Comparative Analysis of Deep Learning-Based Models in Sketch Recognition

Fazhan Liu[1], Minxiao Lu[2], Wei Zhang[3, *]

[1]School of English and Software Engineering, Dalian Jiaotong University, Liaoning,116000, China
[2]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, 100000, China
[3]School of Artificial Intelligence, Hebei University of Technology, Tianjin, 300401, China
*Corresponding Author: 215306@stu.hebut.edu.cn

Abstract. People usually use hand-drawn sketches to draw simple lines to express or record their ideas and intentions and create images or videos by hand-drawing. For some people who are not specialized in the field of art, the sketches of the same object will vary depending on the artistic style and drawing ability. As a result, they are often highly abstract, which makes the automatic recognition of sketches more difficult compared to other fields. Due to its visualization, sketch recognition has become an important hotspot problem, which effectively improves the efficiency and diversity of generation compared with the traditional manual creation method and becomes an important research direction within computer vision and graphics, and plays a crucial role in the fields of design and visual creation. Existing recognition approaches depend on hand-drawn features and depth features are deficient in recognizing their local information. By using a dataset QuickDraw, which consists of 345 categories with 50 million vector drawings released by Google, this work applies Graph Neural Networks (GNN) to improve the performance, improves the recognition accuracy of targeting sketches drawn by different people.

Keywords: Graph Neural Networks, Sketch Recognition, Computer Vision.

1 Introduction

Human-beings began using sketches a long time ago to express their lives or to show their emotions, from paintings left in caves by ancestors to portraits of various schools of thought in the Renaissance. Nowadays, as the fever of paperless office is gradually rising, the popularity of touching screen devices has led to a boom in applications related to sketch recognition. Sketch recognition plays a very important role in many fields, especially in design, engineering, education, and medicine. Sketch recognition technology allows people to convert hand-drawn sketches into digital form, which can then be analyzed, modified, and shared [1]. For example, in the medical field, sketch recognition can be used to analyze and diagnose medical images. Doctors can mark

Y. Wang (ed.), *Proceedings of the 2024 International Conference on Artificial Intelligence and Communication (ICAIC 2024)*, Advances in Intelligent Systems Research 185,
https://doi.org/10.2991/978-94-6463-512-6_34

lesions or describe anatomical structures through hand-drawn sketches, and then use sketch recognition technology to convert them into digital form for further analysis and diagnosis.

Sketches are usually abstract and stylized due to the increasing integration of non-art majors into the industry and their varying styles and levels of drawing. There is a big difference between sketches and traditional images; sketches are composed of chronological ratios, whereas images are composed of static groups of pixels with intricate color and texture patterns. Compared to a typical image, a sketch captures a visual item at a high level of abstraction with relatively little information., which makes sketch modeling very challenging [2,3].

Deep learning is a representation learning method that maps data from low-level features to high-level abstract concepts through multilevel nonlinear transformations [4]. Firstly, using deep learning to solve sketch recognition allows for better learning of complex features, deep learning models are capable of learning and extracting complex representations of input, which is important for tasks such as sketch recognition, which has diverse and abstract features. Second, it has strong adaptability, deep learning models are usually able to adapt to different types and styles of input data after being trained on large-scale datasets, which makes sketch recognition systems more versatile. Finally, deep learning has superior performance, deep learning has been used in a wide variety of domains, including sketch recognition, as it has achieved great success with superior performance on many visual and speech tasks. In this paper, the deep learning-based Graph Neural Network (GNN) method will be used to compare with Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) methods and discuss their advantages and disadvantages.

2 Methods

2.1 GNN Model

GNN Overview. GNN is a machine learning model for working with graph data. In traditional neural networks, the inputs are fixed-length vectors, whereas in graph data, each sample may be a graph where the number of nodes and edges varies. The goal of GNN is to learn the relationships between nodes in a graph structure and to feature representations of the nodes in order to carry out diverse tasks. The GNN is a graph that performs an optimizable transformation, its input is a graph and its output is also a graph. It transforms only the attribute vectors (i.e., V, E, U), but it does not change the connectivity of the graph. After obtaining the optimized attribute vectors, it is then connected to a fully connected neural network for classification and regression according to the actual task [5].

