Multimedia Tools and Applications
https://doi.org/10.1007/s11042-020-09837-y

SketchFormer: transformer-based approach for sketch recognition using vector images

Anil Singh Parihar[1] · Gaurav Jain[1] · Shivang Chopra[1] · Suransh Chopra[1]

Received: 26 February 2020 / Revised: 10 August 2020 / Accepted: 9 September 2020 /
Published online: 07 November 2020

Abstract
Sketches have been employed since the ancient era of cave paintings for simple illustrations to represent real-world entities and communication. The abstract nature and varied artistic styling make automatic recognition of these drawings more challenging than other areas of image classification. Moreover, the representation of sketches as a sequence of strokes instead of raster images introduces them at the correct abstract level. However, dealing with images as a sequence of small information makes it challenging. In this paper, we propose a Transformer-based network, dubbed as AttentiveNet, for sketch recognition. This architecture incorporates ordinal information to perform the classification task in real-time through vector images. We employ the proposed model to isolate the discriminating strokes of each doodle using the attention mechanism of Transformers and perform an in-depth qualitative analysis of the isolated strokes for classification of the sketch. Experimental evaluation validates that the proposed network performs favorably against state-of-the-art techniques.

Keywords Sketch recognition · Transformers · Vector images · Deep learning

1 Introduction

The ability to draw and comprehend eclectic notions through sketches reflects the intellectual capabilities of human beings. Drawing has long been a part of human behavior,

✉ Anil Singh Parihar
parihar.anil@gmail.com

Gaurav Jain
gauravjain13298@gmail.com

Shivang Chopra
shivangchopra11@gmail.com

Suransh Chopra
suransh2008@gmail.com

[1] Machine Learning Research Laboratory, Department of Computer Science and Engineering, Delhi Technological University, New Delhi, 110042, India

mutating its utility from rock carvings in the ancient era, to blueprints of drafts in the modern age. Additionally, these sketch drawings assist humans in communication and creative design, such as art. Hence, intelligent machines must efficiently fathom the rapid proliferation of interaction with these human activities to be pervasive in all environments. Sketch analysis has been efficiently utilised in diverse domains like mathematics [18], chemistry [24], and electronics [23]. Further, sketch analysis spans a wide spectrum of applications, such as sketch segmentation [16, 19, 20, 35], image retrieval [12, 41, 43], and sketch recognition [11, 46, 48].

In this work, we focus on *sketch recognition*, which aims to identify human drawings and classify them into their respective categories. Although existing methods provide satisfactory accuracy performance for *image recognition* [17], only a few approaches have addressed the issue of identifying human drawn illustrations [27, 28, 45]. Sketch recognition was first introduced by Sutherland et al. [36]. Since then, several approaches have been introduced for better sketch comprehension in this active area of research. Conventional studies [28, 45] consider sketches as nothing but images, converting them to binary representations and extracting features for training classifiers like Support Vector Machines (SVM). With the success of deep learning-based approaches in recent years for image classification tasks [17], these techniques have gained importance in sketch analysis. In particular, convolutional neural network-based approaches [25, 46] have outperformed the conventional techniques in sketch recognition with significantly dominant accuracy performances. However, most of the techniques often overlook the characteristic properties of sketches, and hence leave room for improvement.

Perceiving sketch is a challenging task due to two main reasons, (1) heterogeneous representations and (2) level of abstraction. Representation of the same object is dependent upon the interpretation of that object, which consequently leads to high intraclass variation between samples of the same class. Figure 1 illustrates this diversity among different samples of the same class. For example, distinct views of a cat in Fig. 1b include different representations which render the task of finding patterns relatively difficult for the

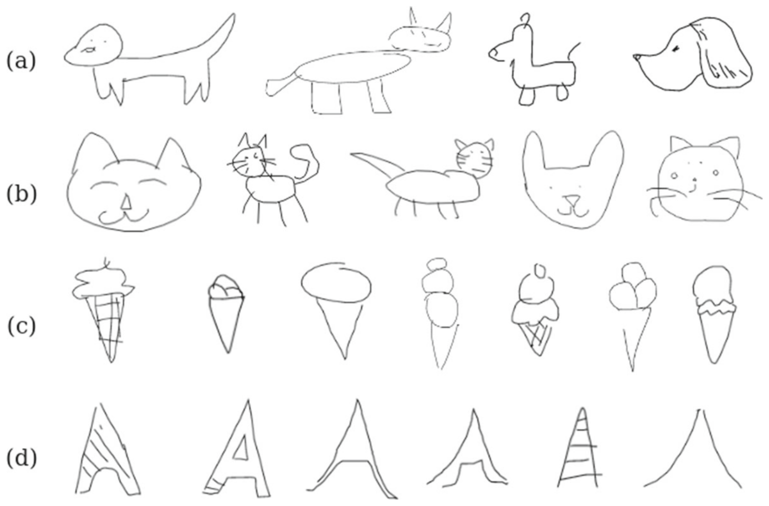

Fig. 1 Visualization of few classes in the QuickDraw [14] dataset, **a** Dog, **b** Cat, **c** Ice-cream, **d** Eiffel Tower showing the inter-class and intra-class homophilic behaviour and the highly abstract nature of sketches which poses significant challenges to the models

models. Alternatively, sketches of one class could be interpreted as belonging to another class. For instance, the second sketch of the dog class in Fig. 1a resembles a cat shown in Fig. 1b. Hence, learning such overlapping representations which may exhibit a significantly anomalous, yet acceptable, view proves to be an arduous task. Further, pixel-level images of sketches misinterpret how humans actually comprehend these drawings on an abstract level. Sketches are rendered as a series of strokes that build up the desired illustration over time. Hence, a more plausible expression of these sketches is the *vector image format*, which represents sketch drawings as a sequence of pen strokes. Besides, sketches generated using these vector image formats in Sketch-RNN [14] have corroborated their aptness for sketches. Therefore, an approach that deals with these complex correlations among sketches, interpreting sketches as a sequential abstraction is required. Moreover, applications like Handwriting Recognition, Optical Character Recognition (OCR) can hugely benefit from the advent of a robust classifier that can handle data with such huge variations.

