Q. Yang), liucl@nlpr.ia.ac.cn (C.-L. Liu).

https://doi.org/10.1016/j.patcog.2021.107859
0031-3203/© 2021 Elsevier Ltd. All rights reserved.

belonging to each class and the distance between strokes are predicted by different fully connected layers. The EPAT is trained end-to-end using multi-task learning strategy, where the node classification task is trained by the cross entropy loss and the text line grouping task is trained by an improved version of the distance metric learning loss used in Delaye and Lee [10]. In experiments on two public datasets of online handwritten documents, IAMOnDo and Kondate, the proposed method is demonstrated to promise both stroke classification and text line grouping, and outperform state-of-the-art methods.

The main contributions of this work are summarized as follows:

1. We present a unified framework to address stroke classification and text line grouping jointly using graph neural networks (GNNs). The stroke classification and text line grouping problems are formulated as node classification and node clustering problems in the same relational graph.
2. We propose a new graph network architecture, *edge pooling attention network (EPAT)*, to effectively aggregate information between neighboring node features and edge features. In this new architecture, edge features are leveraged to control the aggregation of node features with attention mechanism and updated by a newly designed edge pooling module to capture the dependency between adjacent edges.
3. Using the proposed framework, we achieve state-of-the-art performance in both stroke classification and text line grouping on two public datasets. We provide results of comprehensive evaluation metrics and conduct multiple ablation experiments to demonstrate the key design of our model.

The rest of the paper is organized as follows. Section 2 reviews related works in online stroke classification, text line grouping and GNN. Section 3 describes the details of the proposed method. Section 4 presents experimental results, and Section 5 draws concluding remarks.

2. Related work

2.1. Stroke classification

The grouping of strokes into different categories in online documents is a fundamental step in document analysis. These methods can be divided into two categories, isolated classification and contextual classification, according to whether contextual information are used or not. For brevity, we only review the contextual classification methods that most related to our work. Bishop et al. [11] first proposed to use a hidden Markov model (HMM) with neural networks for modeling the interactions between temporally successive strokes. Ye et al. [5] improved this model by replacing the HMM framework with conditional random field (CRF) and jointly learning all the parameters. Phan et al. [4] proposed an approach based on bidirectional long short term memory (BLSTM) and achieved the highest accuracy among the temporal context based methods. Zhou et al. [2] first proposed to use Markov random field (MRF) to exploit the spatial context. Delaye et al. [3] extended the contextual system and proposed to exploit temporal, spatial, intersecting, lateral and stroke continuation relation under the CRF framework. Ye et al. [6,12] proposed to use attention based graph neural networks enhanced with edge features to model the temporal and spatial contexts, which achieved state-of-the-art performance on the IAMOnDo dataset [13].

2.2. Text line grouping

Segmentation of text lines from online documents has received a lot of attention. Projection-based methods [14,15] were proposed to estimate the inter-line distance and group text lines based on the inter-line distance. Methods based on global optimization were proposed to overcome the dependence on inter-line distance estimation. Shilman et al. [7] proposed to use dynamic programming to over-segment the stroke sequence with a cost function reflecting the confidence that a given set of strokes belong to one word, and the text lines are grouped by merging pairs of stroke clusters in successive steps. Ye et al. [8] used gradient decent to minimize a global cost function integrating the likelihood of the resulting lines and the consistency of their configuration after initial segmentation obtained by dynamic programming. Zhou et al. [1] proposed to use string level minimum classification error (MCE) criterion to learn the parameters in stroke merging. Their system involves multiple steps (block grouping, pre-segmentation, temporal merge and spatial merge) for text line grouping and can correct merging errors caused by stroke classification errors, and achieved state-of-the-art performance on the Kondate dataset [16].

Grouping strokes or structure elements into text lines can be based on distance metric learning. In offline handwritten document segmentation, Yin et al. [17] proposed a metric learning method aimed to minimize the inter-line distance and maximize the intra-line distance. The distance learning problem was formulated as a convex optimization program to learn the distance metric between connected components that are adjacent spatially, and text line grouping was accomplished by minimum spanning tree (MST) clustering on the graph of connected components. Distance metric learning was also used in natural scene text detection [18,19] for grouping characters into text lines. Delaye et al. [10] transferred the idea in Yin et al. [18] to online handwritten documents and proposed the distance learning method to group strokes into text lines by single linkage agglomerative clustering algorithm. Their method can extend from text line grouping to general symbol segmentation.

2.3. Graph neural network

Graph neural network (GNN) is a kind of machine learning model which applies neural networks in graph. Some networks [20,21] based on recurrent update are not efficient for large scale learning. In recent years, the convolution operation has been popularly adopted in GNNs, following the great success of convolutional neural network (CNN) in deep learning. Bruna et al. [22] first proposed the spectral convolutional neural network (SCNN) to apply convolution in the graph from the spectral view. Defferrand et al. [23] improved the method in Bruna et al. [22] by applying Chebyshev polynomials to approximate the filters. Kipf et al. [24] proposed the graph convolutional networks (GCN) using only first order expansion of Chebyshev polynomials and normalizing the adjacency matrix in the layer. The convolution operation derived in Kipf and Welling [24] from spectral view is similar to the convolution derived from the spatial view [25,26] and it can be seen as the bridge between spectral and spatial view.

To better capture different influences brought by different neighbor nodes, Velickovic et al. [27] adopt the self attention mechanism [28] into graph learning and proposed the graph attention network (GAT). The introduction of self attention mechanism in GAT integrates the node feature information for aggregating neighborhood nodes and improves the performance of node classification tasks. For exploiting discrete edge features, Schlichtkrull et al. [29] proposed to learn one weight matrix per edge type in the graph convolution procedure, and Busbridge et al. [30] extended the idea to graph attention networks. For exploiting continuous edge features, Gong et al. [31] proposed a unified framework to update the node features and edge features simultaneously in one layer and used doubly stochastic normalization technique to control the norm of the edge features and stabilize the training process.

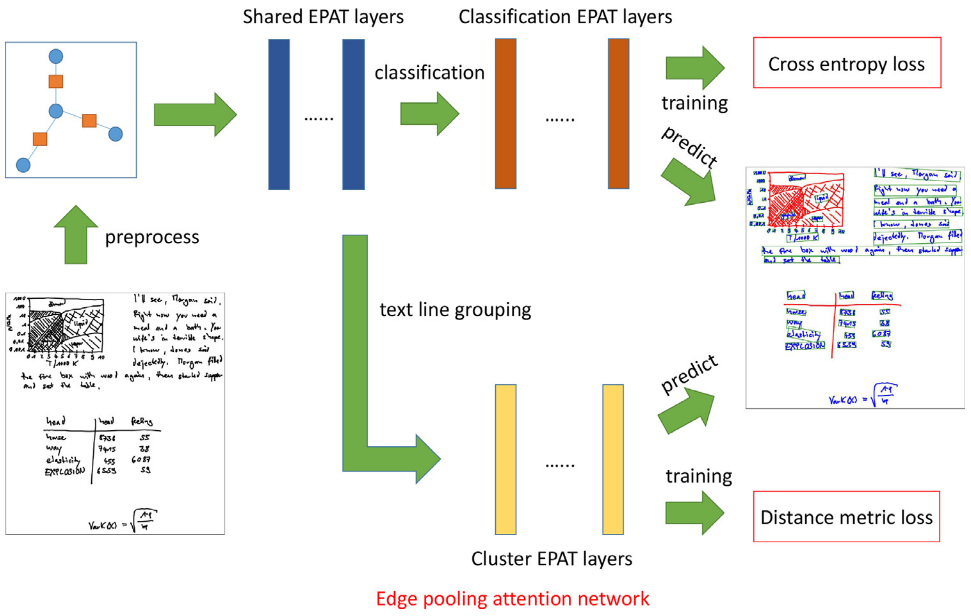

Fig. 1. An illustration of the training and prediction process of EPAT for stroke classification and text line grouping.

3. Proposed method

In this work, we formulate the stroke classification and text line grouping problem as node classification problem and node clustering problem, respectively, with learned edge weights in the graph. In the following, we first introduce the definition of stroke classification and text line grouping problems, then describe the pipeline of our model.

3.1. Problem definition

An online handwritten document consists of a set of strokes, each stroke as a sequence of pen-down samples points. A set of labeled online documents S are provided as the training set, in which each document $\mathbf{x}^{(i)}$ is formed by a sequence of strokes $\{\mathbf{x}_t^{(i)}, t = 1, \dots, T_i\}$ and each stroke has one associated label $y_t^{(i)} \in \{0, 1\}$. Besides, text lines in the ith document are denoted as $L^{(i)} = \{l_j^{(i)}, j = 1, \dots, m_i\}$, where $l_j^{(i)}$ is the index collection for the jth text line and m_i is the number of annotated text lines. The goal is to learn a model from the training set S that can predict stroke labels and group text lines in test documents.

3.2. Framework

Fig. 1 illustrates the framework of our model, which consists of three modules: construction of the relational graph, extraction of node and edge features, and edge pooling attention network (EPAT). An input document is first preprocessed to generate node features, edge features and the relational graph, then these components are fed into EPAT to learn context-aware node and edge features for stroke classification and text line grouping. During training, the model is learned by minimizing the cross entropy loss and distance metric loss. When performing prediction, the class with maximum probability is chosen for each stroke for stroke classification. Text lines are grouped on predicted text strokes according to learned distances.