The original GNN input graph data contains nodes and undirected edges with labeling information. Graph data is data consisting of nodes and edges. Nodes can represent various objects while edges represent associations, interactions, or dependencies between nodes.

Tasks on graph data are three folds, including graph-, edge- and node-level tasks. (1) Graph level tasks. The tasks at the graph level do not depend on the properties of a node or an edge, but rather on the overall structure of the graph to realize tasks such as classification, representation, and generation. (2) Edge level tasks. The tasks at the edge level consist mainly of edge classification and prediction tasks. Edge classification refers to the prediction of a certain property of an edge; edge prediction refers to whether an edge will be formed between two given nodes. (3) Node level tasks. Node-level tasks consist mainly of classification tasks and regression tasks. These types of tasks, although predicting node-level properties, should obviously not be based on a single node, and the relationships of the nodes need to be considered.

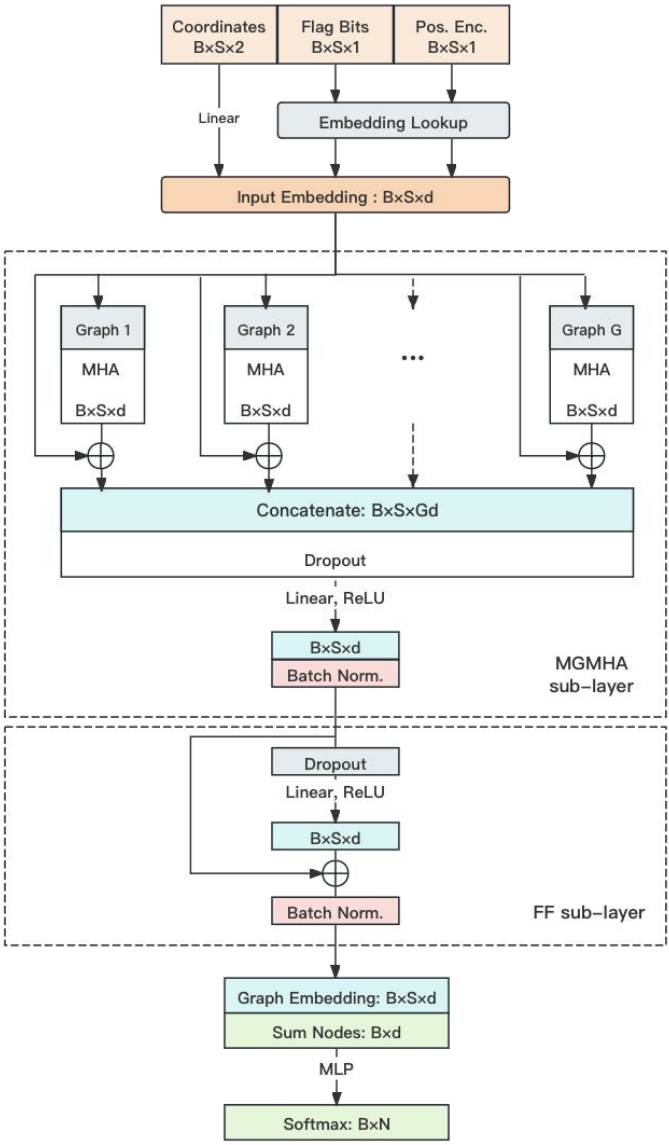

Fig. 1. Structure of Multi-Graph Transformer Network [6].

Multi-Graph Transformer (MGT). MGT is a graph neural network-based model designed to process multiple graph data [6]. Unlike traditional GNN, MGT is able to process multiple graph structures simultaneously, which makes it useful for an extensive variety of purposes in areas such as social network analysis, bioinformatics, and finance.