In order to address the aforementioned issues, a transformer-based approach is proposed for attentive sketch recognition. The main contributions of the proposed approach are as follows:

– **Sketch Recognition using Transformers:** This is the first approach to the best of our knowledge that employs transformers for sketch recognition. We leverage the attention mechanism of Transformers to identify objects. For this, sketches are interpreted as a sequence of strokes, like humans actually comprehend drawings in real-time.
– **Attention-based analysis of sketches:** Through extensive qualitative analysis of the sketches using the attention mechanism, we isolate parts of the sketches necessary for object classification.

The rest of the paper is organized as follows: In Section 2, a literature review has been conducted to discuss related approaches. Section 3 discusses the preliminary concepts used in the proposed architecture. The proposed approach is presented in Section 4, which is followed by an exhaustive qualitative and quantitative experimental evaluation of the proposed technique is in Sections 5 and 6. Finally, Section 7 concludes the paper along with future directions.

2 Related work

In this section, existing frameworks for sketch classification have been discussed in depth. These approaches can be broadly classified as, (1) Conventional Classifiers, (2) Convolutional Neural Network (CNN) based approaches, and (3) Recurrent Neural Network (RNN) based approaches.

2.1 Conventional classifiers

Most of the traditional methods employ raster images as input. These pre-processed pixel-level images are used to extract local features from these sketches. These features are then utilized to classify sketches, using either classifier or guiding different learning strategies. For instance, Eitz et al. [11] proposed to represent sketches as bag-of-features, which trains a multi-class Support Vector Machine (SVM) to classify sketches. On the other hand, Schneider et al. [28] chose a Fisher Vector representation of raster images. In this, a Gaussian Mixture Model (GMM) was first generated and then used to encode the image using its deviation from the GMM, in terms of the Fisher Vector representation. Finally, an SVM is trained to classify images. In order to overcome the high structural complexity and absence of visual cues, Li et al. [21] proposed a star graph representation and employed a multi-kernel framework that fuses features extracted using different approaches.

Further, Yanki et al. [45] focused on reducing the amount of annotated data required, while maintaining recognition accuracy. For this, active learning was introduced, which measures the informativeness of unlabeled samples and selects batches of informative samples. Alternative approaches have been introduced by exploiting domain-specific knowledge [5, 7, 8, 32, 48, 51]. Although these approaches made progress in sketch recognition, most of these methods failed to address the complex and subtle style variations. Additionally, freehand sketches are casually drawn and hence pose the issue of inaccurate inputs, which further aggravates the complexity involved in sketch recognition.

2.2 Convolutional neural network-based

Recently, deep learning-based models have been found to perform favorably against traditional methods for classification and detection in diverse fields [22, 34, 38, 39] and even for sketch recognition [25, 42, 47, 50]. Several approaches on optimization algorithms have also been proposed [1–4, 6]. Among these, CNN based methods that use the same raster images as input have considerably improved recognition accuracy. In [26], authors propose a residual network (ResNet) based CNN architecture for freehand sketch recognition, validating that deeper networks perform favorably in comparison to shallow networks. Although the performance of different architectures were compared, this work lacks an exhaustive hyper-parameter tuning and discusses networks only as deep as 25 layers. Similarly, Yang et al. [44] proposed a deep CNN model, in order to overcome the need for feature engineering, usually required in traditional approaches. The authors further extended recognition capabilities to design an image retrieval method.

In [29], the authors proposed to extract features using deep CNN, and employed K-nearest neighbors for similarity search, in order to retrieve images. Additionally, the authors proposed to extract medium and high-level features that are extracted from deeper and shallow layers of the network, respectively. Seddati et al. [30] proposed DeepSketch 3, which utilizes the capabilities of deep convolutional neural networks for diverse applications like, partial sketch recognition, and sketch retrieval using query-by-example. In [31], authors propose to explore transfer learning for sketch recognition. Recently, Zhang et al. [49] proposed to exploit both the appearance and shape representations in a sketch. For this, a hybrid CNN was introduced, which extracted features using two different CNNs, one for shape, and the other for appearance. These features were then used to train an SVM and classify sketches. Several efforts have been focused on exploiting the powerful CNN architectures like Alexnet [17], and Google net [37].

Although a CNN-based approach [46] was the first to surpass human performance on the TU-Berlin dataset [11], these approaches employ a flawed representation, which proves to be the bottleneck in achieving even superior accuracy. Using current state-of-the-art deep CNN -based recognition models for this task proves unavailing due to the fact that the sketches are far less visually complex than photographs. The absence of color channels and the highly abstract nature of the sketches can be observed in Fig. 1d, where it is very hard for most of the sketches to qualify as the respective class. Furthermore, while people can agree on what an object looks like, how they ultimately render the object can vary significantly, thereby leading to lots of artistic variation. This lack of homophily in the sketches and their abstract nature renders most of the existing works done for sketch recognition at a serious disadvantage.

2.3 Recurrent neural network-based

In order to address the issue of faulty representation, Sketch-RNN [14] is the first approach to represent sketches in the vector image format. In this, recurrent neural networks (RNNs) process these vectors that constitute a sequence of strokes. The RNN encoder and RNN decoder, collectively compute the temporal correlation between strokes in order to classify sketches in their respective category. Vector representations have been proved to perform favorably for tasks such as sketch generation [13, 33] as well. Although RNN based approaches have been able to overcome the representation barriers, these approaches fail to model the underlying distribution when the number of object categories increase. Most of these methods use Long Short Term Memory (LSTM) units to encode the temporal information. While LSTMs have been exhaustively employed in both natural language processing applications and time series analysis, they have been observed to provide lower performance for large and complex sequences.

In order to model such sequences, *attention mechanism* [9] has been brought into action. This allows models to direct their resources to part of the information that most effectively represents that object. Consequently, facilitating models to achieve superior accuracy with lower data requirements. Our approach aligns with this ideology, and hence adapts the state-of-the-art RNN model, Tranformers [40] to these requirements. RNNs and Long Short Term Memory (LSTM) cells fail to encode long term dependencies and adapt to variable-length inputs. To address these issues, we propose to exploit the self-attention mechanism in Transformers, along with its capability to encode larger information, to identify multiple classes with one model. The following section discusses the proposed approach in detail.

3 Preliminaries

LSTMs perform recurrent operations in a sequential manner. On the contrary, Transformers [40] replace these complex recurrent computations with the attention mechanism. This allows parallel computation, completely relying on self-attention, thereby reducing computation complexity.

