

document, in which each node represents a stroke and nodes are connected by edges according to the temporal and spatial relationship between strokes. Given the raw features and the graph, GNN updates the representation of each node by exploiting the current states of itself and its neighbors. In addition, due to the fact that different nodes in the neighborhood should have different influences, we also propose an edge attention mechanism to dynamically control the message passing routine between adjacent nodes.

Compared with previous methods, our proposal has two notable advantages. First, because of the graph structure and the deep architecture, our model enjoys much more powerful capacity in feature learning and relation learning, which results in improved classification accuracy over existing methods. Second, since the graph convolution can be expressed as a sparse matrix multiplication operation and support arbitrary graph structures, our model is easy to extend to integrate different types of contextual information without any approximation. We perform experiments on the IAMonDo dataset and show our model achieves state-of-the-art performances on the two tasks, text/non-text classification and multi-class classification. In addition, we conduct ablation experiments to demonstrate the effect of different types of context, the edge attention mechanism and the number of attention layers.

The rest of this paper is organized as follows. In Section 2, we briefly review works that are related to this paper. In Section 3, we present our model for online stroke classification. In Section 4, we introduce the training details of our model. In Section 5, the experiments and results are presented. Finally, we draw conclusions and discuss future directions in Section 6.

2. Related Works

Online stroke classification. The task of separating strokes into different categories in online documents has been researched for a long period. In the early years, this task was addressed mainly by extracting local features and building classifiers on those geometric features, such as [6], [7], [12]. With the popularization of structured prediction methods, methods based on Markov random fields or conditional random fields were proposed to capture various types of label dependencies, such as [1], [2], [18], [20]. Different works exploited different types of label dependencies and the temporal and spatial dependencies are the most adopted two. Moreover, Delaye et al. [1] also introduced some other types of dependencies that may be helpful to classification, like intersection, lateral and stroke continuation dependencies. Another path for addressing this problem is based on recurrent neural networks. Bidirectional long-short term memory (BLSTM), one of the most powerful variants of RNN, were employed in [5], [14] to exploit the temporal information in the online documents. Van et al. [14] extracted global and local context features and ensembled multiple BLSTM neural networks, achieving state-of-the-art accuracy on the text/non-text classification problem on the IAMonDo dataset.

Graph neural networks. Graph neural networks (GNN) are connectionist models that can capture the dependencies of graphs via message passing between the nodes of graphs [19]. In the past, variants of GNN models were mostly used in the setting of semi-supervised learning [9], graph classification [10] as the graph structure in these fields is usually defined by nature. A notable variant of GNN is graph attention networks (GAT), which was first proposed in [16]. In this model, except for the graph convolution operation, it also includes the self-attention mechanism [15] to evaluate the individual importance of the adjacent nodes and therefore it can be applied to graph nodes having different degrees by specifying arbitrary weights to the neighbors [16].

3. Method Description

In this work, we formulate the online stroke classification as a node classification problem in the document graph. In the following, we first introduce the problem definition of stroke classification, then describe the pipeline of our model.

3.1. Problem Definition

We are given a set of labeled online documents $S = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), i = 1, \dots, N\}$, in which each document $\mathbf{x}^{(i)}$ is represented by a sequence of strokes $\{\mathbf{x}_t^{(i)}, t = 1, \dots, T_i\}$ and each stroke has one associated label $\mathbf{y}_t^{(i)} \in \{0, \dots, L-1\}$, where L is the number of classes. Our goal is to learn a model from the training set S that can predict strokes in test documents with high accuracy.

3.2. Proposed Model

Figure 2 illustrates the framework of our model. Our model consists of three modules, including the construction of the document graph, extraction of node and edge features, and graph attention networks.

3.2.1. Document graph

In this section, we introduce the definition of the *document graph* used in this work. A document graph is a 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) .

To build the document graph, we consider two major types of contextual information, namely temporal context and spatial context.

Since the online nature of the data permits us to see the document as a sequence of strokes, the temporal neighborhood captures the interaction between strokes that are written successively in the document. To exploit this dependency, we define our temporal neighborhood system by considering every temporally adjacent stroke.

On the other hand, the spatial context is also a vital source of knowledge as spatially adjacent strokes tend to have same labels. We define our spatial neighborhood system by considering that two strokes \mathbf{x}_i and \mathbf{x}_j are spatial adjacent if their minimal distance $d(\mathbf{x}_i, \mathbf{x}_j)$ is below a given threshold T_1 . We set the hyper-parameter $T_1 = 10$ same as in [1].