MGT is a Transformer-based model exploiting the self-attention mechanism to handle the relationships of nodes and edges between different graphs. Its structure is shown in Fig. 1. In this way, MGT is capable of exchanging and integrating information among multiple graphs, thus improving the ability to model complex relationships. The advantage of this approach is that it can effectively capture the interactions between multiple graphs and is able to share parameters across different graph structures, thus improving the generalization ability of the model. This enables MGT to achieve better performance when dealing with multiple graph data and is highly scalable.

The paper adopts Google QuickDraw data, and each node is represented as a 4-dimensional vector, the first two bits are the horizontal and vertical coordinates of the node on the canvas, the third bit is the flag bit used to describe the state of the brush, and the fourth bit is the position code. The horizontal and vertical coordinates are upscaled by a linear layer, and the flag bits and position encoding are upscaled by an embedding layer, which are then concatenated to form the input of the MGT.

As shown in Fig. 1, overall, MGT is an L-layer structure, and each layer has two components: the Multi-Graph Multi-Head Attention (MGMHA) and the position-wise fully connected Feed-Forward (FF).

The MGMHA sublayer of the paper is a multiplexed parallel structure, each of which is a Multi-Head Attention module based on a graph structure. The graph structure here is the graph structure defined by the paper based on the domain knowledge of hand-drawn sketches, i.e., the multiple adjacency matrices defined in the original paper. These adjacency matrices are used to describe the connectivity between nodes on each hand-drawn sketch. In turn, the connectivity described by the adjacency matrices is used in the Multi-Head Attention operation to control the connectivity in the Attention Score Matrix, allowing or blocking out attentional relationships between specific nodes. The FF sublayer mainly performs operations such as residual connectivity and batch normalization.

Given a sketch, after MGT, each of its nodes is marked as a vector, and the representation vectors of these nodes are summed up as the vector representation of the sketch. The summing process does not consider the extra nodes introduced by the padding operation during data preprocessing. The classifier at the tail end of the network is implemented by a multilayer perceptron using a SoftMax cross-entropy loss function.

2.2 CNN Model

CNN Overview. CNN exploits a depth structure and convolutional operation, which is frequently used in the fields of voice recognition, natural language processing, and recognition of images [7].

Structurally, it contains several layers. (1) The input layer is usually the original or pre-processed information input to the convolutional neural network, which could refer to the original three-dimensional, colorful image in this field of image recognition. (2) Convolutional layers are typically employed to extract features from the input information which is received from the input layer and it is an abstraction of

the hidden correlation in the original data through the convolution kernel matrix. (3) The activation layer is responsible for applying a non-linear function to the features recovered by the convolution layer, hence enhancing their representation. The convolution operation involves multiplying the input matrix with the convolution kernel matrix, which creates a linear relationship. To introduce non-linearity, the activation layer is used to perform a non-linear mapping on the result. (4) The pooling layer is used to analyze the features in the sensing domain and extract the most indicative features in the region, leading to a reduction in the output feature scale and consequently minimizing the model size. (5) The fully connected layer could combine features obtained from the training of the CNN and transforming the multidimensional feature. The high dimension indicates the sample batch, whereas the low dimension often corresponds to the task's target.

MobileNet v2. MobileNet v2 proposes a new variant structure, as demonstrated in Fig. 2, on the basis of depth-separable convolution: Inverted bottleneck [8].