The transformer forms the building block in our model, and we introduce some background on the attention mechanism it uses. Attention can be described as a mapping between the query vector (Q) and key-value pairs (K, V) to obtain an output vector. The output vector is the weighted sum of values, $V = \{v_i | i \in \{1, 2, ..., d\}\}$, where Attention ($\mathcal{A}$) is computed using the *scalar-dot product* between a query and the keys, followed by its softmax, to obtain the weights of values:

$$\mathcal{A}(Q, K, V) = \text{SOFTMAX}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{1}$$

where $K \in \mathbb{R}^d$. With a single attention, averaging disarms the model to combine information from different representations. In order to facilitate this, *multi-head self-attention* is performed (Fig. 2). In this, queries, keys, and values are linearly projected to obtain varying set of projections to incorporate representations at different positions. Attention is computed in parallel for m attention heads. These are then concatenated to generate the final output. Equation (2) computes the output:

$$\mathcal{MA}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_m)\mathbf{W}^O \tag{2}$$

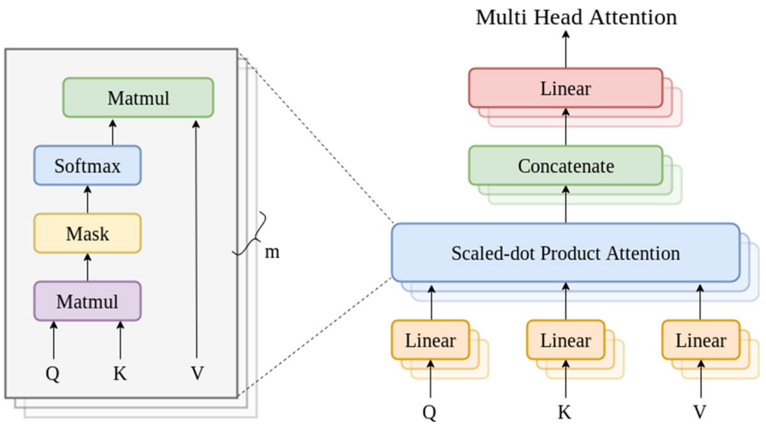

Fig. 2 Internal structure of the Multi-Headed Self-Attention Mechanism in a Transformer block explaining the various operations involved in obtaining the multi-head attention maps used for the ablation and attention-based analysis done in Section 5

where $\text{head}_j = \mathcal{A}(Q\mathbf{W}_j^Q, K\mathbf{W}_j^K, V\mathbf{W}_j^V)$, such that the project matrices $\mathbf{W}_j^Q, \mathbf{W}_j^K, \mathbf{W}_j^V \in \mathbb{R}^{\frac{d}{m} \times d}, \forall j \in \{1, 2, ..., m\}$, and $\mathbf{W}^O \in \mathbb{R}^{d \times d}$. The proposed network uses multi-head attention with $m = 8$ attention heads, along with $d = 64$. The parallel computation of these heads is analogous to performing a single attention. Hence, allowing the model to capture positional and temporal information without increasing time complexity. The following section discusses the role of these multi-head attention in the proposed approach in detail.

4 Proposed methodology

In this section, we establish the motivation and then describe the technical details of the proposed approach. Figure 3 shows the proposed architecture. Firstly, the data is pre-processed to transform it into a scale and rotation invariant format. Next, the Auto-Encoder module extracts a latent vector from the input to increase the input dimension, followed by processing of this latent vector by the Transformer module. Finally, the class probabilities are used to obtain sketch category.

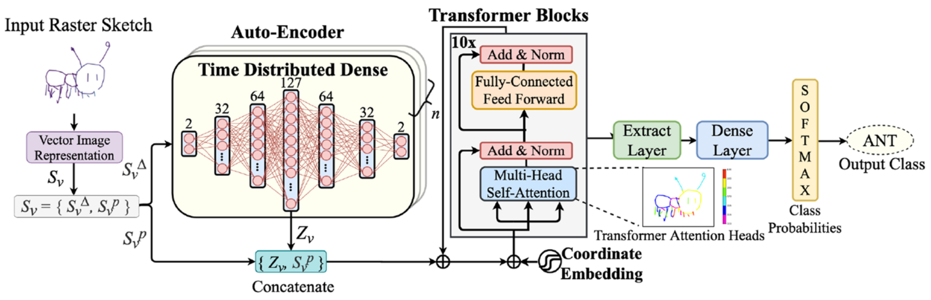

Fig. 3 Overview of the proposed Network depicting the pre-processing of raster sketch to form the vector image representation. Projecting the input to a higher latent dimension using the Auto-Encoder and the final classification using the attention maps from the Transformer blocks connected via skip connections

4.1 Motivation

The proposed approach is based on the premise of choosing a vector-image representation to facilitate real-time detection. This format enables the data to be proposed across the *time* dimension, instead of the *space* dimension as in case of a CNN-based approach. The major advantage of the proposed approach is to be able to detect sketches in real-time, i.e., even partially rendered sketches can be processed. Hence, we employ a Transformer-based architecture to process the sequential data.

4.2 Input representation

The input data is preprocessed to transform the sketches into the vector-image format. Our representation is an extension of the format used by Graves [13]. Sketches are represented using a multi-state pen stroke event. In this, $\mathcal{S}$ is a sketch with sequence of strokes s_i, i.e. $\mathcal{S} = \{s_1, s_2, ..., s_n\}$, where n is the sequence length. Each stroke, s_i, is defined using a 3-point format:

$$s_i = \{\Delta x_i, \Delta y_i, p_i\}, \quad \forall i \in \{1, 2, ..., n\} \tag{3}$$

where $(\Delta x_i, \Delta y_i)$ is the offset distance in the x and y direction. For each sketch, $(\Delta x_1, \Delta y_1)$ begin with origin as the initial starting point. Pen-state, p_i is a binary variable, with $p_i = 1$ indicating that the pen is in contact with surface and a stroke is drawn over the offsets $(\Delta x_i, \Delta y_i)$, while $p_i = 0$ represents that the pen is lifted off the surface and moved from the previous point in the direction of offsets. The following subsection discusses the processing of these inputs, $\mathcal{S}$, which is fed into the proposed network to obtain respective object categories.