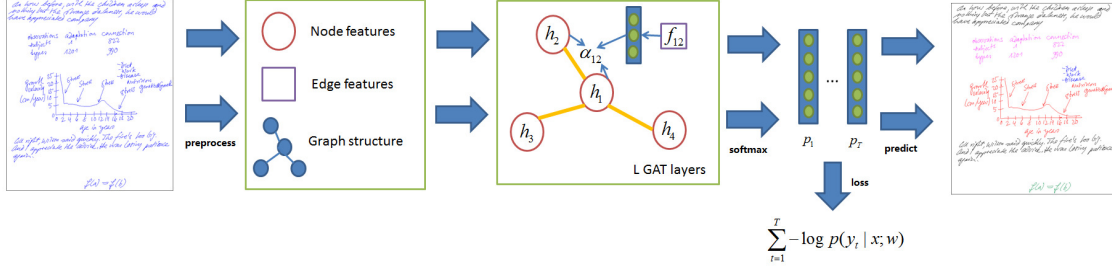


Figure 2. An illustration of the training and prediction process for one input document. An input document is first preprocessed to generate node features, edge features and the document graph, then these components are fed into networks stacking with multiple graph attention layers to learn context-aware features. During the training, the model is learned by the cross entropy loss. When performing prediction, the class with maximum probability is chosen for each stroke.

TABLE 1. 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

3.2.2. Node features

For extracting the node features in the graph, we follow the similar procedure in [1], [14], [18], extracting 13 contour-based shape features and 10 context features. The shape features are extracted from each stroke directly, while the context features are extracted by considering its interactions with neighboring strokes. All these node features are listed in Table 1.

3.2.3. Edge features

For evaluating different importance of contextual information carried by different neighbors, we also extract 19 edge features between strokes that are adjacent in the graph. These features calculate the distance between pairs of strokes with different distance metrics and can be regarded as the indicators to measure similarity between two strokes. Therefore, it can be used to learn gates for controlling the message passing routine between nodes, which is beneficial

TABLE 2. Edge features extracted from a pair of stroke x_i, x_j .

#	Description
1	Minimum distance between 2 strokes
2	Minimum distance between the endpoints of 2 strokes
3	Maximum distance between the endpoints of 2 strokes
4	Distance between the centers of the 2 bounding boxes of 2 strokes
5	Horizontal distances between the centroids of 2 strokes
6	Vertical distances between the centroids of 2 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	Ratio of off-stroke distance to temporal distance
11	Ratio of off-stroke distance on X, Y axes to temporal distance
12	Ratio of area of the largest bounding box of 2 strokes to their union
13	Ratio of widths of the bounding boxes of 2 strokes
14	Ratio of heights of the bounding boxes of 2 strokes
15	Ratio of diagonals of the bounding boxes of 2 strokes
16	Ratio of areas of the bounding boxes of 2 strokes
17	Ratio of lengths of 2 strokes
18	Ratio of durations of 2 strokes
19	Ratio of curvatures of 2 strokes

for nodes to collect useful contextual information. All these edge features are listed in Table 2.

3.2.4. Graph attention networks

In this section, we present the *graph attention layer*, which is stacked to construct the graph attention networks.

The input to each graph attention layer is a set of node features $\mathbf{H} = \{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_N\}$, $\mathbf{h}_i \in \mathbb{R}^C$ and a set of edge features $\mathbf{F} = \{\mathbf{f}_{ij}, (i, j) \in E\}$, $\mathbf{f}_{ij} \in \mathbb{R}^D$, where N is the number of nodes, E is the set of edges and C, D are the number of features in each node and each edge, respectively. The layer outputs a new set of node features $\mathbf{H}' = \{\mathbf{h}'_1, \mathbf{h}'_2, \dots, \mathbf{h}'_N\}$, $\mathbf{h}'_i \in \mathbb{R}^{C'}$, where C' is the number of features of outputs.

In the layer, the first step is to apply a shared linear transformation to every node, then perform self-attention on the nodes by a shared attentional mechanism $a : \mathbb{R}^{C'} \times \mathbb{R}^{C'} \rightarrow \mathbb{R}$.

$$s_{ij} = a(\mathbf{W}_h \mathbf{h}_i, \mathbf{W}_h \mathbf{h}_j). \quad (1)$$

where $\mathbf{W}_h \in \mathbb{R}^{C' \times C}$ is a learnable parameter. The attention mechanism a used in this work is the additive attention

which is defined as

$$a(\mathbf{W}_h \mathbf{h}_i, \mathbf{W}_h \mathbf{h}_j) = \sigma(\mathbf{v}^T (\mathbf{W}_h \mathbf{h}_i + \mathbf{W}_h \mathbf{h}_j)) \quad (2)$$

where $\mathbf{v} \in \mathbb{R}^{C'}$ is a learnable parameter and $\sigma(\cdot)$ is leaky ReLU activation function.

