Sketch Input Method Editor: A Comprehensive Dataset and Methodology for Systematic Input Recognition

Guangming Zhu Xidian University Xi'an, China gmzhu@xidian.edu.cn

Kelong Wu Xidian University Xi'an, China 22031212500@stu.xidian.edu.cn Siyuan Wang Xidian University Xi'an, China siyuanwang@stu.xidian.edu.cn

Hao Li Xidian University Xi'an, China xdulihao@stu.xidian.edu.cn Qing Cheng National University of Defense Technology Changsha, China sgggps@163.com

Liang Zhang Xidian University Xi'an, China liangzhang@xidian.edu.cn

ACM International Conference on Multimedia (MM '23). ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXXXXXXXXXXX

1 INTRODUCTION

Free-hand sketching is a communication modality that transcends barriers and connects human societies [39]. Despite its abstract and concise nature, it can be illustrative, making it useful in various scenarios such as communication and design. It has been extensively studied in fields such as computer vision [9, 35, 43], computer graphics [26], human-computer interaction [8, 14] fields. Many related tasks have also been researched, including recognition [43], retrieval [2], generation [5], grouping [17], segmentation [38], and sketch-based image retrieval [13].

In the era after WIMP (Windows, Icons, Menus, Pointer), there is a growing interest in using multi-touch, hand gesture, speech, and gaze analysis to achieve more intelligent and natural humancomputer interaction [1].Free-hand sketching is another natural form of interaction that has been used since ancient times and is even learned by children before they start writing. Its universal nature allows it to transcend language barriers [39]. While the nature of free-hand sketching limits its usage in standardized WIMPbased Graphical User Interfaces (GUIs), the increasing prevalence of touchscreen devices has made it a highly promising input modality that allows users to quickly sketch their concepts as low fidelity prototypes[29].

Imagine drawing a flow chart using software like Visio or PowerPoint. You would need to select or drag the operational symbols from the toolbars to the drawing area before adjusting and editing them. However, this process can be time-consuming when the hierarchical toolbars contain a large number of operational symbols, especially for systems like the professional Command, Control, Communications, Computer, and Intelligence (C4I) system that contains over a thousand types of operational symbols. Our proposed solution is the Sketch Input Method Editor (SketchIME) system, which allows users to first sketch their desired symbols as low-fidelity prototypes. The system then automatically recognizes the symbols and their categories, replaces the sketches with corresponding standardized symbols at the correct positions, size, and orientation. This improves the overall efficiency of the interaction.

A SketchIME system should be designed in a way that enables it to recognize sketches and segment them into different semantic

ABSTRACT

With the recent surge in the use of touchscreen devices, freehand sketching has emerged as a promising modality for humancomputer interaction. While previous research has focused on tasks such as recognition, retrieval, and generation of familiar everyday objects, this study aims to create a Sketch Input Method Editor (SketchIME) specifically designed for a professional C4I system. Within this system, sketches are utilized as low-fidelity prototypes for recommending standardized symbols in the creation of comprehensive situation maps. This paper also presents a systematic dataset comprising 374 specialized sketch types, and proposes a simultaneous recognition and segmentation architecture with multilevel supervision between recognition and segmentation to improve performance and enhance interpretability. By incorporating few-shot domain adaptation and class-incremental learning, the network's ability to adapt to new users and extend to new task-specific classes is significantly enhanced. Results from experiments conducted on both the proposed dataset and the SPG dataset illustrate the superior performance of the proposed architecture. Our dataset and code are publicly available at https: //github.com/Anony517/SketchIME.

CCS CONCEPTS

• Human-centered computing \rightarrow Interaction techniques.

KEYWORDS

sketch, datasets, recognition, segmentation

ACM Reference Format:

Guangming Zhu, Siyuan Wang, Qing Cheng, Kelong Wu, Hao Li, and Liang Zhang. 2018. Sketch Input Method Editor: A Comprehensive Dataset and Methodology for Systematic Input Recognition. In *Proceedings of the 31st*

MM '23, October 29-November 3, 2023, Ottawa, Canada

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-8651-7/21/10...\$15.00

https://doi.org/XXXXXXXXXXXXXXX

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

MM '23, October 29-November 3, 2023, Ottawa, Canada



Figure 1: Examples of the sketched symbols in our dataset. These sketches have different sketching styles, large diversity on stroke length and count, and subtle differences between some categories.

components. This will allow standardized symbols to be selected as replacements and adjusted according to the control points obtained from the segmentation results. Additionally, given the nature of free-hand sketch and the diversity of sketching styles among users, a well-designed SketchIME should be able to adapt to various styles online. Moreover, the system's ability to extend to new task-specific classes can make it even more attractive to users.