4.3 AttentiveNet

Figure 3 illustrates the proposed architecture, which consists of two modules, (1) *auto-encoder*, and (2) *transformer-encoder*. The auto-encoder module extracts features from the input, while simultaneously facilitating a larger set of features to be fed into the transformer module. Further, the transformer module processes this input to attentively capture the characteristic information from the strokes at each time step. We now discuss these modules in detail.

4.3.1 Auto-encoder

The input representation in the form of 3-point format (3) works for RNNs [14]. However, the number of input features to the transformer must be divisible by the number of attention heads in each transformer block. Hence, a mechanism was required to project the input features to a higher latent dimension. Initial attempts of using a *convolution layer* and a *dense layer* were successful only to a limited extent and failed to capture the exact representation of the input features. To mitigate this issue, an auto-encoder module is proposed. The auto-encoder performs two major functions, (1) extract latent vector ($\mathcal{Z}_v$) from the input and (2) reshape the input dimension to feed a larger context vector to the transformer module for effective processing of the input data.

The input $\mathcal{S}_v$ is not directly fed into the auto-encoder. Instead, it is decomposed as $\mathcal{S}_v = \{\mathcal{S}_v^{\Delta}, \mathcal{S}_v^{p}\}$, where $\mathcal{S}_v^{\Delta} = (\Delta x_i, \Delta y_i)$ and $\mathcal{S}_v^{p} = p_i$. We only feed the offsets, $\mathcal{S}_v^{\Delta} \in \mathbb{R}^{2\times n}$, to obtain latent vector, $\mathcal{Z}_v \in \mathbb{R}^{127\times n}$, with a sequence length of n. Since $\mathcal{S}_v^{p}$ is a binary variable, it is not processed by the auto-encoder. This ensures that the pen-state information is not processed with the offsets, because these two sets of information are independent of each other. Hence, we concatenate the $\mathcal{S}_v^p$ with the latent vector obtained, as discussed below.

Figure 3 shows the auto-encoder module, which consists of *time-distributed dense layers* with dimensions (2, 32, 64, 127, 64, 32, 2). The middle layer, dubbed as the bottleneck layer is extracted as the latent vector $\mathcal{Z}_v$. Each layer uses ReLU activation. Further, latent vector is concatenated with the pen-state information to form the input to the tranformer module, $\mathcal{S}_v^T \in \mathbb{R}^{128\times n}$, defined as:

$$\mathcal{S}_v^T = \{\mathcal{Z}_v, \mathcal{S}_v^p\} \tag{4}$$

The encoder is trained independently to learn the input and output mapping between the offsets, $\mathcal{S}_v^{\Delta}$, and its extended representation in the form latent vector $\mathcal{Z}_v$. The processing of extended input, $\mathcal{S}_v^T$, by the transformer follows.

4.3.2 Transformer

Transformers [40] have been employed in the proposed approach to utilize the parallel processing capabilities while enabling real-time sketch recognition. In the past, Transformers have been primarily used for language modeling. We proposed significant changes to the original transformer architecture to adapt it to the sketch classification task. First, the embedding layer is removed since semantic information is not required to be preserved by our model as opposed to the domain of Natural Language Processing.

Further, to preserve the temporal nature of our data, we use *coordinate embedding* [10]. In this embedding, a tuple with (position, time) is computed, instead of calculating just the position. $\mathcal{P}^t \in \mathbb{R}^{x\times d}$ is obtained by generating a sinusoidal position embedding with $1 \le i \le h$ positions, and $1 \le t \le T$ time steps for each vector-dimension $1 \le j \le d$:

$$\mathcal{P}^t_{i,2j} = sin\left(\frac{i}{10000^{2j/d}}\right) + sin\left(\frac{t}{10000^{2j/d}}\right) \tag{5}$$

$$\mathcal{P}^t_{i,2j+1} = cos\left(\frac{i}{10000^{2j/d}}\right) + cos\left(\frac{t}{10000^{2j/d}}\right) \tag{6}$$

In the proposed network, we use 10 identical transformer blocks connected via skip-connections (for better information retention), as shown in Fig. 3. Each block consists of multi-head self-attention with $m = 8$ heads, which is followed by addition of the residual and normalization. The attention mechanism allows the model to learn characteristic information from each sketch category on an abstract level. This ability mitigates the inherent problem of high intra-class variation and low inter-class variations present in the human-drawn sketches. In addition, identical feed-forward fully-connected layer is applied at each position separately, which use varying set of parameters at each layer. The feed-forward ($\mathcal{F}$) layer is defined as:

$$\mathcal{F}(x) = ReLU(xW_1 + b_1)W_2 + b_2 \tag{7}$$

The encoded input $\mathcal{P}$ is fed into the Transformer blocks to obtain the point-wise attention-maps $\mathcal{A}_k \forall k \in \{1, 2, ..., m\}$. The custom *Extract layer* is used to pick the features (ext_{out}) corresponding to the first timestamp of the transformer output. Further, ext_{out} is passed through a dense layer with a softmax activation function to obtain class probabilities $\Psi = \{\psi_1, \psi_2, ..., \psi_c\}$, where c represents the number of classes. Finally, dense layers with softmax activation facilitate the model to return class probabilities. The following subsection specifies the model training configurations.

4.4 Implementation details

For training the proposed architecture, the two modules are trained separately. Firstly, we train the auto-encoder with a batch size of 1024, for 50 epochs and an initial learning rate of 0.0001. Next, we freeze the weights of the auto-encoder and train the complete network to learn the transformer parameters with a batch size of 32. Both modules share the same training configurations. Early stopping configured with $\delta_e = 0.00001$ and patience= 10 is used. Also, Learning Rate Scheduler defines the learning rate (lr_i) for i^{th} epoch as:

$$lr_i = ((lr_{beg} - lr_{min}) * \delta_{lr}^{i}) + lr_{min} \tag{8}$$

where $\delta_{lr} = 0.9$ is the decay rate, $lr_{beg} = 0.0001$ represents the initial learning rate, and $lr_{min} = 0.00001$ is the minimum learning rate. Finally we use Adam Optimizer to reduce the cross-entropy loss defined as:

$$\mathcal{L} = \sum_{j=1}^{c} \psi_j \times log(\psi_j) \tag{9}$$

All computations were carried out on a Linux workstation with Intel Xeon Silver 4110 CPU, having 128 GB RAM and a TITAN RTX GPU with 24 GB memory. A stratified 5-fold grid search was carried out on the hyper-parameters. The following section analyses the proposed method, both quantitatively and qualitatively.