3.2.1. Graph construction

In this work, an online handwritten document (a set of strokes) is represented as a relational graph $G(V, E)$, where each node $i \in V$ represents a stroke and each edge $(i, j) \in E$ represents the interaction between a pair of stroke (i, j). We follow the similar procedure to construct the relational graph as in our previous work [12] but improve in several aspects. Specifically, three different construction methods are considered in this work.

Temporal edges. The temporal relationship between strokes is the most important relationship in the online documents [4,5] so it is natural to build edges between strokes in the successive writing order. Specifically, the temporal edges are all pairs of strokes $(i, i+1) \dots (i, i+k_t)$, where k_t is the hyperparameter chosen by the validation.

Radius-based spatial edges. The effect of spatial relationship between strokes is also shown to be beneficial in stroke classification [2,3,6,12] and text line grouping [1]. In this part the spatial edges are added to the graph according to the minimum distance between strokes. If the minimum distance of two strokes u and v is less than a threshold k_r, the edge (u, v) is added to the graph. The threshold k_r is chosen with the following heuristic manner. First the minimum distance of each stroke to the nearest stroke in the same text line is calculated in the every text line in the training set, then median distance d_{median} of these distances is computed and the threshold k_r is set to $2d_{\text{median}}$.

K-nearest spatial edges. In ink documents, the large variability of stroke size and between-stroke distance can make the graph based on radius-based spatial edges miss some informative edges. We thus consider the k-nearest spatial edges as alternative way of graph consruction. Specifically, the nearest k_s spatial neighbors of each node will be added as spatial neighbors in the graph. The hyperparameter k_s is chosen by checking how many k-nearest spatial neighbors are sufficient to achieve high coverage rate of temporal neighbors (say, $\geq 98\%$) in the training set.

Last, a unified graph is constructed by unifying the three types of edges in one graph. In addition, self loop edges are added to the graph as self node features are important for aggregation in the message passing procedure of GNN.

3.2.2. Feature extraction

Each node and each edge of the relational graph are associated with some features describing their geometric properties. Geometric and local context features of each stroke are extracted as node features, and pairwise geometric features of pairs of strokes are extracted as edge features.

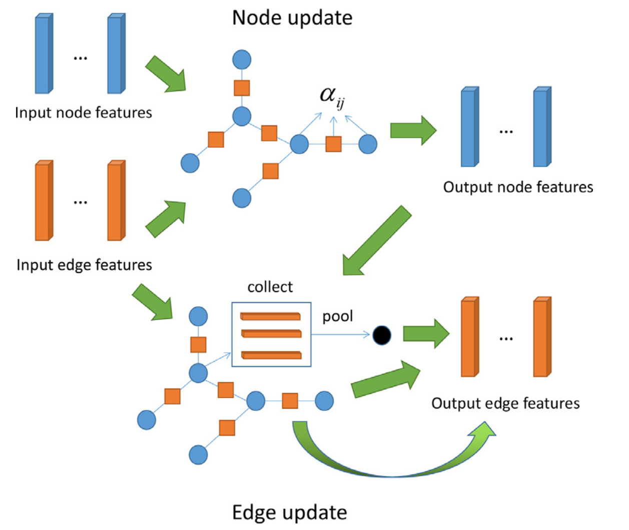

Fig. 2. An illustration of the computation procedure of the edge pooling attention (EPAT) layer.

Node features. 13 contour-based shape features and 10 local context features are extracted as node features following the similar procedure in Delaye and Liu [3], Van Phan and Nakagawa [4], Ye et al. [12]. The shape features are extracted from each stroke directly, while the context features are extracted by considering its interactions with neighboring strokes. Table A.8 lists all the node features.

Edge features. 33 pairwise stroke features are extracted as edge features in the graph for estimating the similarity between pair of strokes. These features are collected from the previous works [3,4,10] for stroke classification or text line grouping task in the ink document. Some features are also made symmetric between (u, v) and (v, u) pairs with the min-max trick as the relation between pairs of strokes should be the same. These features can be regarded as the indicators to measure similarity between two strokes with various distance metrics. Therefore, it is suitable to use these features to learn gates to control the feature aggregation procedure in the stroke classification task and learn appropriate distance for text line grouping. Table A.9 lists all the edge features (Fig. 2).

In the last step of feature extraction, power transform and Z-score normalization are applied to both the unary and pairwise features to standardize the feature values into the same scale as in Ye et al. [12].

3.2.3. Edge pooling attention network

The basic building block of the proposed edge pooling attention network is *edge pooling attention layer*. The proposed layer in this paper is an extension to the architecture in our previous work [12]. In this work, we propose the edge pooling technique in the edge update procedure for directly aggregating information between edges that share the same vertex, which is specifically designed for text line grouping task. The motivation behind that is whether a pair of strokes should be merged into same text line not only depends on the geometric relationship between them, but is also affected by the geometric relationship of strokes that are adjacent to them.

The *edge pooling attention layer* includes two steps to update the input node features and edge features. For completeness, we briefly review the node update procedure as in Ye et al. [12] and then introduce the new edge update procedure integrated with the edge pooling technique. The input to each edge pooling attention layer is a set of node features $\mathbf{H} = \{\mathbf{h}_i, i \in V\}, \mathbf{h}_i \in \mathbb{R}^C$ and a set of edge features $\mathbf{F} = \{\mathbf{f}_{ij}, (i, j) \in E\}, \mathbf{f}_{ij} \in \mathbb{R}^D$, where V is the set of nodes, E is the set of edges. The output of layer are new sets of node features $\mathbf{H}' = \{\mathbf{h}'_i, i \in V\}, \mathbf{h}'_i \in \mathbb{R}^{C'}$ and edge features $\mathbf{F}' = \{\mathbf{f}'_{ij}, (i, j) \in E\}, \mathbf{f}'_{ij} \in \mathbb{R}^{D'}$.

Node feature update. In the node update procedure, two types of attention mechanisms are applied to aggregate information between neighborhood nodes. The first one is the self attention mechanism [27,28]. A shared linear transformation is applied to every node by an attention mechanism $a : \mathbb{R}^{C'} \times \mathbb{R}^{C'} \to R$ to compute the node attention score:

$$s_{ij} = a(\mathbf{W}_h\mathbf{h}_i, \mathbf{W}_h\mathbf{h}_j),$$
$$a(\mathbf{W}_h\mathbf{h}_i, \mathbf{W}_h\mathbf{h}_j) = \sigma(\mathbf{v}^T(\mathbf{W}_h\mathbf{h}_i + \mathbf{W}_h\mathbf{h}_j)), \tag{1}$$

where $\mathbf{W}_h \in \mathbb{R}^{C \times C'}$, $\mathbf{v} \in \mathbb{R}^{C'}$ are learnable parameters. $\sigma(\cdot)$ is leaky ReLU activation function [32].

The second one is the edge attention mechanism that exploits the pairwise geometric features to learn the attention score, which is computed by

$$s'_{ij} = \sigma(\mathbf{v}_f^T\sigma(\mathbf{W}_f\mathbf{f}_{ij} + \mathbf{b}_f)), \tag{2}$$

where $\mathbf{W}_f \in \mathbb{R}^{C' \times D}, \mathbf{b}_f \in \mathbb{R}^{C'}, \mathbf{v}_f \in \mathbb{R}^{C'}$ are learnable parameters.

Two types of attention score are then summed to calculate the final attention score and normalized across the edges with temperature β. Then the output node features are computed as the weighted combination of the neighborhood node features with the attention scores:

$$\alpha_{ij} = \frac{\exp(\beta(s_{ij} + s'_{ij}))}{\sum_{k \in \mathcal{N}_i} \exp(\beta(s_{ik} + s'_{ik}))},$$
$$\mathbf{h}'_i = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}\mathbf{W}_h\mathbf{h}_j\right), \tag{3}$$

where $\mathcal{N}_i$ is the neighborhood of node i.

At last, the standard technique *multi-head attention* is also employed, which can be expressed as

$$\mathbf{h}'_i = \|_{k=1}^{K} \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}_h^k \mathbf{h}_j\right), \tag{4}$$

where $\|$ represents concatenation.