Using the aforementioned motivation and analysis, we collected and annotated a sketch dataset based on the public standard of the APP-6(B) joint military symbolism for C4I systems ¹. This standard outlines common operational symbols and their display/plotting details. We recruited a total of 67 participants to sketch the selected 374 types of symbols, as shown in Fig. 1. This is the first large-scale and systematic dataset to feature sketch categories from a professional C4I system, as opposed to familiar everyday objects. Our proposed network is a simultaneous recognition and segmentation architecture, with the recognition stream providing supervision to the segmentation stream based on prior knowledge. The incorporation of multilevel supervision enhances the interpretability of the network. Additionally, the utilization of domain adaptation techniques enhances the network's ability to adapt to new personal sketching styles at the application level. In addition, we have also explored a few-shot class-incremental learning mechanism to improve the network's ability to extend to new task-specific classes.

Our contributions are summarized as follows:

- The SketchIME dataset is the first of its kind, consisting of a large and systematic collection of sketches from a professional C4I system, rather than from familiar everyday objects. It is also annotated for easy reference.
- Our proposed approach is a simultaneous recognition and segmentation network that improves performance by incorporating multilevel supervision between recognition and segmentation tasks, while also enhancing interpretability.
- The incorporation of few-shot domain adaptation and classincremental learning enhances the network's practicality by improving its adaptability to new users and extendibility to new task-specific classes

¹https://www.scribd.com/document/314367702/APP-6-B-Joint-Symbology-pdf

2 RELATED WORKS

2.1 Sketch Dataset

This paper focuses on the uni-modal free-hand sketch datasets. The widely used sketch datasets for recognition are TU-Berlin [7] and QuickDraw [9], which contain 250 and 345 object categories, respectively. Based on the QuickDraw dataset, some sketch datasets for grouping or segmentation are constructed, such as SPG [17] and SketchSeg-150K [25]. SketchSeg-10K [36] is another segmentation dataset, which consists of 10 categories and 24 semantic labels. These datasets contains a lot of familiar everyday objects.

In this study, we construct a large-scale and systematic dataset whose sketch categories come from a professional C4I system. In contrast to the large inter-category differences in the reviewed datasets, the difference between two categories in our dataset may be a single short line. Therefore, effective recognition and segmentation methods must learn the subtle differences.

2.2 Sketch Recognition and Segmentation

Sketch recognition aims to predict the class label of a given sketch. Sketch-a-Net [44] is the first Convolutional Neural Network (CNN) based sketch recognition model that beats the human performance. Since a sketch is plotted as a continuous sequence of stroke points, Recurrent Neural Network (RNN) based models are also proposed in the literature. Stroke vectors or CNN features of strokes are fed to these models for sketch recognition [10, 15, 23, 27, 40]. In contrast to the above models which exploit either the static nature of sketches with CNNs or the temporal sequential property with RNNs, sketches are represented as multiple sparsely connected graphs in [41], and the global and local geometric stroke structures as well as temporal information are simultaneously captured in the multi-graph transformer.

Compared to sketch recognition, segmentation is a more finegrained analysis task. CNN, RNN, and Graph Convolutional Network (GCN) based models have been proposed for stroke-level analysis. CNN-based network in [18] takes the binary image of a Table 1: Comparison between uni-modal sketch datasets. Our SketchIME dataset contains the largest number of classes for recognition and the largest number of semantic components for segmentation. "seg" denotes the segmentation annotation, and "SC-Count" denotes the type count of semantic components in each dataset.

Datasets	Amount	Category	Stroke	Annotations	SC-Count
TU-Berlin[7]	20K	250	\checkmark	class	-
QuickDraw[9]	50M+	345	\checkmark	class	-
QuickDraw-5-step[3]	38M+	345	-	class	-
SketchFix-160[28]	3904	160	\checkmark	class	-
COAD[33]	620	20	\checkmark	class	-
SPG[17]	20K	25	\checkmark	class, grouping	86
SketchSeg-150K[25]	150K	20	\checkmark	class, seg	57
SketchSeg-10K[36]	10K	10	\checkmark	class, seg	24
SketchIME(Our)	56K	374	\checkmark	class, seg	139

sketched object as input and produces a corresponding segmentation map with per-pixel labels. The SketchSegNet [38] and Sketch-SegNet+ [25] work in a RNN-based Variational Auto Encoder (VAE) pipeline. SPFusionNet [36] uses late fusion of CNN-RNN branches to represent sketches for segmentation. SketchGNN [42] uses a convolutional Graph Neural Network (GNN) for semantic segmentation and uses a mixed pooling block to fuse the intra-stroke and inter-stroke features.