5 Experiments

5.1 Dataset and evaluation metrics

The performance of the proposed Transformer-based network is evaluated using the benchmark Quick Draw dataset [14], which consists of over 50 million sketches scattered across 345 categories. This dataset was generated by collecting real-time sketches by players of the online game, *The Quick, Draw!*. While acquisition, the players were given 20 seconds to draw a sketch. Hence, generating abstract level sketches.

To validate the performance of the proposed and state-of-the-art methods, recognition accuracy is reported on the test-set. A train-test split of 80-20% is performed with 50,000 samples per class for training, and 10,000 samples per class during testing. For exhaustive comparison, we formulate three subsets of the Quick Draw dataset, having 5, 20, and 50 classes, respectively. For each subset, the respective number of classes are picked up randomly, with 10-fold cross-validation. Further, we compute the top-k accuracy for $k = \{1, 5, 10\}$, i.e., top-1, top-5, and top-10 recognition rates as well. Hence, we report the average accuracy for each subset. Trends in recognition accuracy on these datasets reveal the dependence of accuracy on the number of classes. The following subsection compares the accuracy performance with baselines. [1]

5.2 Comparative evaluation

We compared the proposed AttentiveNet with three types of approaches, (1) traditional classifiers, (2) CNN-based approaches, and (3) RNN-based approaches. Table 1 reports

[1] Official TensorFlow implementation of RNN for QuickDraw

Table 1 Comparative evaluation of recognition accuracy on the Quick Draw dataset, with (a) 5 classes, (b) 20 classes, (c) 50 classes

Method	5	20	50
HOG-SVM [11]	75.21%	66.79%	63.22%
Fisher-Vectors [28]	79.53%	75.80%	72.90%
AlexNet [17]	77.18%	75.22%	73.06%
Sketch-a-Net v2 [46]	94.78%	88.64%	85.19%
Resnet50-CNN [15]	**96.47%**	90.06%	86.20%
RNN-Tensorflow[1]	95.58%	87.33%	83.98%
Ours	96.21%	**90.31%**	**88.72%**

Values in **bold** depict the best accuracy for each subset

the recognition accuracy. We implemented and trained various approaches to compare the performance of the proposed approach. To ensure fairness, training is performed on the same subsets for all techniques. AttentiveNet outperforms the state-of-the-art approaches for 20 and 50 classes. However, Resnet50-CNN [46] performs marginally better than AttentiveNet with a gain of 0.26% for dataset with 5 classes.

As the number of classes increases from 5 to 20, most of the approaches show a significant decrease inaccuracy. The drop in accuracy is observed to be 11.20% for HOG-SVM [11], 4.69% for Fisher-Vectors [46], 2.53% for AlexNet [17], 6.48% for Sketch-a-Net v2 [46], 6.22% for Fisher-Vectors [46], 8.63% for RNN-Tensorflow[1], 6.13% for AttentiveNet. Further increasing the number of classes from 20 to 50, drop-in accuracy was observed to be 5.34% for HOG-SVM [11], 3.8% for Fisher-Vectors [46], 2.88% for AlexNet [17], 3.45% for Sketch-a-Net v2 [46], 4.22% for Fisher-Vectors [46], 3.83% for RNN-Tensorflow[1], 1.17% for AttentiveNet. For the proposed approach, increasing the number of classes observes a lower drop in accuracy, compared to other approaches. Due to limited resource availability, analysis over the complete dataset could not be performed. To account for this, we randomly selected c classes, where $c \in \{5, 20, 50\}$, and report the averages over 10-fold cross-validation.

In addition, we compare the proposed approach with the state of the art approaches over all 345 categories in the dataset. For this, we have compared the top-1, top-5, and top-10

Table 2 Comparative evaluation of recognition accuracy on the *Quick, Draw!* dataset, with (a) Top-1, (b) Top-5, (c) Top-10 accuracy

	Method	Top-1	Top-5	Top-10
Traditional	HOG-SVM [11]	50.25 ± 2.05%	62.51 ± 0.64%	64.22 ± 1.64%
Classifiers	Fisher-Vectors [28]	56.11 ± 1.34%	74.28 ± 1.00%	79.19 ± 0.33%
CNN-based	AlexNet [17]	63.25 ± 0.95%	80.83 ± 2.07%	85.02 ± 1.36%
Classifiers	ResNet50-CNN [15]	78.86 ± 2.18%	91.10 ± 0.86%	94.14 ± 0.97%
	Sketch-a-Net v2 [46]	74.84 ± 1.01%	87.19 ± 1.13%	91.25 ± 2.00%
RNN-based	RNN-Tensorflow[1]	75.59 ± 2.04%	90.14 ± 1.27%	**94.20 ± 2.08%**
Classifiers	Ours	**79.35 ± 0.96%**	**91.30 ± 0.84%**	93.95 ± 1.38%

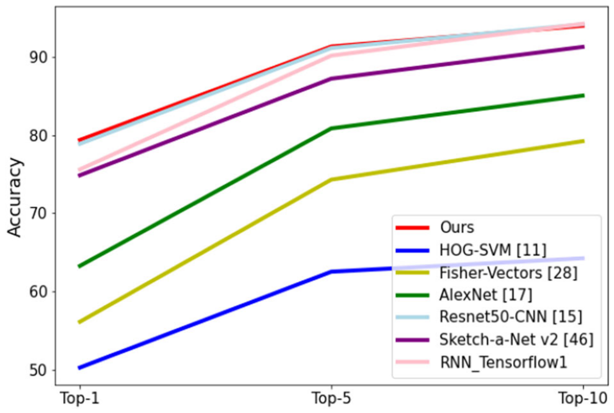

Fig. 4 Relative classification accuracy of the baselines and their trends with respect to Top-k accuracies

recognition accuracy. Table 2 reports the accuracy performance, and Fig. 4 visualizes these results. Quantitative evaluation confirms the effectiveness of representing sketches as vector images. Moreover, this representation facilitates real-time recognition of sketches. Ablation studies follow in the next subsection.