Edge feature update. After the update of the node features with feature aggregation of neighborhood nodes, we also like to update the edge features so as to contain enough information to decide whether two strokes should be merged. The information considered in this paper are from three views. First it should contain the information of itself:

$$\mathbf{r}_{ij}^e = \sigma(\mathbf{W}_{edge}\mathbf{f}_{ij}). \tag{5}$$

Second, the node feature should be included to update the edge feature as the node feature should contain the information of stroke type (text/non-text). Knowing this information is likely helpful for deciding whether two strokes should be merged as it is invalid to link a text stroke and a non-text stroke:

$$\mathbf{r}_{ij}^n = \sigma(\mathbf{W}_{node}[\mathbf{h}'_i \,\|\, \mathbf{h}'_j]). \tag{6}$$

Third, since the global information about the whole text line plays an important role in the text line grouping [1] and sometimes the pairwise geometric features are not sufficient to decide the merge of two strokes, it is beneficial to have a module for edge features to communicate directly with adjacent edge features. So we propose to use two common pooling techniques to first pool the edge features to the nodes and update the edge features with the pooling features:

$$\mathbf{t}_i^{\max} = \text{maxpool}_{(i,j) \in E}\ \{\mathbf{f}_{ij}\},$$
$$\mathbf{t}_i^{\text{ave}} = \text{avepool}_{(i,j) \in E}\ \{\mathbf{f}_{ij}\},$$

$$\begin{aligned} \mathbf{r}_{ij}^{\max} &= \sigma(\mathbf{W}_{max}[\mathbf{t}_i^{\max} \mid\mid \mathbf{t}_j^{\max}]), \\ \mathbf{r}_{ij}^{\text{ave}} &= \sigma(\mathbf{W}_{ave}[\mathbf{t}_i^{\text{ave}} \mid\mid \mathbf{t}_j^{\text{ave}}]). \end{aligned} \tag{7}$$

At last, another fully connected layer is used to project the concatenation of these features to the output features:

$$\mathbf{f}'_{ij} = \sigma(\mathbf{W}_{reduce}[\mathbf{r}_{ij}^{e} \mid\mid \mathbf{r}_{ij}^{n} \mid\mid \mathbf{r}_{ij}^{\max} \mid\mid \mathbf{r}_{ij}^{\text{ave}}]). \tag{8}$$

Some standard techniques including residual connection and batch normalization are also employed to stabilize and accelerate the training procedure. We set the same hyperparameters in all layers to the same to reduce the complexity of hyperparameter tuning. Because of that, the residual connection can be added except the first layer:

$$\begin{aligned} \mathbf{h}'_i &= \text{BatchNorm}(\mathbf{h}_i + \text{NodeUpdate}(\mathbf{h}_i)), \\ \mathbf{f}'_{ij} &= \text{BatchNorm}(\mathbf{f}_{ij} + \text{EdgeUpdate}(\mathbf{f}_{ij}, \mathbf{h}'_i, \mathbf{h}'_j)). \end{aligned} \tag{9}$$

After the design of appropriate layer for learning context-aware features, the next step is to design an appropriate architecture that can tackle with two tasks simultaneously. There are various architectures proposed to deal with the multi-task learning in recent years, like the MoE model [33] and MMoE model proposed in Ma et al. [34]. But in this work, we choose the traditional share-bottom model [35] as the multi-task learning framework as it is simple to implement and achieves the satisfactory performance. Specifically, a number of bottom layers are shared, then the features output by the last shared layer are fed into two different branches to get the different node and edge features for stroke classification and text line grouping, which can be expressed as

$$\begin{aligned} \mathbf{H}_{\text{share}}, \mathbf{F}_{\text{share}} &= \phi_{\text{share}}(\mathbf{H}_{\text{raw}}, \mathbf{F}_{\text{raw}}), \\ \mathbf{H}_{\text{cls}}, \mathbf{F}_{\text{cls}} &= \phi_{\text{cls}}(\mathbf{H}_{\text{share}}, \mathbf{F}_{\text{share}}), \\ \mathbf{H}_{\text{seg}}, \mathbf{F}_{\text{seg}} &= \phi_{\text{seg}}(\mathbf{H}_{\text{share}}, \mathbf{F}_{\text{share}}), \\ \mathbf{p}_i &= \text{softmax}(\mathbf{W}_{\text{cls}}\mathbf{h}_i^{\text{cls}} + \mathbf{b}_{\text{cls}}), \quad \forall i \in V \\ d(u,v) &= \mathbf{W}_{\text{seg}}\mathbf{f}_{ij}^{\text{seg}} + b_{\text{seg}}, \quad \forall (u,v) \in E \end{aligned} \tag{10}$$

where $\mathbf{H}_{\text{raw}}$ and $\mathbf{F}_{\text{raw}}$ are raw node and edge features, $\phi_{\text{share}}, \phi_{\text{cls}}, \phi_{\text{seg}}$ are three different branches which is stacked with EPAT layers. The output $\{\mathbf{p}_i \mid i \in V\}$ and $\{d(u,v) \mid (u,v) \in E\}$ are the correspodoning node probability for stroke classification and edge weights for text line grouping.

3.3. Inference

In the inference process, first the node probability of each stroke $\{\mathbf{p}_i \mid i \in V\}$ and distances $\{d(u,v) \mid (u,v) \in E\}$ are output according to (10). The prediction of stroke classification is obtained by choosing the class with maximum probability of each stroke. Then the subgraph containing only text strokes are selected to group into text lines according to distances. Algorithm 1 summarizes the inference process.

Algorithm 1 Inference process.

```
Input: Document sample $S = \{\mathbf{H}_{\text{raw}}, \mathbf{F}_{\text{raw}}, G\}$;
Output: Predicted stroke labels $\hat{\mathbf{y}} = \{\hat{y}_i \mid i \in V\}$ and text lines $\hat{L} = \{\hat{l}_j, j = 1, \ldots, \hat{m}\}$;
1: Compute $\{\mathbf{p}_i \mid i \in V\}$ and $\{d(u,v) \mid (u,v) \in E\}$ according to (10);
2: $\hat{\mathbf{y}} = \{\hat{y}_i = \arg\max \mathbf{p}_i \mid i \in V\}$;
3: Select the subgraph $G' = (V_{G'}, E_{G'})$ with $V_{G'} = \{v \mid \hat{y}_v = 1\}$ and $E_{G'} = \{(u,v) \mid u \in V_{G'}, v \in V_{G'}\}$.Initialize $\hat{L} = \{\hat{l}_j = \{j\} \mid j \in V_{G'}\}$;
4: for $(u,v) \in E_{G'}$ do
5:     if $d(u,v) \le 0$ then
6:         $\hat{l}_u \leftarrow \hat{l}_u \cup \hat{l}_v$;
7:         $\hat{L} \leftarrow \hat{L} \setminus \hat{l}_v$;
8: return $\hat{\mathbf{y}}$ and $\hat{L}$;
```

3.4. Network training

For training the EPAT, the loss functions and and training algorithm are introduced as follows.

3.4.1. Loss functions

For stroke classification, the loss function is the standard cross entropy loss. Given the output probability $\mathbf{p} \in \mathbb{R}^{T_i \times 2}$ and the ground truth label $\mathbf{y} \in \{0,1\}^{T_i}$ of one document,

$$Loss_{\text{node}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{2}\delta(y_t = j)\log p_t(j;\theta). \tag{11}$$

For text line grouping, a modified version of distance learning method in Delaye and Lee [10] is used. Given the annotated text line $L = \{l_j, j = 1, \ldots, m\}$ of one document and distances $\{d(u,v;\theta), (u,v) \in E\}$ output by the model, the edge with maximum distance within the same text line and the edge with minimum distance between different text lines can be found, which can be expressed as

$$\begin{aligned} \mathcal{M} &= \left\{(u,v) = \mathop{\arg\max}_{(u,v)\in E, u\in l_j, v\in l_j} d(u,v;\theta)\right\}_{j=1}^{m}, \\ \mathcal{C} &= \left\{(u,v) = \mathop{\arg\min}_{(u,v)\in E, u\in l_j, v\in l_k, k\neq j} d(u,v;\theta)\right\}_{j=1}^{m}. \end{aligned} \tag{12}$$

Both $\mathcal{M}$ and $\mathcal{C}$ contain m links. Assuming that the clusters match with the ground truth segmentation, a reasonable objective is to maximize the distance for links in $\mathcal{C}$ to encourage cluster separability, and to minimize the distance for links in $\mathcal{M}$ to encourage cluster compactness. This goal can be achieved by optimizing the following logistic loss of distances:

$$Loss_{\text{edge}}(\theta) = -\frac{1}{m}\left(\sum_{(u,v)\in\mathcal{C}} J_{u,v}^{\mathcal{C}}(\theta) + \sum_{(u,v)\in\mathcal{M}} J_{u,v}^{\mathcal{M}}(\theta)\right), \tag{13}$$

where

$$\begin{aligned} J_{u,v}^{\mathcal{C}}(\theta) &= \log(s(-d(u,v;\theta))), \\ J_{u,v}^{\mathcal{M}}(\theta) &= \log(1 - s(-d(u,v;\theta))), \end{aligned} \tag{14}$$

and $s(\cdot)$ is the sigmoid function

$$s(x) = 1/(1+\exp(-x)). \tag{15}$$

The above loss is applied in the situation that only text strokes are grouped into text lines. It ignores the situation that at least one endpoint of edge is a non-text stroke. Here we simply extend the loss to deal with the non-text case: if one end of edge u is the text stroke and the another end of edge v is the non-text stroke, the non-text stroke v is interpreted as the "text stroke in the different text line" and update the corresponding link in $\mathcal{C}$; if both ends of an edge are non-text strokes, this edge is thrown away as the text line grouping problem is undefined with non-text strokes. We refer to this modification as "non-text (NT) loss" and its effect will be analyzed in the ablation experiment. For clarity, the computation of $\mathcal{M}$ and $\mathcal{C}$ is summarized as Algorithm 2.