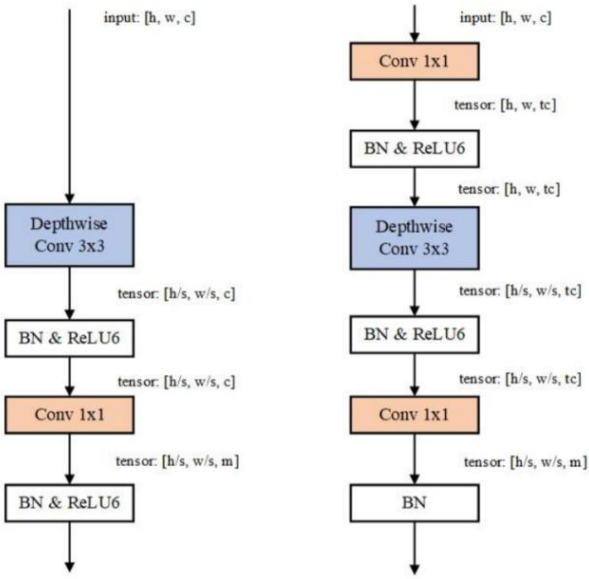

Fig. 2. Structure of Depth-separable convolution (left) compared with Inverted bottleneck structure (right) [9].

In Fig.2, the variables h and w represent the dimensions of the input feature, namely its height and width, the variable c represents the quantity of channels. t denotes channel. s is the convolution kernel step.

The main changes to the Inverted bottleneck compared to depth-separable convolution are as follows: (1) An additional 1×1 convolution before the deep convolution is exploited to expand the channel by t times. The effect of this is to increase the information dimension (number of channels) of deep convolution processing, essentially increasing the expressiveness of the model. (2) The last 1×1 convolution usually has a lower output dimension and is not followed by an activation

function. (3) When the input and output feature maps are the same size, an additional jump connection is integrated between the input and output.

The core assumption of Inverted bottleneck is that information storage can depend on low-dimensional space; However, the computation and processing need to be in high dimensional space.

Corresponding to this variation, the core is computed in higher dimensions (deep convolution), while the final information output is in lower dimensions (last 1×1 convolution). Because the final output is for information storage and transmission, the activation function is not connected (It is non-linear and destructive from the point of view of data compression alone).

MobileNet v2 is stacked based on Inverted bottleneck. Its detailed structure is demonstrated in Table 1.

Table 1. MobileNet v2 network structure table

Input	Operator	t	c	s
224×224×3	conv2d	-	32	2
112×112×32	bottleneck	1	16	1
112×112×16	bottleneck	6	24	2
56×56×24	bottleneck	6	32	2
28×28×32	bottleneck	6	64	2
14×14×64	bottleneck	6	96	1
14×14×96	bottleneck	6	160	2
7×7×160	bottleneck	6	320	1
7×7×320	conv2d 1×1	-	1280	1
7×7×1280	avgpool 7×7	-	-	-
1×1×1280	conv2d 1×1	-	k	-

2.3 RNN Model

RNN Overview. RNNs are a specific category of neural networks that are designed to process sequential data. RNN execute recursive operations alongside the sequence, with all nodes interconnected in a chain-like fashion. They possess extensive applications in several domains, encompassing natural language processing, speech, and image analysis.

Just as humans using their past memories to better understand the world, RNN implement a similar mechanism to the human brain by retaining certain memories of the information they process. This capability facilitates time series analysis, in contrast to other types of neural networks that do not retain processed information.

A sketch can be understood as a collection of key points composed of strokes, where the coordinate sequence of strokes in the sketch is temporally ordered. This temporal sequence creates conditions for using RNN methods for sketch recognition.

BiGRU Model. Building on the foundation of RNN, the Gated Recurrent Unit (GRU) model introduces gating mechanisms to address the vanishing gradient problem and

enhance model performance. GRU units primarily consist of an update gate and a reset gate. These gates receive the prior latent state and the current timestamp to obtain the updated hidden state and output. The GRU model is illustrated in Fig. 3.

Bidirectional GRU (BiGRU) model builds upon the GRU model by introducing a bidirectional structure, which better captures the bidirectional dependencies in sequential data [10]. The architecture of BiGRU model is illustrated in Fig. 4.