6 Discussions

6.1 Ablation analysis

6.1.1 Model configurations

The proposed approach majorly constitutes two modules, the auto-encoder, and the transformer blocks. In this section, we evaluate the effectiveness of these propositions. For this, we compare the recognition accuracy of different models by tweaking four model configurations. First, is the choice between Transformers [40] and Universal Transformers [10]. Since we adopt these sequential models given their success in language modeling, universal transformers provide a lucrative alternative against transformers. Second, we vary the number of transformer blocks. Third, we add skip-connections between each transformer block to exploit the power of residual learning. Finally, experiments that corroborate the effectiveness of the proposed auto-encoder is performed. Table 3 reports the observed performance.

For most of the language modeling tasks, Universal transformers perform favorably against the original transformer. However, the proposed architecture outperforms universal transformers. This is because universal transformer blocks share weights. Consequently, reducing the number of parameters. This leads to its inability to capture complex intraclass and interclass variations in sketches. Transformers with 5 blocks perform significantly better than its counterpart with universal transformers for all three subsets, with a gain of 0.1%, 3.07%, and 4.84%, respectively. Further, using the Adaptive computation time (ACT) mechanism with universal transformers still performs inferior to transformers, due to the early halting.

Varying the number of transformer blocks alters the number of parameters. The proposed network with 10 blocks performs significantly better than the 5-block variant. This enables the model to address the complexity of sketch recognition. Increasing the number

Table 3 Ablation Analysis and comparative results for different models on the QuickDraw [14] Dataset in terms of Accuracy reported on three subsets with 5, 20 and 50 classes respectively

Model Configurations				Recognition Accuracy		
T/UT	No. of Blocks	Skip-connections	Auto-encoder	5	20	50
T	5	✗	✗	93.66%	84.44%	78.90%
T	5	✓	✗	94.56%	86.38%	81.60%
T	10	✗	✗	92.91%	85.32%	75.03%
T	10	✓	✗	94.60%	86.55%	77.44%
UT	5	✗	✓	93.76%	87.11%	82.87%
UT + ACT	5	✓	✓	92.73%	83.14%	79.55%
T	10	✗	✓	95.71%	89.13%	87.21%
T	10	✓	✓	**96.21%**	**90.31%**	**88.72%**

T: Transformer UT: Universal Transformer ACT: Adaptive Computation Time. Values in **bold** depict the highest accuracy achieved in each subset

of blocks further provided an insignificant increase in accuracy with a relatively large training overhead, such as time and space complexity. Further, inclusion of residual learning observes improvement in model performances for CNNs [15]. In our network, we add skip-connections between the transformer blocks. This is because gradients often get lost when passed through multiple layers. In order to address this issue, skip-connections ensure that lower layers receive information, even if the network gets deeper.

The proposed architecture with auto-encoder displays significantly superior recognition accuracy, as compared to other ablations. This is due to the fact that the offsets $(\Delta x_i, \Delta y_i)$, contain fairly low amount of information for computationally intensive architectures like transformers. Hence, mapping the offsets to a larger latent vector enables the transformer to tweak the attention heads with a lot more available input data.

6.1.2 Attention heatmaps

The proposed method introduces transformers to exploit the sequence of strokes to draw attention to parts of sketches. Figure 5 illustrates the attention maps for various classes. The attention softmax scores corresponding to the first input of the first head is used for the attention-based ablation study. This particular choice is made taking into consideration the fact that the sketches are being recognised by extracting the logits from output corresponding to the first timestamp of the input. For the *bat* class, attention on both the wings are equally focused. This is analogous to how humans recognize bats through their peculiar wing shape. For the *cello* class, the model directs its attention towards the typical round shape. This trend is also observed for the *fan* class, where central spikes and pedestal are aptly recognized by the proposed model for recognition. It is interesting to note that for the *star* class, equally high attention is given to the complete structure. Perhaps, structures like stars that are symmetric in nature and constitute a distinct shape are identified based on the overall view of the sketches. Similarly, certain parts of the sketch are more important in recognizing an object; this is supported by the attention heatmaps presented.

The role of auto-encoders is discussed in the previous sub-subsection. In Fig 6, we compare the attention maps generated with (a) network containing the auto-encoder module, and (b) network without an auto-encoder module. The auto-encoder enhances the attention

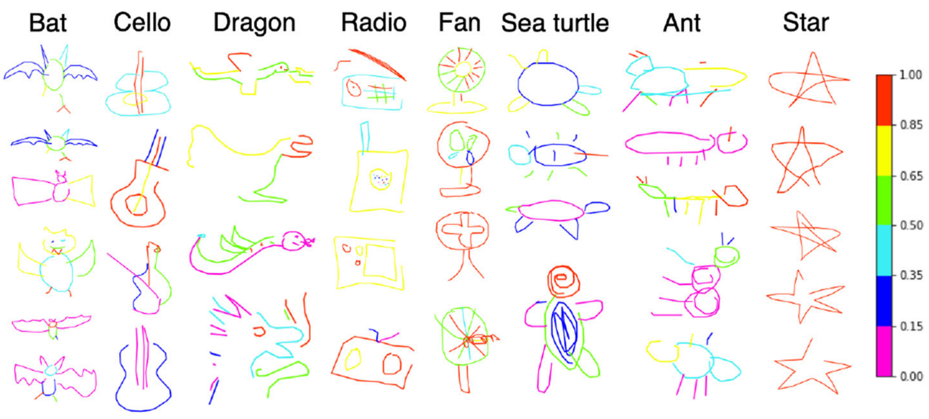

Fig. 5 Visualization of the attention maps of a few classes in the QuickDraw [14] dataset depicting the relative importance of the points in the strokes as observed by the model while inference

magnitude over the right areas for most of the classes. While the absence of auto-encoder provides fewer patches with high attention. This supports the hypothesis that the autoencoder module effectively extracts information and hence plays a pivotal role in the proposed AttentiveNet architecture. Figure 7 visualizes a sample User-Interface performance using the proposed approach, with inference in real-time.