The final loss is the sum of these two losses:

$$Loss(\theta) = Loss_{\text{node}}(\theta) + Loss_{\text{edge}}(\theta). \tag{16}$$

Hard sample mining. The distance learning loss stated above can be seen as a kind of hard mining technique for selecting hard samples to distinguish text lines in one document. As there are different performance needs in downstream tasks, e.g. high recall or precision, we also employ a simple hard sample mining technique for optimizing specific metric. For simplicity, the text line recall rate is chosen as the metric for evaluating the difficulty. The range

Algorithm 2 Computation of critical links.

```
Input: Distances $\{d(u,v) \mid (u,v) \in E\}$;
       Annotated text lines $L = \{l_j\}$;
       Ground truth stroke labels $\mathbf{y}$;
Output: Critical links $\mathcal{M}$ and $\mathcal{C}$;
1: Initialize $\mathcal{M} = \{\}$, $\mathcal{C} = \{\}$;
2: for $(u,v) \in E$ do
3:   if $(u,v)$ from the same text line $l_j$ then
4:     if $\mathcal{M}_j = \emptyset$ or $d(u,v) > d(\mathcal{M}_j)$ then
5:       Update $\mathcal{M}_j$ with $(u,v)$;
6:   else if $(u,v)$ from different text lines $l_j$ and $l_k$ then
7:     if $\mathcal{C}_j = \emptyset$ or $d(u,v) < d(\mathcal{C}_j)$ then
8:       Update $\mathcal{C}_j$ with $(u,v)$;
9:     if $\mathcal{C}_k = \emptyset$ or $d(u,v) < d(\mathcal{C}_k)$ then
10:      Update $\mathcal{C}_k$ with $(u,v)$;
11:  else if $y_u = 0$ or $y_v = 0$ then
12:    if $y_u = 0$ and $v$ from text line $l_j$ then
13:      if $\mathcal{C}_j = \emptyset$ or $d(u,v) < d(\mathcal{C}_j)$ then
14:        Update $\mathcal{C}_j$ with $(u,v)$;
15:    if $y_v = 0$ and $u$ from text line $l_j$ then
16:      if $\mathcal{C}_j = \emptyset$ or $d(u,v) < d(\mathcal{C}_j)$ then
17:        Update $\mathcal{C}_j$ with $(u,v)$;
18: return $\mathcal{M}$ and $\mathcal{C}$;
```

of recall rate lies in $[0,1]$ and the higher recall rate means the better performance. In the training process, given a mini batch of samples B, we first compute the text line recall rate of each document $\{z^{(i)} \mid i = 1,\ldots,|B|\}$, then the weight of each sample can be computed by

$$w^{(i)} = \frac{1 - z^{(i)}}{\sum_{j=1}^{|B|} 1 - z^{(j)}}, \quad i = 1,\ldots,|B| \tag{17}$$

The overall training process is summarized in Algorithm 3.

Algorithm 3 Training process.

```
Input: Samples $D^{(i)} = \{\mathbf{H}^{(i)}_{\text{raw}}, \mathbf{F}^{(i)}_{\text{raw}}, G^{(i)}\}, i = 1,\ldots,N$;
       Ground truth stroke labels $\mathbf{y}^{(i)}, i = 1,\ldots,N$;
       Annotated text lines $L^{(i)} = \{l^{(i)}_j\}, i = 1,\ldots,N$;
Output: Learned model parameter $\theta^*$;
1: Randomly initialize $\theta$
2: repeat
3:   Sample a mini-batch $B = \{D^{(i)}, \mathbf{y}^{(i)}, L^{(i)}\}, i = 1,\ldots,|B|$
4:   for $i = 1$ to $|B|$ do
5:     Compute $\mathbf{p}^{(i)}$ and $\{d^{(i)}(u,v)|(u,v) \in G^{(i)}\}$ according to (10);
6:     Compute $Loss^{(i)}_{\text{node}}$ with $\mathbf{p}^{(i)}$ and $\mathbf{y}^{(i)}$ according to (11);
7:     Compute critical links $\mathcal{M}^{(i)}$ and $\mathcal{C}^{(i)}$ according to Algorithm 2;
8:     Compute $Loss^{(i)}_{\text{edge}}$ with $\mathcal{M}^{(i)}$ and $\mathcal{C}^{(i)}$ according to (13);
9:     Compute the predicted text lines $\hat{B}^{(i)}$ with Algorithm 1;
10:    Compute the specific performance metric $z^{(i)}$ with $\hat{B}^{(i)}$ and $B^{(i)}$;
11:  Compute the sample weight $\{w^{(i)} \mid i = 1,\ldots,|B|\}$ according to (17);
12:  Compute the mini-batch loss as
       $$Loss_{\text{batch}}(\theta) = \frac{1}{|B|}\sum_{i=1}^{|B|} Loss^{(i)}_{\text{node}}(\theta) + \sum_{i=1}^{|B|} w^{(i)} Loss^{(i)}_{\text{edge}}(\theta) \tag{18}$$
13:  Compute the gradient of $Loss_{\text{batch}}$ and update $\theta$;
14: until limit on the number of training iterations is reached.
```

3.4.2. Training details

We implement the training algorithm with the DGL library [36] and its PyTorch backend [37].

The weight matrix parameters are initialized with normal samples $\mathcal{N}(0,\sigma^2)$, where $\sigma = \sqrt{\frac{2}{r+c}}$ and r, c are the number of rows and columns in the matrix [38].

The training procedure is optimized with Adam optimizer [39]. The initial learning rate η_0 is chosen as 0.005 and the learning rate is adjusted with the cosine decay [40]. The early stopping strategy is used to select the best model based on the performance on the validation set. The text line recall rate metric is chosen as the indicator of performance. The stochastic weight average technique [41] is also used to balance the stroke classification and text line grouping tasks.

4. Experiments

To evaluate the performance of the proposed method, we conducted experiments on two public datasets of online handwritten documents: IAMOnDo [13] and Kondate [16]. For each dataset, we perform two sets of experiments. The first set is to predict the text lines given the ground truth stroke labels, which is denoted as **PerfectWD** setting in Zhou et al. [1] and the second set is to predict the stroke labels and text lines given the input document, which is denoted as **CrudeWD** setting in Zhou et al. [1].

4.1. Datasets

IAMOnDo. The IAMOnDo dataset [13] is a publicly available collection of freely handwritten English online documents. The dataset consists of 1000 documents, mixing texts, drawing, diagrams, formulas, tables, lists and marking elements arranged in an unconstrained way. We follow the traditional data partitioning of the dataset [13]. Documents from *set0* and *set1* constitute the training set, *set2* is the validation set, and *set3* is the test set. The corresponding ground truth labels are derived as suggested in Indermühle [42].

Kondate. The Kondate dataset [16] is a collection of online handwritten Japanese documents. The dataset contains the documents written by 67 writers, 41 pages per writer covering the stroke types of text, formula, figure, ruled line and editing mask. We follow a similar partitioning of the dataset as in Zhou et al. [1], where 2056 pages are used for training, 820 pages for validation, 861 pages for testing.

Both datasets have the formula stroke type and the formula type contains both characters and non-characters. There is no annotation of text line in the formula in both datasets and we follow the similar processing method as [1] to exclude those strokes in the text line grouping task.

4.2. Evaluation metrics

For text/non-text classification, the accuracy is used as the evaluation metric, which is defined as:

$$\text{accuracy} = \frac{\sum_{i=1}^{N}\sum_{t=1}^{T_i} \delta(\hat{y}_{it} = y_{it})}{\sum_{i=1}^{N} T_i}, \tag{19}$$

where N is the number of documents, T_i is the number of strokes in the ith document. $\hat{y}_{it}$ and y_{it} are the corresponding prediction and ground truth.

For evaluating the performance of text line grouping, we follow the similar procedure as in Zhou et al. [1]. First, some definitions are introduced for metrics computation. A *one-to-one match* is a match between a result line and a ground-truth line that contain identical strokes. A *g_one-to-many match* is a match between a ground truth line and two or more result lines that their union equals the ground truth line. Similarly, a *d_one-to-many match* is a match between a predicted line and two or more ground truth lines that their union equals the predicted line. *misses* are those text lines in the ground truth that do not match with any predicted lines, i.e. the intersection between the ground truth line and any predicted line is empty. *false-alarms* are those predicted lines that do not match any ground truth lines. A *g_segmentation* error is produced if there exists at least one predicted text line, such that the intersection of the ground truth line is a nonempty proper subset of the ground truth line. A *d_segmentation error* is produced if there exists at least one ground truth line, such that the intersection of the predicted line is a nonempty proper subset of the predicted line. A *g_merge error* is produced if there exists at least one predicted line with some strokes belonging to the ground truth line cannot have *g_one-to-many match*. A *d_merge error* is produced if there exists at least one ground truth line, such that some strokes of the predicted line belong to the ground truth line and some do not.