Except for calculating the importance score between nodes by self-attention mechanisms, we also calculate another attention score using edge features through one layer neural network.

$$s'_{ij} = \sigma(\mathbf{v}_f^T \sigma(\mathbf{W}_f \mathbf{f}_{ij} + \mathbf{b}_f)). \quad (3)$$

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.

To make coefficients comparable across different nodes, the scores should be normalized across all choices of j using the softmax function with temperature β :

$$\begin{aligned} \alpha_{ij} &= \text{softmax}_{\beta}(s_{ij} + s'_{ij}) \\ &= \frac{\exp(\beta(s_{ij} + s'_{ij}))}{\sum_{k \in \mathcal{N}_i} \exp(\beta(s_{ik} + s'_{ik}))}. \end{aligned} \quad (4)$$

When $\beta = 0$, the model reduces to the vanilla graph convolutional network.

After obtaining the attention coefficients, we can use them to compute a weighted combination of the neighborhood features, to produce the final output features for every node.

$$\mathbf{h}'_i = \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} \mathbf{W}_h \mathbf{h}_j \right). \quad (5)$$

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

Inspired by [16], we also introduce *multi-head attention* in our model. Specifically, K independent attention mechanisms execute the transformation in Equation (5) and their outputs are concatenated.

$$\mathbf{h}'_i = \parallel_{k=1}^K \sigma \left(\sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}_h^k \mathbf{h}_j \right). \quad (6)$$

where \parallel represents concatenation.

And in the last layer, the concatenation operation is replaced by averaging operation and then apply softmax function:

$$\begin{aligned} \mathbf{h}'_i &= \sigma \left(\frac{1}{K} \sum_{k=1}^K \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k \mathbf{W}_h^k \mathbf{h}_j \right) \\ \mathbf{p}_i &= \text{softmax}(\mathbf{W}_o \mathbf{h}'_i) \end{aligned} \quad (7)$$

where $\mathbf{W}_o \in \mathbb{R}^{C' \times L}$ is the learnable weight matrix to transform features to outputs.

Finally, we also employ some standard tricks to stabilize and accelerate the training procedure, including residual connection and batch normalization. In this work, all the hyper-parameters of hidden layers are the same, and the residual connection is added except the input layer.

$$\mathbf{h}'_i = \text{BatchNorm}(\mathbf{h}_i + \text{Layer}(\mathbf{h}_i)) \quad (8)$$

4. Network Training

In this section, we provide details about training the proposed graph attention networks. We implement the graph attention network using the dgl library¹ and its pytorch backend. The experiments are run on a GeForce GTX 1080Ti. Using the settings discussed in this section, the model training requires about 10 minutes for the text/non-text classification task and 20 minutes for the multi-class classification task.

4.1. Parameter Initialization

Weight matrix parameters are randomly 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 [3].

4.2. Loss and Optimization

The loss we use in this work is the standard cross entropy loss, which is defined as

$$L(\mathbf{W}) = - \sum_{i=1}^N \sum_{t=1}^{T_i} \log \mathbf{p}_t[y_t^{(i)}] \quad (9)$$

Parameter optimization is performed with Adam [8] with batch size 16. We choose the initial learning rate $\eta_0 = 0.005$ and the learning rate decays according to the validation accuracy. The decay rate $\rho = 0.1$ and the number of patience round $r = 10$. We use early stopping based on the performance on the validation set and the best parameter is chosen by the best accuracy on the validation set.

4.3. Regularization

To mitigate overfitting, the dropout method [13] is applied in the input of each layer to regularize our model. We fix dropout rate at 0.2 for all dropout layers.

4.4. Tuning Hyper-Parameters

Table 3 summarizes the chosen hyper-parameters for all experiments. We tune the hyper-parameters on the development sets by random search. To show the robustness of our model, we try to share as many hyper-parameters as possible among two tasks except the number of attention layers.

TABLE 3. Hyper-parameters for all experiments.