6.2 Computational complexity analysis

In this subsection, we discuss the computational cost of the proposed approach in comparison to existing techniques. Conventional and CNN-based techniques can only perform classification, once the sketch has been rendered completely. On the other hand,

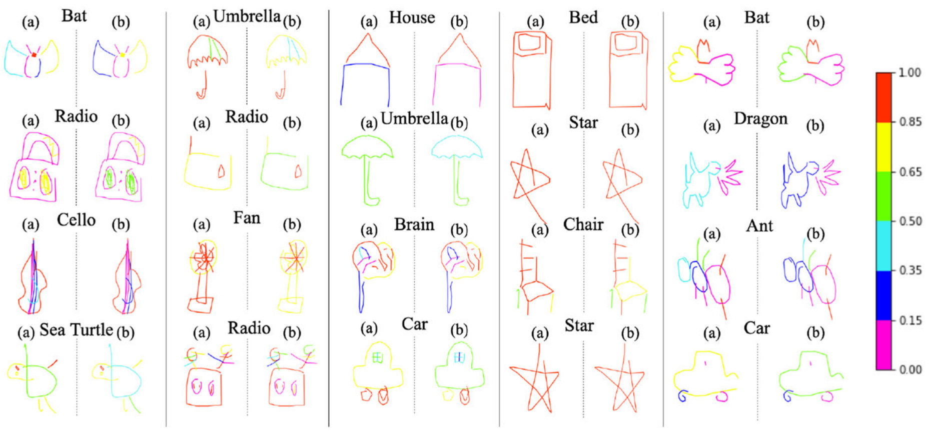

Fig. 6 Attention-map based ablation study using a Transformer (10 Blocks) + Auto Encoder b Transformer (10 Blocks)

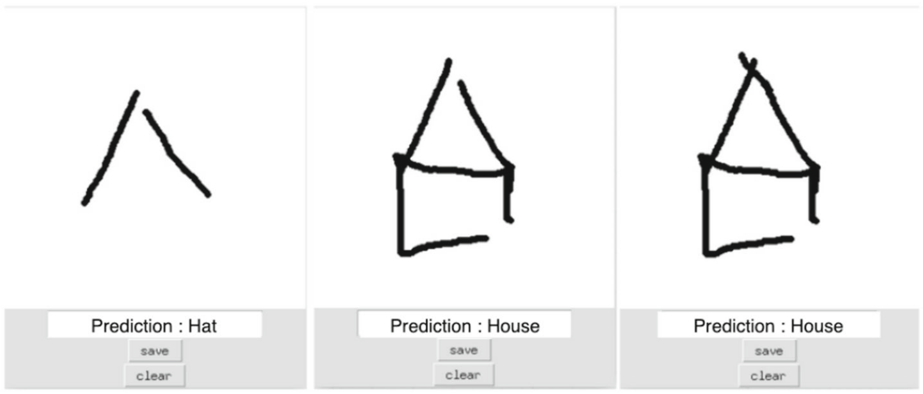

Fig. 7 Custom user interface designed to facilitate real-time inference. Progressive inference is performed on the sketch to accurately predict the object being drawn on the interface

sequence-based models have the ability to classify partially rendered drawings as well – facilitating real-time inference. Therefore, the use of sequence-based classifiers over the conventional and CNN-based sketch classifiers provides a significant advantage in terms of the inference times. This extended functionality of sequence-based classifiers establishes dominance over these conventional and CNN-based approaches in terms of computational efficiency.

Amongst the sequence-based classifiers, the Transformer-based architectures outperform the RNN-based techniques in terms of computational costs involved. This is mainly due to ability of Transformers to utilize parallel processing. To process an input with n time-steps, the Transformers require O(1) sequential operations, while RNN-based architectures require O(n) sequential operations [40]. We compare the average inference times per sample, averaged across 10,000 samples, for the difference sequence-based architectures, reported in Table 4. A noticeable improvement in inference times can clearly be observed for the proposed Transformer-based architecture as compared to existing RNN-based implementation i.e. RNN-Tensorflow[1].

7 Conclusion and future directions

In this work, we proposed a novel Transformer-based architecture, dubbed as AttentiveNet, for recognizing sketches using vector images. The autoencoder extracts information from the input data in the form of a latent vector. Moreover, this renders the input dimension compatibility between the input data and transformer-blocks. The proposed approach enables real-time inference capabilities, which is an essential functionality in any practical setup. Moreover, this functionality is incorporated at reduced computational cost, when

Table 4 Average per-sample inference times for the various architectures for the sequence-based models

Model	Inference Time
RNN-Tensorflow[1]	112.29 ± 10 ms
Transformer	80.38 ± 6 ms

compared with existing approaches. Further, transformer-blocks enable the model to focus on characteristic information from each sketch. The proposed approach achieves favorable recognition accuracy when compared with state-of-the-art approaches. Ablation analysis, along with attention heatmaps reveal that the proposed approach makes prediction based on certain parts of sketches, that are intuitive in nature.

In future, transformer-based architectures can be adapted to address problems in the domain of computer vision applications. Potential applications may involve tasks such as sketch retrieval or sketch synthesis, which require descriptive line-drawing features, could be exciting to explore. The proposed approach opens avenues to improve the cognition abilities of computers to process data in sequential nature. Alternate realms of computer vision that utilize such sequential information include human action recognition for complex motion sequences like, dance, martial arts etc.

References

1. Abualigah L (2020) Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. Neural Comput Applic pp 1–21
2. Abualigah L, Diabat A (2020) A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. Clust Comput pp 1–19
3. Abualigah LM, Khader AT (2017) Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering. J Supercomput 73(11):4773–4795
4. Abualigah LM, Khader AT, Hanandeh ES (2018) A combination of objective functions and hybrid krill herd algorithm for text document clustering analysis. Eng Appl Artif Intell 73:111–125
5. Abualigah LM, Khader AT, Hanandeh ES (2018) Hybrid clustering analysis using improved krill herd algorithm. Appl Intell 48(11):4047–4071. https://doi.org/10.1007/s10489-018-1190-6
6. Abualigah LM, Khader AT, Hanandeh ES (2018) A new feature selection method to improve the document clustering using particle swarm optimization algorithm. J Comput Sci 25:456–466
7. Abualigah LMQ (2018) Feature selection and enhanced krill herd algorithm for text document clustering, 1st edn, Springer Publishing Company, Incorporated
8. Arandjelović R, Sezgin TM (2011) Sketch recognition by fusion of temporal and image-based features. Pattern Recogn 44(6):1225–1234
9. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv:1409.0473
10. Dehghani M, Gouws S, Vinyals O, Uszkoreit J, Kaiser Ł (2018) Universal transformers. arXiv:1807.03819
11. Eitz M, Hays J, Alexa M (2012) How do humans sketch objects? ACM Trans Graph 31(4):44–1
12. Eitz M, Richter R, Boubekeur T, Hildebrand K, Alexa M (2012) Sketch-based shape retrieval. ACM Trans Graph (TOG) 31(4):31
13. Graves A (2013) Generating sequences with recurrent neural networks. arXiv:1308.0850
14. Ha D, Eck D (2017) A neural representation of sketch drawings. arXiv:1704.03477
15. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
16. Huang Z, Fu H, Lau RW (2014) Data-driven segmentation and labeling of freehand sketches. ACM Trans Graph (TOG) 33(6):175
17. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
18. LaViola JJ Jr, Zeleznik RC (2004) Mathpad 2: a system for the creation and exploration of mathematical sketches. ACM Trans Graph (TOG) 23(3):432–440
19. Li K, Pang K, Song J, Song YZ, Xiang T, Hospedales TM, Zhang H (2018) Universal sketch perceptual grouping. In: Proceedings of the european conference on computer vision (ECCV), pp 582–597
20. Li L, Fu H, Tai CL (2018) Fast sketch segmentation and labeling with deep learning. IEEE Comput Graph Appl 39(2):38–51
21. Li Y, Hospedales TM, Song YZ, Gong S (2015) Free-hand sketch recognition by multi-kernel feature learning. Comput Vis Image Underst 137:1–11