After definitions of the matches and errors, various metrics for evaluating text line grouping performance can be defined. Let *one2one, g_one2many, d_many2one, misses, false_alarms, g_segmentation, d_segmentation, g_merge, d_merge* denote the counts of the corresponding matches and errors, M_{gt} is the number of ground truth text lines, and M_{pred} is the number of predicted text lines. In the following, the performance metrics are defined:

Segmentation recall rate (SR)

$$SR = \frac{one2one}{M_{gt}}. \tag{20}$$

Detection rate (DR)

$$DR = \frac{one2one}{M_{gt}} + \frac{g\_one2many}{M_{gt}}. \tag{21}$$

Missed detection rate (MDR)

$$MDR = \frac{misses}{M_{gt}}. \tag{22}$$

Recognition accuracy (RA)

$$RA = \frac{one2one}{M_{pred}} + \frac{d\_many2one}{M_{pred}}. \tag{23}$$

False-alarm rate (FAR)

$$FAR = \frac{false\_alarms}{M_{pred}}. \tag{24}$$

Entity detection metric (EDM)

$$EDM = \frac{2 \cdot DR \cdot RA}{DR + RA}. \tag{25}$$

Edit cost index (ECI)

$$ECI = 1 - \frac{2 \cdot one2one}{M_{gt} + M_{pred}}. \tag{26}$$

Edit distance rate (EDR)

$$EDR = \frac{d\_segmentation + d\_merge}{M_{gt}}. \tag{27}$$

Segmentation error rate (SER)

$$SER = \frac{2 \cdot g\_segmentation \cdot d\_segmentation}{M_{pred} \cdot g\_segmentation + M_{gt} \cdot d\_segmentation}. \tag{28}$$

Merge error rate (MER)

$$MER = \frac{2 \cdot g\_merge \cdot d\_merge}{M_{pred} \cdot g\_merge + M_{gt} \cdot d\_merge}. \tag{29}$$

4.3. Results and discussions

In this section, the results of the proposed method are presented. First we compare our method with some previous representative methods, then we discuss the effects of different designs in our method by ablation experiments. For comparing with other methods, the default settings (hyperparameters) of the proposed method are summarized in Table A.10. The graphs are contructed with temporal edges and radius-based spatial edges. We will show in ablation experiments that these settings are reasonable to achieve good performance.

When comparing with other methods, as the variance of accuracies was not reported in the previous work, we use the z-test [43] as the statistical test method. Specifically, denote the metric of two systems by p_1 and p_2 ($p_1 < p_2$) and take the null hypothesis that two metrics p_1 and p_2 do not differ significantly, the distribution of variable $p_2 - p_1$ can be approximated as normal with zero mean and variance $\sigma^2 = 2p(1-p)/n$, where p is the average of p_1 and p_2 and n is the number of test samples. The variable $z = (p_2 - p_1)/\sigma$ has a standard normal distribution. When the value of $|z| > 1.96$, the null hypothesis can be rejected with confidence higher than 0.95.

4.3.1. Compared with previous methods

The results of PerfectWD and CrudeWD on two datasets are shown in Tables 1–4. In the PerfectWD setting of IAMOnDo dataset, we compare our result with that in Delaye and Lee [10], where recall rate (SR) 0.9188 and tolerant recall rate (TSR, same as the detection rate in our paper) 0.9555 were reported. Our work uses the similar metric distance learning framework as in Delaye and Lee [10], but we consider the interaction between edge features and learn the distributed edge representations using deep GNN. Besides, our inference process uses only edges in the temporal-spatial relational graph, while all pairs of edges (a complete graph) were used in Delaye and Lee [10]. The recall rate and detection rate of our system are 0.9380 and 0.9613, outperforming the results of [10] by 1.92% and 0.58% with z-value 3.58 and 1.40. This implies that our method performs better on the recall rate with high level of significance($|z| > 1.96$) and comparably on the detection rate ($|z| < 1.96$). This work is the first to report results for the CrudeWD setting on the IAMOnDo dataset. The recall rate, detection rate, EDM and EDR are 0.9034, 0.9131, 0.9077 and 0.0618, respectively.

In the PerfectWD setting on the Kondate dataset, we compare our method with the method in Zhou et al. [1], using the metrics EDM and EDR. Our method yields EDM 0.9594 and EDR 0.0489, outperforming the results of [1] by 0.03% and 1.80% with z-value 0.07 and 4.09, respectively. This implies that our method performs better on the EDR with high level of significance($|z| > 1.96$) and comparably on the EDM ($|z| < 1.96$). In the CrudeWD setting on the Kondate dataset, we also compare with the method in Zhou et al. [1]. The EDM, EDR and stroke classification accuracy of our method are 0.9247, 0.0634 and 0.9971, outperforming the results of [1] by 2.55%, 4.80% and 0.30%, with z-value 4.93, 9.31 and 15.29, respectively. This implies that our method performs better on EDM, EDR and stroke classification accuracy with high level of significance ($|z| > 1.96$).

4.3.2. Ablation experiments

We also conducted ablation experiments to evaluate the effects of different settings. Particularly, we evaluate the options of construction of relational graph, the loss functions, the edge pooling technique, and the number of EPAT layers.

Construction of relational graph. We perform detailed analysis in different ways of relational graph construction on the IAMOnDo

Table 1
Text line grouping results for PerfectWD on the IAMOnDo dataset. Bold face indicates the best performance among different methods.

Method	SR	DR	MDR	RA	FAR	EDM	ECI	EDR	SER	MER
Delaye et al. [10]	0.9188	0.9555	–	–	–	–	–	–	–	–
w/o hard mine	0.9304	0.9529	0.0000	0.9769	0.0000	0.9647	0.0736	0.0803	0.0358	0.0278
w/o edge pool	0.9248	0.9520	0.0000	0.9667	0.0000	0.9593	0.0890	0.1104	0.0478	0.0287
EPAT	**0.9380**	**0.9613**	0.0000	**0.9814**	0.0000	**0.9712**	**0.0675**	**0.0758**	**0.0355**	**0.0230**

Table 2
Text line grouping results for CrudeWD on the IAMOnDo dataset. The last column shows stroke classification accuracy. Bold face indicates the best performance among different methods.

Method	SR	DR	MDR	RA	FAR	EDM	ECI	EDR	SER	MER	acc
Delaye et al. [3]	–	–	–	–	–	–	–	–	–	–	0.9721
Ye et al. [5]	–	–	–	–	–	–	–	–	–	–	0.9796
Phan et al. [5]	–	–	–	–	–	–	–	–	–	–	0.9830
w/o hard mine	0.8978	**0.9172**	**0.0255**	0.9014	0.0635	**0.9092**	0.1230	0.0851	0.0394	**0.0328**	0.9856
w/o NT loss	0.8794	0.8928	0.0309	**0.9025**	**0.0498**	0.8977	0.1254	0.0838	0.0325	0.0475	0.9840
w/o edge pool	0.8816	0.9071	0.0302	0.8855	0.0753	0.8962	0.1481	0.1091	0.0459	0.0359	0.9853
EPAT	**0.9034**	0.9131	0.0300	0.9024	0.0597	0.9077	**0.1077**	**0.0618**	**0.0235**	0.0352	**0.9860**

Table 3
Text line grouping results for PerfectWD on the Kondate dataset. Bold face indicates the best performance among different methods.

Method	SR	DR	MDR	RA	FAR	EDM	ECI	EDR	SER	MER
Pre-segmentation [1]	–	0.8024	0.0000	0.8891	0.0012	0.8435	0.1702	0.1219	0.0277	0.1160
Temporal segmentation [1]	–	0.9547	0.0000	0.9494	0.0011	0.9520	0.0656	0.0870	0.0424	**0.0246**
Spatial merge [1]	–	**0.9573**	0.0000	0.9609	0.0010	0.9591	**0.0530**	0.0669	0.0298	0.0265
w/o hard mine	0.9290	0.9453	0.0000	0.9709	0.0000	0.9579	0.0604	0.0531	0.0218	0.0353
w/o edge pool	0.9191	0.9333	0.0000	0.9661	0.0000	0.9494	0.0588	0.0584	0.0202	0.0405
EPAT	**0.9340**	0.9467	0.0000	**0.9725**	0.0000	**0.9594**	0.0552	**0.0489**	**0.0175**	0.0334

Table 4
Text line grouping results for CrudeWD on the Kondate dataset. The last column shows stroke classification accuracy. Bold face indicates the best performance among different methods.

Method	SR	DR	MDR	RA	FAR	EDM	ECI	EDR	SER	MER	acc
Pre-segmentation [1]	–	0.6611	0.0050	0.5752	0.0466	0.6152	0.3944	0.4710	0.2275	0.1082	–
Temporal segmentation [1]	–	0.7760	0.0075	0.6563	0.0258	0.7111	0.2987	0.4102	0.2304	0.0413	–
Temporal merge [1]	–	0.9052	**0.0072**	0.8822	0.0279	0.8936	0.1195	0.1271	0.0563	0.0537	–
Spatial merge [1]	–	0.9047	**0.0072**	0.8939	0.0242	0.8992	0.1097	0.1114	0.0450	0.0582	0.9941
w/o hard mine	0.8861	0.8912	0.0163	0.9311	0.0183	0.9107	0.0938	0.0719	0.0270	0.0507	0.9962
w/o NT loss	0.8838	0.8906	0.0232	0.9289	0.0158	0.9093	0.0941	0.0726	0.0285	0.0488	0.9950
w/o edge pool	0.8660	0.8753	0.0183	0.9301	**0.0135**	0.9019	0.1093	0.0864	0.0312	0.0619	0.9965
EPAT	**0.9102**	**0.9183**	0.0133	**0.9312**	0.0183	**0.9247**	**0.0834**	**0.0634**	**0.0262**	**0.0357**	**0.9971**

dataset. Since the results of PerfectWD and CrudeWD show similar tendency for different hyperparameter settings, to save space, we only show the results of CrudeWD in Table 5.