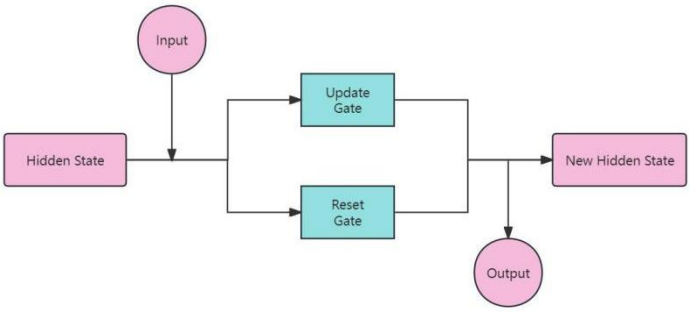

Fig. 3. The structure of GRU Model [10].

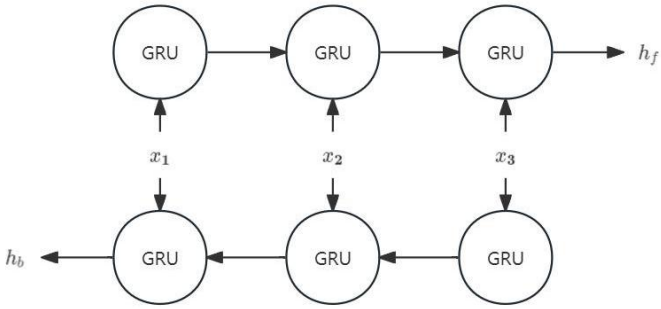

Fig. 4. The structure of BiGRU Model [10].

Firstly, for the input sequence $X=[x_1,x_2,...,x_T]$, the forward hidden state h_f of BiGRU can be computed as:

$$h_f=GRU(h_{t-1},x_t) \tag{1}$$

Next, the backward hidden state h_b can be computed as:

$$h_b=GRU(h_{t+1},x_t) \tag{2}$$

Here, GRU represents the mapping of the GRU unit. h_{t-1} and h_{t+1} represent the hidden states of the current GRU unit in the forward and backward directions respectively. x_t is the t-th element of the input sequence $t\in[1,T]$, and T represents the sequential length.

Then, the concealed states from both directions are concatenated to form the final hidden state:

$$h_t=[h_f,h_b] \tag{3}$$

In the end, the state is transmitted to a fully linked layer in order to acquire the output:

$$y_t=\text{softmax}(Wh_t+b) \tag{4}$$

Here, W and b represent the weights and biases of the fully connected layer, and the activation function is softmax.

The BiGRU model considers the dependencies in both directions of the sequence data, thereby improving the model's capacity to understand sequential data and capturing rich contextual information. Additionally, the model has fewer parameters, typically trains faster, and requires a relatively small amount of data to generalize effectively.

3 Experiments and Results

3.1 Dataset

With 50 million sketches, Google's QuickDraw is presently the largest sketch dataset accessible [11]. Some example sketches are shown in Fig. 5. Unlike smaller sketch datasets, the samples in QuickDraw were gathered through an international online game that requires users to sketch various categories of objects in seconds, such as apples, frogs, clocks, airplanes, and so on. Consequently, QuickDraw's sketches can be quite abstract and incorporate a variety of drawing styles, making it a useful testing platform for comparing the performance of various neural network designs for sketch recognition.

From the QuickDraw dataset, 345 categories of sketches are selected, with 345,000 training samples, 34,500 testing samples, and 34,500 validation samples.