22. Liu P, Yu H, Cang S (2019) Adaptive neural network tracking control for underactuated systems with matched and mismatched disturbances. Nonlinear Dynam 98(2):1447–1464
23. Lu T, Tai CL, Su F, Cai S (2005) A new recognition model for electronic architectural drawings. Comput Aided Des 37(10):1053–1069
24. Ouyang TY, Davis R (2011) Chemink: a natural real-time recognition system for chemical drawings. In: Proceedings of the 16th international conference on Intelligent user interfaces, ACM, pp 267–276
25. Sangkloy P, Burnell N, Ham C, Hays J (2016) The sketchy database: learning to retrieve badly drawn bunnies. ACM Trans Graph (TOG) 35(4):119
26. Sarvadevabhatla RK, Babu RV (2015) Freehand sketch recognition using deep features. arXiv:1502.00254
27. Sarvadevabhatla RK, Surya S, Mittal T, Babu RV (2018) Game of sketches: deep recurrent models of pictionary-style word guessing. In: Thirty-second AAAI conference on artificial intelligence
28. Schneider RG, Tuytelaars T (2014) Sketch classification and classification-driven analysis using fisher vectors. ACM Trans Graph (TOG) 33(6):174
29. Seddati O, Dupont S, Mahmoudi S (2015) Deepsketch: deep convolutional neural networks for sketch recognition and similarity search. In: 2015 13th international workshop on content-based multimedia indexing (CBMI), IEEE, pp 1–6
30. Seddati O, Dupont S, Mahmoudi S (2017) Deepsketch 3. Multimed Tools Appl 76(21):22,333–22,359
31. Sert M, Boyacı E (2019) Sketch recognition using transfer learning. Multimed Tools Appl 78(12):17,095–17,112
32. Sezgin TM, Davis R (2008) Sketch recognition in interspersed drawings using time-based graphical models. Comput Graph 32(5):500–510
33. Song J, Pang K, Song YZ, Xiang T, Hospedales TM (2018) Learning to sketch with shortcut cycle consistency. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 801–810
34. Sun L, Zhao C, Yan Z, Liu P, Duckett T, Stolkin R (2018) A novel weakly-supervised approach for rgb-d-based nuclear waste object detection. IEEE Sensors J 19(9):3487–3500
35. Sun Z, Wang C, Zhang L, Zhang L (2012) Free hand-drawn sketch segmentation. In: European conference on computer vision. Springer, New York, pp 626–639
36. Sutherland IE (1964) Sketchpad a man-machine graphical communication system. Simulation 2(5) R–3
37. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9
38. Tang Z, Yu H, Lu C, Liu P, Jin X (2019) Single-trial classification of different movements on one arm based on erd/ers and corticomuscular coherence. IEEE Access 7:128,185–128,197
39. Tang ZC, Li C, Wu JF, Liu PC, Cheng SW (2019) Classification of eeg-based single-trial motor imagery tasks using a b-csp method for bci. Front Inform Technol Electron Eng 20(8):1087–1098
40. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008
41. Wang F, Kang L, Li Y (2015) Sketch-based 3d shape retrieval using convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1875–1883
42. Wang X, Chen X, Zha Z (2018) Sketchpointnet: a compact network for robust sketch recognition. In: 2018 25th IEEE international conference on image processing (ICIP), IEEE, pp 2994–2998
43. Xu P, Huang Y, Yuan T, Pang K, Song YZ, Xiang T, Hospedales TM, Ma Z, Guo J (2018) Sketchmate: deep hashing for million-scale human sketch retrieval. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8090–8098
44. Yang Y, Hospedales TM (2015) Deep neural networks for sketch recognition. arXiv:1501.07873 1(2), 3
45. Yanık E, Sezgin TM (2015) Active learning for sketch recognition. Comput Graph 52:93–105
46. Yu Q, Yang Y, Liu F, Song YZ, Xiang T, Hospedales TM (2017) Sketch-a-net: a deep neural network that beats humans. Int J Comput Vision 122(3):411–425
47. Zhang H, Liu S, Zhang C, Ren W, Wang R, Cao X (2016) Sketchnet: sketch classification with web images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1105–1113
48. Zhang J, Chen Y, Li L, Fu H, Tai CL (2018) Context-based sketch classification. In: Proceedings of the joint symposium on computational aesthetics and sketch-based interfaces and modeling and non-photorealistic animation and rendering, ACM, p 3
49. Zhang X, Huang Y, Zou Q, Pei Y, Zhang R, Wang S (2019) A hybrid convolutional neural network for sketch recognition. Pattern Recognition Letters

50. Zhao P, Liu Y, Lu Y, Xu B (2019) A sketch recognition method based on transfer deep learning with the fusion of multi-granular sketches. Multimed Tools Appl 78(24):35,179–35,193
51. Zou C, Yu Q, Du R, Mo H, Song YZ, Xiang T, Gao C, Chen B, Zhang H (2018) Sketchyscene: richly-annotated scene sketches. In: Proceedings of the european conference on computer vision (ECCV), pp 421–436