First, graphs with different number of temporal edges k_t (EPAT (time)) are compared. The results in Table 5 show that the best performance of most metrics is achieved by EPAT (time, $k_t = 2$) on the IAMOnDo dataset. When the number of time neighbors is increased further, the performance is not improved, because adding too many time neighbors also introduces noise edges.

Then graphs with spatial edges of different radius and k-nearest parameters are compared. Since our cluster algorithm relies on the edges of relational graph, we can see that graphs of EPAT (radius space) with small k_r (5,10,20) do not perform well for text line segmentation as some strokes in one line are not connected. Only the EPAT (radius space, $k_r = 40$) model performs competitively, but it is much slower because of large number of edges. Nevertheless, combining temporal edges and radius spatial edges of moderate size ($k_r = 10, 20$) leads to superior performance. The performance of the default setting ($k_t = 2, k_r = 20$) is nearly the best. We can see that graph construction with k-nearest spatial edges only does not perform well, combining temporal edges and k-nearest sptial edges performs better, but not so well as combining radius and k-nearest sptial edges. Finally, combining all the three types of edges (temporal, radius and k-nearest edges) performs comparably well as the default setting (temporal $k_t = 2$, radius spatial $k_r = 20$). This verifies that the default setting is reasonable and competitive.

After determining the graph construcion parameters, the variants of the configurations of the proposed method are summarized in Table 6.

Variants of loss function. We evaluate the effect of the hard sample mining technique and non-text loss introduced in Section 3.4.1. To evaluate the effect of the hard sample mining technique, the recall rate is chosen to measure the difficulty of document segmentation. From Tables 1–4, it is shown that the recall rate of EPAT model has a consistent improvement compared to the model w/o hard sample mining in both PerfectWD and CrudeWD settings on IAMOnDo and Kondate datasets. This verifies that the hard sample mining technique can boost specific metric effectively.

Another important design of loss function is to consider the non-text strokes in the computation of critical links. As non-texts stroke cannot be linked to any text stroke to form text line, it is beneficial to increase the distance between them. The results in Tables 2 and 4 show that the recall rate, detection rate, EDM and EDR of the model w/o non-text (NT) loss have consistent drops compared to the EPAT model.

Table 5
Ablation study of different construction of relational graph for CrudeWD on the IAMOnDo dataset. '*' indicates the 'EPAT' model in Table 2.

Method	SR	DR	MDR	RA	FAR	EDM	ECI	EDR	SER	MER	acc
EPAT (time, $k_t = 1$)	0.8660	0.8896	0.0386	0.9057	0.0436	0.8976	0.1407	0.1065	0.0520	0.0386	0.9855
EPAT (time, $k_t = 2$)	0.8701	0.8844	0.0372	0.9110	0.0331	0.8975	0.1255	0.0894	0.0397	0.0465	0.9853
EPAT (time, $k_t = 3$)	0.8662	0.8781	0.0380	0.8988	0.0478	0.8884	0.1318	0.0832	0.0328	0.0503	0.9845
EPAT (radius space, $k_r = 5$)	0.2351	0.9468	0.0210	0.9468	0.0273	0.9468	0.9346	5.7932	0.8263	0.0019	0.9773
EPAT (radius space, $k_r = 10$)	0.4944	0.9412	0.0292	0.9521	0.0323	0.9466	0.7494	2.3589	0.5846	0.0069	0.9812
EPAT (radius space, $k_r = 20$)	0.6392	0.9170	0.0244	0.9238	0.0392	0.9204	0.5520	1.1396	0.3996	0.0209	0.9837
EPAT (radius space, $k_r = 40$)	0.8896	0.9116	0.0197	0.8615	0.1005	0.8858	0.1495	0.0946	0.0405	0.0382	0.9806
EPAT (time + radius space, $k_t = 1, k_r = 5$)	0.8740	0.9120	0.0341	0.9162	0.0491	0.9141	0.1471	0.1271	0.0650	0.0305	0.9849
EPAT (time + radius space, $k_t = 1, k_r = 10$)	0.8984	0.9192	0.0229	0.9017	0.0589	0.9104	0.1229	0.0903	0.0426	0.0320	0.9851
EPAT (time + radius space, $k_t = 1, k_r = 20$)	0.8987	0.9086	0.0231	0.8942	0.0620	0.9013	0.1152	0.0728	0.0310	0.0392	0.9843
EPAT (time + radius space, $k_t = 2, k_r = 5$)	0.8917	0.9058	0.0248	0.8926	0.0695	0.8992	0.1273	0.0780	0.0323	0.0397	0.9850
EPAT (time + radius space, $k_t = 2, k_r = 10$)	0.9032	0.9180	0.0255	0.8952	0.0688	0.9064	0.1169	0.0715	0.0306	0.0325	0.9856
EPAT (time + radius space, $k_t = 2, k_r = 20$)	0.9034	0.9131	0.0300	0.9024	0.0597	0.9077	0.1077	0.0618	0.0235	0.0352	0.9860
EPAT (k-nearest space, $k_s = 5$)	0.7772	0.9058	0.0311	0.9052	0.0550	0.9055	0.3087	0.4062	0.1951	0.0307	0.9835
EPAT (k-nearest space, $k_s = 10$)	0.8792	0.9123	0.0303	0.8682	0.0614	0.8897	0.1880	0.2193	0.0757	0.0309	0.9834
EPAT (k-nearest space, $k_s = 20$)	0.8892	0.9049	0.0264	0.8536	0.0795	0.8785	0.1829	0.1977	0.0502	0.0361	0.9798
EPAT (time + k-nearest space, $k_t = 1, k_s = 20$)	0.8935	0.9151	0.0279	0.8713	0.0592	0.8927	0.2055	0.2848	0.0608	0.0290	0.9835
EPAT (time + k-nearest space, $k_t = 2, k_s = 20$)	0.8922	0.9205	0.0177	0.8203	0.1389	0.8756	0.2277	0.2867	0.0671	0.0358	0.9786
EPAT (radius + k-nearest space, $k_r = 10, k_s = 10$)	0.8881	0.9164	0.0292	0.8827	0.0803	0.8992	0.1480	0.1117	0.0519	0.0303	0.9821
EPAT (radius + k-nearest space, $k_r = 20, k_s = 10$)	0.8907	0.9164	0.0255	0.8894	0.0742	0.9027	0.1387	0.1003	0.0493	0.0307	0.9821
EPAT (radius + k-nearest space, $k_r = 10, k_s = 20$)	0.8948	0.9205	0.0326	0.9127	0.0578	0.9166	0.2097	0.2995	0.0654	0.0244	0.9817
EPAT (radius + k-nearest space, $k_r = 20, k_s = 20$)	0.9019	0.9129	0.0240	0.7920	0.0737	0.8482	0.1741	0.2001	0.0517	0.0301	0.9805
EPAT (time + radius + k-nearest space, $k_t = 1, k_r = 10, k_s = 20$)	0.9000	0.9090	0.0236	0.8838	0.0795	0.8963	0.1192	0.0624	0.0224	0.0371	0.9830
EPAT (time + radius + k-nearest space, $k_t = 2, k_r = 10, k_s = 20$)	0.9036	0.9118	0.0229	0.8755	0.0850	0.8933	0.1206	0.0646	0.0233	0.0376	0.9838

Table 6
Methods and its associated modules. 'time' denotes temporal edges. 'radius space' denotes radius-based spatial edges. '*k*-nearest space' denotes *k*-nearest spatial edges. 'HM' denotes hard mining. 'NT' denotes non-text loss. 'EP' denotes edge pooling technique.

Method	Time	Radius space	*k*-nearest space	HM	NT	EP
EPAT (time)	✓			✓	✓	✓
EPAT (radius space)		✓		✓	✓	✓
EPAT (k-nearest space)			✓	✓	✓	✓
w/o hard mine	✓	✓			✓	✓
w/o non-text loss	✓	✓		✓		✓
w/o edge pool	✓	✓		✓	✓	
EPAT	✓	✓		✓	✓	✓

Edge pooling technique. The motivation of introducing the edge pooling technique is to enhance the aggregation between edge features as the text line grouping task needs a broad repetitive field to decide whether two strokes should be merged. The results in Tables 1–4 show that the edge pooling technique is very powerful to boost the system performance in nearly all metrics. This is because edge pooling can effectively capture the dependency between adjacent edge features and the aggregation of adjacent edge features is beneficial to the text line grouping task.

Number of EPAT layers. We perform experiments with different number of EPAT layers in the graph network. The results in Fig. 3 show that the recall rate, DR, EDM and 1-EDR are consistently increasing before 5 layers and become flat or oscillating after that. This implies the increase of layers will boost the performance when the number of layers is small but may have little effect when there are enough layers.