Hyper-parameter	Binary	Multi
hidden layer size	32	32
number of heads	8	8
number of heads in output layer	2	2
number of layers	5	10
dropout rate	0.2	0.2
batch size	16	16
initial learning rate	0.005	0.005
decay rate	0.1	0.1
number of patience round	10	10
temperature	0.5	0.5

1. <https://github.com/dmlc/dgl>

5. Experiment Results

5.1. IAMonDo Dataset

We conduct experiments on the IAMonDo dataset [6], a publicly available collection of freely handwritten online documents. The dataset consists of 1000 documents, mixing texts, drawing, diagrams, formulas, tables, lists and marking elements arranged in an unconstrained way. The dataset is split into five disjoint sets, each containing roughly 200 documents. We follow the traditional data partitioning of the dataset [6]. Documents from *set0* and *set1* constitute the training set, *set2* is the validation set, and *set3* is the independent test set. We perform two sets of experiments to evaluate the performance of our model. In the first set of experiments, strokes are classified into text or non-text, and the corresponding ground truth labels are derived as suggested in [4]. In the second set, a more complicated task is considered to separate strokes into graphic/text/table/list/math five classes as in [2]. Table 4 presents the statistics of IAMonDo dataset. To reduce the volatility of the system, we conduct each experiment 10 times and report the mean accuracy and standard deviation for each model.

TABLE 4. Statistics of IAMonDo dataset: number of documents, strokes and strokes per category.

Set	Training	Validation	Test
Docs	403	200	203
Strokes	143348	68725	70927
Text	116801	57821	57959
Non-text	26547	10904	12968
Graphic	37496	15481	17488
Text	79812	39796	40469
Table	13044	6562	6883
List	6337	3474	3115
Math	6659	3412	2972

5.2. Compared with Previous Work

5.2.1. Text/non-text classification

Table 5 illustrates the results of our model for text/non-text classification, together with seven previous top-performance systems for comparison. Our model significantly outperforms all the single models based on CRF or BLSTM. Moreover, our model achieves 0.29% improvements on accuracy over the model in [14] which is an ensemble model of several BLSTM networks. It proves the effectiveness of graph attention networks to integrate different context information. Comparing with Delaye [1] that has the similar concept to explore multiple context information with CRF, our model is more efficient to exploit the interaction between neighborhood strokes.

5.2.2. Multi-class classification

Table 6 compares the results of our model for multi-class classification together with the previous work in [2]. Our model also achieves 1.18% improvements on accuracy over the model in [2]. In [2], Delaye proposed a hierarchical CRF framework and use distance learning to construct the tree CRF structure. It shows that his model based on CRF could not support the arbitrary graph structure and a massive

TABLE 5. Performance of different methods for text/non-text stroke classification on the IAMonDo dataset. An * symbol indicates the method is ensembled by multiple classifiers.

Method	Description	Accuracy
Indermühle [6]	Local classification with SVM	91.30
Weber* [17]	Multiple classifiers system	97.00
Indermühle [5]	BLSTM network	97.01
Delaye [1]	CRF with multiple context	97.21
PCC [14]	Local context integrated by BLSTM	97.96
Ye [18]	Joint training of MLP and CRF	97.96
Van* [14]	Global and local BLSTM ensembled	98.30
GCN	Graph convolutional networks	97.21±0.07
GAT(w/o edge)	GAT without edge features	97.87±0.05
GAT(w/o space)	GAT without spatial edges	98.45±0.07
GAT	Graph attention networks	98.59±0.06

performance decrease occurred when the CRF graph is no longer the tree. On the contrary, our model can build the graph with the arbitrary structure, and the increasing number of edges brings performance improvements.

TABLE 6. Performance of different methods for multi-class stroke classification on the IAMonDo dataset.

Method	Description	Accuracy
Delaye [2]	Hierarchical CRF with loopy structure	79.22
Delaye [2]	Hierarchical CRF with tree structure	93.46
GCN	Graph convolutional networks	88.92±0.27
GAT(w/o edge)	GAT without edge features	92.28±0.17
GAT(w/o space)	GAT without spatial edges	91.94±0.16
GAT	Graph attention networks	94.64±0.29

5.3. Ablation Study

To discuss the vital elements in the success of our proposed model, we conduct an ablation study on three aspects. Specifically, we have tested the performance of models without spatial edges, edge attention mechanisms or calculating attention scores without edge features. In addition, we also compare the performances between different numbers of attention layers.

5.3.1. Effect of spatial edges

In Table 5 and Table 6, except for the best model GAT, we also conduct experiments on three simpler models. All hyper-parameters remain the same as in Table 3.

GAT(w/o space) is our model without the spatial edges in the document graph. We can see that the performance decreases to some extent in both tasks but the decline is more serious in the multi-class scenario, which shows that the spatial information is crucial in the more complicated multi-class classification task and it is beneficial to add this prior knowledge to the model.