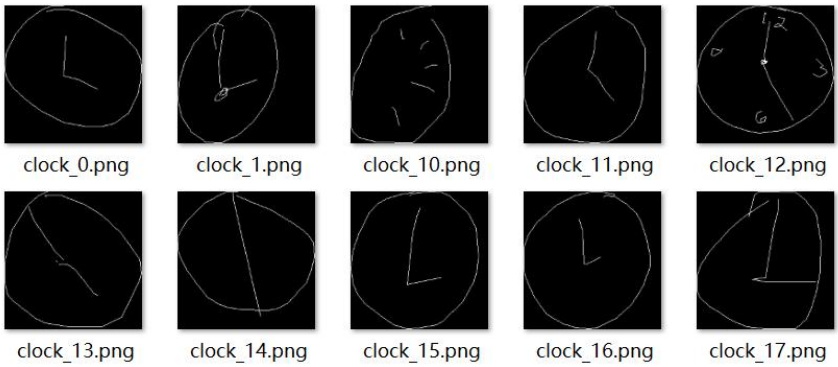

Fig. 5. Examples of sketch images [11].

3.2 Training Details

The evaluation metric used for this research is top K accuracy, which represents the percentage of samples being correctly predicted among the predictions of the top K categories made by the model, where k = 1, 5, 10 (acc.@k represents the corresponding accuracy). Adam served as the optimizer in this work, and the loss function was softmax cross-entropy loss. The batch sizes for RNN, GNN, and CNN models were 256, 128, and 64, respectively. The GNN model underwent 200, 15, and 5 rounds of iterations. Additionally, this work implemented model loading and saving based on checkpoint mechanisms to record the highest testing performance of models.

This experiment was implemented in PyTorch and executed on a computer with Windows 11 and Nvidia RTX 3060 GPU.

3.3 Results

Experimental results are demonstrated in Table 2. Regarding accuracy, the sketch recognition algorithm based on GNN achieved an acc.@1 of 69.0%, surpassing the RNN-BiGRU algorithm (51.2%) and slightly below the more mature CNN-MobileNet v2 algorithm (71.0%) and ordinary human performance (73.1%); acc.@5 was 89.5%, and acc.@10 was 93.1%, both higher than the RNN-BiGRU algorithm (78.2%, 85.7%) and comparable to the CNN-MobileNet v2 algorithm (90.8%, 93.8%).

Table 2. Comparison of model training time and accuracy for different algorithms.

Algorithm	Model Training Time	acc.@1	acc.@5	acc.@10
RNN-BiGRU	200 epoches / 18 h	51.2%	78.2%	85.7%
GNN-MGT	15 epoches / 53 h	69.0%	89.5%	93.1%
CNN-MobileNet v2	5 epoches / 28 h	71.0%	90.8%	93.8%
Human	-	73.1%	-	-

4 Discussion and Perspective

By training and testing sketch data sets, it could be found out that GNN can effectively capture the structural information between sketches and achieve high recognition accuracy. Specific experimental conclusions are as follows: (1) In terms of the sample proportion (k=1) of the actual category as determined by the model's initial prediction, the accuracy rate of the sketch recognition algorithm based on GNN model is 69.0%, which is similar to but slightly lower than that of ordinary human (73.1%). (2) Considering the complexity and challenge of the sketch recognition task, the accuracy rate of the sketch recognition algorithm based on GNN model reaches more than 90% when k=10, indicating that among the 345 sketch categories, the top 10 characters with the highest probability of the recognition results given by the algorithm have a small deviation from the real categories, and can identify the sketch more accurately in most cases. It has high reliability and stability. (3) In terms of the

accuracy of recognition, no matter which value of k is taken, it is CNN>GNN>RNN. (4) In terms of training time, GNN model and CNN model have higher computational complexity and memory consumption, and the training time is obviously longer than that of RNN model.

Ordinary human beings have intuitive cognitive ability and rich context understanding in the recognition of sketches, and can infer the objects represented by sketches according to context, experience and background knowledge. While machine learning algorithms such as GNN focus on learning features and patterns from large amounts of data, with strong generalization ability and the ability to deal with complex structures. Due to the lack of hardware conditions, the sketch recognition algorithm based on GNN mentioned in this paper does not get a good convergence result.

Shortcomings in the experiment: 1. Due to hardware performance problems, the models may not be able to achieve better convergence results within a certain number of iterations. 2. There are not enough comparison models to better reflect the application effect of each model algorithm in the field of sketch.