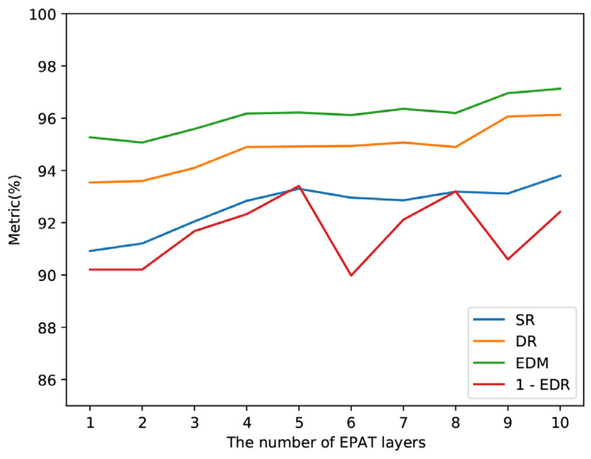

Fig. 3. The performance of different layers in the PerfectWD setting of IAMOnDo dataset. The EDR metric is transformed to 1 - EDR for plot convenience.

4.4. Complexity

We implemented the experiments on a computer with Intel(R) Core(TM) i7-6800K CPU(3.40 GHz) and GeForce GTX1080Ti. Table 7 presents the number of parameters and test speed on different datasets. Enhanced with the modern GPU computation device, the inference speed of our model is about 0.01s on the Kondate dataset, which is much faster than 0.60s reported in Zhou et al. [1].

Table 7
The number of parameters and inference speed on different datasets.

Dataset	Parameters	speed (s/doc)
IAMOnDo-Perfect	0.79 M	0.018
IAMOnDo-Crude	0.79 M	0.029
Kondate-Perfect	0.79 M	0.012
Kondate-Crude	0.79 M	0.015

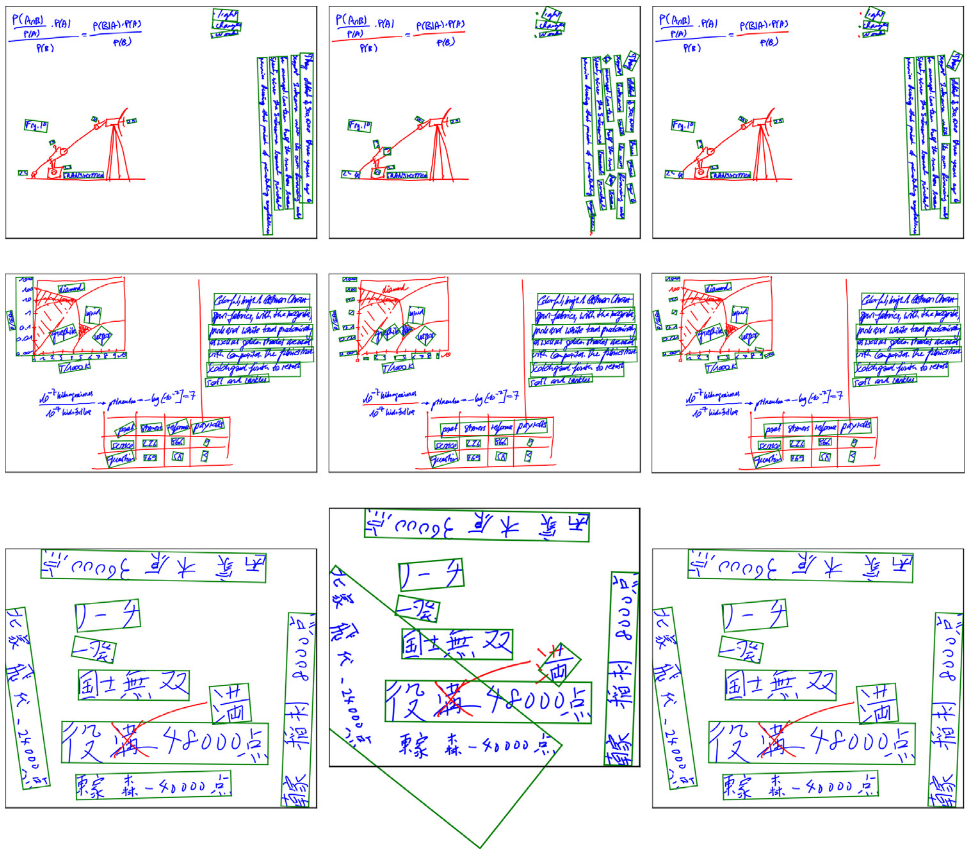

Fig. 4. Examples from IAMOnDo dataset (top two rows) and Kondate dataset (bottom row). The first column shows the ground truth stroke labels and text lines of documents. The second column shows the predicted results by the model w/o edge pool. The third column shows the predicted results by the EPAT model. Text strokes are shown in blue, non-text strokes are shown in red. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.5. Qualitative analysis

Fig. 4 presents some examples of stroke classification and text line grouping results from IAMOnDo and Kondate datasets. The three columns show the ground truth, the results of the model w/o edge pool and the EPAT model, respectively. We can see that compared to the model w/o edge pool, the EPAT model generates fewer errors of text line segmentation. Particularly, in the third example, the document has both horizontal and vertical text lines, and some non-text strokes overlap with text strokes, yet the EPAT model still classify them correctly.

5. Conclusion

In this paper, we propose a novel framework to perform stroke classification and text line grouping problems simultaneously in online handwritten documents using edge pooling attention network (EPAT). The stroke classification and text line grouping problems are formulated as node classification and node clustering in a relational graph, respectively. The edge pooling attention layer in the network can effectively aggregate information between adjacent node and edge features. The EPAT is trained jointly in multi-task learning using cross entropy loss for stroke classification and distance metric loss for text line grouping, respectively. Experimental results show that the joint framework promise both stroke classification and text line grouping, and has achieved state-of-the-art performance on public datasets.

In the future, there are several potential directions that can be explored. In view of features, to extract features directly from the raw stroke trajectory by GNN may further boost the system performance. In view of applications, the proposed EPAT framework can be applied to offline documents, and can detect and group multi-type objects in documents.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been supported by the National Key Research and Development Program Grant 2018YFB1005000, the National Natural Science Foundation of China (NSFC) grants 61773376 and 61721004.

Appendix A

A1. Stroke features

This section summarizes the node features and edge features used in our model.

A2. Hyperparameters

This section summarizes the hyperparameters used in our model. For all EPAT layers, the hyperparameter of each layer(C', D', K) are kept the same. Parameters k_r and k_s are calculated based on statistics of training dataset, as illustrated in Section 3.2.1.

Table A.8
Node features extracted from stroke x_k.

#	Description
1	Trajectory length of x_k
2	Area of the convex hull of x_k
3	Duration of the stroke
4	Ratio of the principal axis of x_k
5	Rectangularity of the minimum area bounding rectangle of x_k
6	Circular variance of points of x_k around its centroid
7	Normalized centroid offset along the principal axis
8	Ratio between first-to-last point distance and trajectory length
9	Accumulated curvature
10	Accumulated squared perpendicularity
11	Accumulated signed perpendicularity
12	Width of x_k, normalized by the median stroke height in the document
13	Height of x_k, normalized by the median stroke height in the document
14	Number of temporal neighbours of x_k
15	Number of spatial neighbours of x_k
16	Average of the distances from x_k to time neighbours
17	Standard deviation of the distances from x_k to time neighbours
18	Average of lengths of time neighbours
19	Standard deviation of lengths of time neighbours
20	Average of the distances from x_k to space neighbours
21	Standard deviation of the distances from x_k to space neighbours
22	Average of lengths of space neighbours
23	Standard deviation of lengths of space neighbours

Table A.9
Edge features extracted from a pair of stroke x_i, x_j.

#	Description
1	Minimum distance between 2 strokes
2	Minimum distance between endpoints of strokes
3	Maximum distance between endpoints of strokes
4	Distance between the centers of the 2 bounding boxes of 2 strokes
5	Horizontal distances between centroids of strokes
6	Vertical distances between centroids of strokes
7	Off-stroke distance between 2 strokes
8	Off-stroke distance projected on X and Y axes
9	Temporal distance between 2 strokes
10	Left endpoints of bounding boxes between strokes
11	Right endpoints of bounding boxes between strokes
12	Top endpoints of bounding boxes between strokes
13	Bottom endpoints of bounding boxes between strokes
14-15	min/max of ratio of off-stroke distance to temporal distance
16-17	min/max of ratio of off-stroke distance on X, Y axes to temporal distance
18-19	min/max of ratio of area of the largest bounding box of 2 strokes to their union
20-21	min/max of ratio of widths of the bounding boxes of 2 strokes
22-23	min/max of ratio of heights of the bounding boxes of 2 strokes
24-25	min/max of ratio of diagonals of the bounding boxes of 2 strokes
26-27	min/max of ratio of areas of the bounding boxes of 2 strokes
28-29	min/max of ratio of lengths of 2 strokes
30-31	min/max of ratio of duration of 2 strokes
32-33	min/max of ratio of curvatures of 2 strokes

Table A.10
Hyperparameters for all experiments.