5.3.2. Effect of attention

In this experiment, we discuss the effect of attention mechanisms used in the model and degrades our model to two simpler model GCN and GAT(w/o edge).

GCN is the vanilla graph convolutional network model that aggregates features from the neighborhood by simple average, which can be instantiated by setting $\beta = 0$ in the equation (5).

GAT(w/o edge) is the model that calculates attention coefficients without using edge features, which can be instantiated by discarding s'_{ij} term in the equation (5).

We can see that both the self-attention mechanism operating on node features and the extra information provided by edge features contribute a lot to the success of the model. Without these two modules, the performance of a vanilla graph convolutional network is far less than satisfactory.

5.3.3. Effect of the number of attention layers

In this experiment, we discuss the effect of the number of attention layers. Like convolutional layers used in computer vision or natural language processing, every attention layer in this model gathers contextual information from the adjacent nodes, and more attention layers will learn features with larger sizes of receptive fields that is helpful for classifying strokes.

In table 7, we can see that the number of attention layers is indeed a critical hyperparameter that can significantly influence the overall performance and the performance boosts with the progressively increasing number of layers.

TABLE 7. Performance of different number of attention layers.

Layer	Binary	Layer	Multi
1	97.96±0.07	2	89.77±0.20
2	98.38±0.06	4	92.71±0.16
3	98.48±0.06	6	93.72±0.14
4	98.55±0.06	8	94.44±0.21
5	98.59±0.06	10	94.64±0.29

6. Conclusion

In this paper, we present a novel framework based on graph attention networks for stroke classification in online written documents. The representation of features is learned by graph convolution and attention mechanisms in a document graph which is built by exploiting different types of contextual information. We discuss the effects of different techniques used in the message passing routine, demonstrating the importance of spatial edges, attention mechanisms and the usage of edge features. With this model, we achieve state-of-the-art performances on two stroke classification tasks of IAMonDo dataset.

Acknowledgement

This work is supported by the National Key Research and Development Program Grant 2018YFB1005000, the National Natural Science Foundation of China (NSFC) Grants 61721004, 61773376, 61836014, the Beijing Science and Technology Program Grant Z181100008918010.

References

[1] Adrien Delaye and Cheng-Lin Liu. Contextual text/non-text stroke classification in online handwritten notes with conditional random fields. *Pattern Recognition*, 47(3):959–968, 2014.

[2] Adrien Delaye and Cheng-Lin Liu. Multi-class segmentation of free-form online documents with tree conditional random fields. *International Journal on Document Analysis and Recognition*, 17(4):313–329, 2014.

[3] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010.

[4] Emanuel Indermühle. *Analysis of Digital Ink in Electronic Documents*. PhD thesis, University of Bern, 2012.

[5] Emanuel Indermühle, Volkmar Frinken, and Horst Bunke. Mode detection in online handwritten documents using blstm neural networks. In *Proceedings of the International Conference on Frontiers in Handwriting Recognition*, pages 302–307, 2012.

[6] Emanuel Indermühle, Marcus Liwicki, and Horst Bunke. Iamondo-database: an online handwritten document database with non-uniform contents. In *Proceedings of the International Workshop on Document Analysis Systems*, pages 97–104, 2010.

[7] Anil K Jain, Anoop M Namboodiri, and Jayashree Subrahmonia. Structure in on-line documents. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 844–848, 2001.

[8] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[9] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[10] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 1666–1674, 2018.

[11] Xiao-Hui Li, Fei Yin, and Cheng-Lin Liu. Printed/handwritten texts and graphics separation in complex documents using conditional random fields. In *Proceedings of the International Workshop on Document Analysis Systems*, pages 145–150, 2018.

[12] Eric Jeffrey Peterson, Thomas F Stahovich, Eric Doi, and Christine Alvarado. Grouping strokes into shapes in hand-drawn diagrams. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 10, page 14, 2010.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[14] Truyen Van Phan and Masaki Nakagawa. Combination of global and local contexts for text/non-text classification in heterogeneous online handwritten documents. *Pattern Recognition*, 51:112–124, 2016.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[16] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[17] Markus Weber, Marcus Liwicki, Yannik TH Schelske, Christopher Schoelzel, Florian Strauß, and Andreas Dengel. Mcs for online mode detection: Evaluation on pen-enabled multi-touch interfaces. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 957–961, 2011.

[18] Jun-Yu Ye, Yan-Ming Zhang, and Cheng-Lin 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*, pages 3264–3269, 2016.

[19] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[20] Xiang-Dong Zhou and Cheng-Lin 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*, volume 1, pages 377–381, 2007.