To solve the above problems, following improvements are suggested: 1. Continue to adjust the model parameters, and use a more powerful computer or cloud platform to train the model. 2. Increase the number of comparison models.

When k=1, the accuracy of GNN algorithm does not exceed the recognition accuracy of ordinary human beings. In the future, the authors will continue to train the model to achieve higher accuracy and strive to be equal to or exceed the accuracy of human beings. The accuracy of GNN algorithm is lower than that of more mature CNN. In the future, the authors will continue to optimize the specific details of the algorithm and strive to surpass CNN algorithm. Due to the high computational complexity and memory consumption, GNN model training takes a long time. The authors will continuously improve the algorithm to achieve a lower time complexity.

In general, it is expected that GNN-based sketch recognition algorithms will have a positive impact in many fields in the future, and according to the methodologies and findings discussed in this study, promote the development of intelligent assisted design, education, healthcare, search and other aspects, and bring convenience and innovation to people's life and work. With the continuous progress of technology and the continuous optimization of the algorithm, it is believed that the algorithm will play an increasingly important role in the future.

5 Conclusion

All in all, this paper mainly explores and uses GNN algorithm to recognize various hand-drawn sketches in QuickDraw data set, and analyzes and discusses the recognition accuracy through a lot of experiments. In addition, this work also references the CNN algorithm, RNN algorithm and human recognition data to compare with GNN algorithm, hoping to find the advantages and disadvantages in the comparison, which will be of reference significance for future research. Through the whole experiment process, this work has the following conclusions: 1. The accuracy

of GNN algorithm is slightly lower than that of ordinary human; 2. In the face of complex and numerous sketch recognition tasks, GNN algorithm can successfully identify more accurately in most cases, with high reliability and stability; 3. The accuracy of GNN algorithm is higher than that of RNN algorithm, but not as good as CNN algorithm; 4. The training time of GNN model and CNN model is similar, but obviously longer than that of RNN model.

Authors Contribution

All the authors contributed equally and their names were listed in alphabetical order.

References

1. Xu, P., Joshi, C. K., & Bresson, X.: Multigraph transformer for free-hand sketch recognition. IEEE Transactions on Neural Networks and Learning Systems, **33**(10), 5150-5161 (2021).
2. Bhunia, A. K., Chowdhury, P. N., Yang, Y., Hospedales, T. M., Xiang, T., & Song, Y. Z.: Vectorization and rasterization: Self-supervised learning for sketch and handwriting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5672-5681. IEEE, Nashville (2021).
3. Xu, P., Hospedales, T. M., Yin, Q., Song, Y. Z., Xiang, T., & Wang, L.: Deep learning for free-hand sketch: A survey. IEEE transactions on pattern analysis and machine intelligence, **45**(1), 285-312 (2022).
4. LeCun, Y., Bengio, Y., & Hinton, G.: Deep learning. nature, **521**(7553), 436-444 (2015).
5. Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., etc.: Graph neural networks: A review of methods and applications. AI open, **1**, 57-81 (2020).
6. Xu, P., Joshi, C. K., & Bresson, X.: Multigraph transformer for free-hand sketch recognition. IEEE Transactions on Neural Networks and Learning Systems, **33**(10), 5150-5161 (2021).
7. Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., etc.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. Journal of big Data, **8**, 1-74 (2021).
8. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510-4520, IEEE. Salt Lake City (2018).
9. Deep Learning Theory and Practice MobileNet Series v1/v2/v3. https://zhuanlan.zhihu.com/p/670600898. Last Accessed: 2024/05/20.
10. Rana, R.: Gated recurrent unit (GRU) for emotion classification from noisy speech. arXiv preprint arXiv:1612.07778 (2016).
11. The Quick, Draw! Dataset. URL: https://github.com/googlecreativelab/quickdraw-dataset. Last Accessed: 2024/05/20