Hyperparameter	IAMOnDo-Perfect	IAMOnDo-Crude	Kondate-Perfect	Kondate-Crude
number of time neighbors (k_t)	2	2	3	3
radius-based spatial threshold (k_r)	10	20	20	20
number of k-nearest spatial neighbors (k_s)	10	10	12	12
number of node output features (C)	16	16	16	16
number of edge output features (D')	64	64	64	64
number of attention heads (K)	8	8	8	8
number of EPAT layers	10	10	10	10
batch size	16	16	16	16
number of training epochs	75	200	150	200
initial learning rate	0.005	0.005	0.005	0.005
temperature (β)	0.5	0.5	0.5	0.5

References

[1] X.-D. Zhou, D.-H. Wang, C.-L. Liu, A robust approach to text line grouping in online handwritten japanese documents, Pattern Recognit. 42 (9) (2009) 2077–2088.

[2] X.-D. Zhou, C.-L. Liu, Text/non-text ink stroke classification in japanese handwriting based on Markov random fields, in: Proceedings of the International Conference on Document Analysis and Recognition, 1, 2007, pp. 377–381.

[3] A. Delaye, C.-L. Liu, Contextual text/non-text stroke classification in online handwritten notes with conditional random fields, Pattern Recognit. 47 (3) (2014) 959–968.

[4] T. Van Phan, M. Nakagawa, Combination of global and local contexts for text/non-text classification in heterogeneous online handwritten documents, Pattern Recognit. 51 (2016) 112–124.

[5] J.-Y. Ye, Y.-M. Zhang, C.-L. Liu, Joint training of conditional random fields and neural networks for stroke classification in online handwritten documents, in: Proceedings of the International Conference on Pattern Recognition, 2016, pp. 3264–3269.

[6] J.-Y. Ye, Y.-M. Zhang, Q. Yang, C.-L. Liu, Contextual stroke classification in online handwritten documents with graph attention networks, in: Proceedings of the International Conference on Document Analysis and Recognition, 2019, pp. 993–998.

[7] M. Shilman, Z. Wei, S. Raghupathy, P. Simard, D. Jones, Discerning structure from freeform handwritten notes, in: Proceedings of the International Conference on Document Analysis and Recognition, 2003, pp. 60–65.

[8] M. Ye, H. Sutanto, S. Raghupathy, C. Li, M. Shilman, Grouping text lines in freeform handwritten notes, in: Proceedings of the International Conference on Document Analysis and Recognition, 2005, pp. 367–371.

[9] M. Liwicki, E. Indermuhle, H. Bunke, On-line handwritten text line detection using dynamic programming, in: Proceedings of the International Conference on Document Analysis and Recognition, 1, 2007, pp. 447–451.

[10] A. Delaye, K. Lee, A flexible framework for online document segmentation by pairwise stroke distance learning, Pattern Recognit. 48 (4) (2015) 1197–1210.

[11] C.M. Bishop, M. Svensen, G.E. Hinton, Distinguishing text from graphics in on–line handwritten ink, in: International Conference on Frontiers in Handwriting Recognition, 2004, pp. 142–147.

[12] J.-Y. Ye, Y.-M. Zhang, Q. Yang, C.-L. Liu, Contextual stroke classification in online handwritten documents with edge graph attention networks, SN Comput. Sci. 1 (163) (2020).

[13] E. Indermühle, M. Liwicki, H. Bunke, Iamondo-database: an online handwritten document database with non-uniform contents, in: Proceedings of the International Workshop on Document Analysis Systems, 2010, pp. 97–104.

[14] A.K. Jain, A.M. Namboodiri, J. Subrahmonia, Structure in on-line documents, in: Proceedings of the International Conference on Document Analysis and Recognition, 2001, pp. 844–848.

[15] E.H. Ratzlaff, Inter-line distance estimation and text line extraction for unconstrained online handwriting, in: Proceedings of the Intenetional Conference on Frontiers in Handwriting Recognition, 2000, pp. 33–42.

[16] K. Mochida, M. Nakagawa, Separating figures, mathematical formulas and japanese text from free handwriting in mixed online documents, Int. J. Pattern Recognit. Artif. Intell. 18 (07) (2004) 1173–1187.

[17] F. Yin, C.-L. Liu, Handwritten chinese text line segmentation by clustering with distance metric learning, Pattern Recognit. 42 (12) (2009) 3146–3157.

[18] X.-C. Yin, X. Yin, K. Huang, H.-W. Hao, Robust text detection in natural scene images, IEEE Trans. Pattern Anal. Mach. Intell. 36 (5) (2013) 970–983.

[19] X.-C. Yin, W.-Y. Pei, J. Zhang, H.-W. Hao, Multi-orientation scene text detection with adaptive clustering, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1930–1937.

[20] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.

[21] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, in: International Conference on Learning Representation, 2016.

[22] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: International Conference on Learning Representation, 2013.

[23] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.

[24] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.

[25] A. Micheli, Neural network for graphs: a contextual constructive approach, IEEE Trans. Neural Netw. 20 (3) (2009) 498–511.

[26] J. Gilmer, S.S. Schoenholz, P. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, 2017, pp. 1263–1272.

[27] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representation, 2018.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[29] M.S. Schlichtkrull, T. Kipf, P. Bloem, R.V. Den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: European Semantic Web Conference, 2018, pp. 593–607.

[30] D. Busbridge, D. Sherburn, P. Cavallo, N. Hammerla, Relational graph attention networks, (2018), arXiv preprint arXiv:1904.05811.

[31] L. Gong, Q. Cheng, Exploiting edge features for graph neural networks, in: Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2019, pp. 9211–9219.

[32] M. Andrew L, H. Awni Y, N. Andrew Y, Rectifier nonlinearities improve neural network acoustic models, in: International Conference on Machine Learning, 30, 2013, p. 3.

[33] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, Outrageously large neural networks: the sparsely-gated mixture-of-experts layer, in: International Conference on Learning Representation, 2017.

[34] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, E.H. Chi, Modeling task relationships in multi-task learning with multi-gate mixture-of-experts, in: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1930–1939.

[35] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.

[36] M. Wang, L. Yu, D. Zheng, Q. Gan, Y. Gai, Z. Ye, M. Li, J. Zhou, Q. Huang, C. Ma, et al., Deep graph library: towards efficient and scalable deep learning on graphs, in: International Conference on Learning Representation, 2019.

[37] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: an imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems, 2019, pp. 8024–8035.

[38] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2010, pp. 249–256.

[39] D. Kingma, J. Ba, Adam: a method for stochastic optimization, in: International Conference on Learning Representation, 2015.

[40] T. He, Z. Zhang, H. Zhang, Z. Zhang, J. Xie, M. Li, Bag of tricks for image classification with convolutional neural networks, in: Proceedings of the International Conference on Computer Vision and Pattern Recognition, 2019, pp. 558–567.

[41] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, A.G. Wilson, Averaging weights leads to wider optima and better generalization, in: Conference on Uncertainty in Artificial Intelligence, 2018, pp. 876–885.

[42] E. Indermühle, Analysis of Digital Ink in Electronic Documents, University of Bern, 2012 Ph.D. thesis.

[43] G.W. Snedecor, W.G. Cochran, Statistical methods, Wiley-Blackwell, 1991.

Jun-Yu Ye received the M.S. degree from Nankai University (Tianjin, China) in 2014. Since 2014, he has been a Ph.D. candidate at the National Laboratory of Pattern Recognition, Institute of Automation of Chinese Academy of Science (Beijing, China). His research interests lies in machine learning with application to ink document analysis.

Yan-Ming Zhang is currently an associate professor at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences. He received the bachelor degree from Beijing University of Posts and Telecommunications in 2005, and the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation of Chinese Academy of Sciences in 2011. His research interests include machine learning, pattern recognition and document image analysis.

Qing Yang is a Professor at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences, Beijing, China. He received the Ph.D. degree in computer science from the Institute of Automation of Chinese Academy of Sciences in 1998. From 1999 to 2003, he worked as a Computer Scientist at the Lawrence Berkeley National Laboratory, Berkeley, CA. He conducts research on computer vision, web search engine and recommender systems.

Cheng-Lin Liu is a Professor at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences, Beijing, China, and is now the director of the laboratory. He is also a deputy director of the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China. He received the B.S. degree in electronic engineering from Wuhan University, Wuhan, China, the M.E. degree in electronic engineering from Beijing University of Technology, Beijing, China, the Ph.D. degree in pattern recognition and intelligent control from the Institute of Automation of Chinese Academy of Sciences, Beijing, China, in 1989, 1992 and 1995, respectively. He was a postdoctoral fellow at Korea Advanced Institute of Science and Technology (KAIST) and later at Tokyo University of Agriculture and Technology from March 1996 to March 1999. From 1999 to 2004, he was a research staff member and later a senior researcher at the Central Research Laboratory, Hitachi, Ltd., Tokyo, Japan. His research interests include pattern recognition, image processing, neural networks, machine learning, and especially the applications to character recognition and document analysis. He has published over 300 technical papers at prestigious journals and conferences. He is an associate editor-in-chief of Pattern Recognition Journal, and is on the editorial board of Image and Vision Computing, International Journal on Document Analysis and Recognition, and Cognitive Computation. He is a Fellow of the CAAI, the IEEE and the IAPR.