2.3 Adaptability and Extendibility

Users may have their own personal sketching styles, similar to handwriting. A trained model may not work well for a new user who has a different sketching style. This means that the training data and the new users' data have different underlying distributions. Domain Adaptation (DA) is therefore needed to reduce the effect of domain shift [22] for SketchIME applications. There are different types of DA techniques [24]. For instance, DeepCORAL [31] is a DA method based on correlation alignment [30]. The Deep Domain Confusion (DDC) [34] uses an additional domain confusion loss to learn a domain invariant representation. The Conditional Domain Adversarial Network (CDAN) and its variant CDAN+E [19] use an adversarial DA technique, motivated by the conditional generative adversarial networks [20], and make use of the discriminative information obtained from a deep classifier network. Based on CDAN, DA-FSL [46] further poses an additional challenge of DA with few training samples. These methods can be explored to ensure the adaptation ability of SketchIME.

The ability to incrementally learn new sketch classes is crucial for SketchIME, particularly in cases where additional task-specific sketches were not included in the original training. Users are not obliged to capture large data for the evolution of trained models. Therefore, it is crucial to apply Few-Shot Class-Incremental Learning (FSCIL) [32] technique to SketchIME. FSCIL is widely studied for image classification. Zhang [45] proposed a Continually Evolved Classifier (CEC) to update the classifier only in each incremental session to avoid knowledge forgetting. Zhu [49] proposed an incremental prototype learning scheme that consists of a random episode selection strategy and a Self-Promoted Prototype Refinement (SPPR) mechanism. CEC [45] and SPPR [49] only consider extending the model in a single phase, while LIMIT [48] considers extension in

\bigcirc		Л	-	L J	ο	0	늬	I	ᠧ	٢	т	т	М	F
\square		•		S	KX	X	\bowtie	N		G	А	Н	Ζ	/
\square	В	4	V	D	F	\bigcirc	Γī	\heartsuit	\diamond	~~~	ථ		\square	്
Г	₽	Ĉ	\bigtriangleup	\mathcal{M}	\cap	X	-4-4-	Ľ	Ъ	\downarrow)(\wedge	\bowtie
W	\bigvee	÷	厶	L	Õ	÷	Ÿ	⊲	_	\vee	\sim	പ	Δ	\sim
/		<u>ر</u> ۲	Y	꺐	Ż	\heartsuit	▽	۵	\asymp	*	≁_	C	ŝ	<
\rangle		Q	N	Т	Ň	Ρ	멋	2	¢	ł	ه_ه	7	₽~ 0	_ _
U	R	Е	\frown	0	\sim	2	\land	$\overline{\}$	7-	8	X	w	v	R
-1	\smile	⊞	Ŷ	\mathcal{D}	\square		\Box	К	\checkmark	∇	Ŷ	\bigcirc	\bigcirc	⇔
\bigcirc	-		#											

Figure 2: Semantic components of sketches in our dataset. The last semantic component denotes blank filling which may be done arbitrarily and result in a large diversity on stroke length and count.

multiple phases. In contrast to the above fake-task based training methods, C-FSCIL [12] relies on the hyper-dimensional embedding space to incrementally create maximally separable classes. The ForwArd Compatible Training (FACT) [47] assigns virtual prototypes to squeeze embeddings of known classes and reserve for new ones.

3 DATASET CONSTRUCTION

3.1 Category Selection

Total 374 categories of symbols are selected from the public standard of the APP-6(B) joint military symbology for our sketch collection. Fig.1 displays some sketched symbols, and all the 374 standardized symbols and their sketches are shown in the supplementary materials. Our dataset is different from the existing ones in several ways: (a) The sketch categories come from a professional C4I system, (b) Only subtle differences exist between some sketch categories, such as a single short line, (c) Blanks in some sketches need to be filled, and this results in a large diversity on stroke length and count, (d) The subtle differences between sketch categories and the large diversity of different personal sketching styles bring challenges to universal recognition and segmentation networks.

3.2 Data Collection and Annotation

Total 67 participants were recruited to sketch the 374 categories of symbols using the Concepts Application ². Phones or Pads with the Android or iOS operating systems were used to sketch. A touch pen was either used or not by participants, according to their personal preference. This also results in different sketching styles. The Scalable Vector Graphics (SVG) format was used to store sketches.

Besides the class labels for each sketch sample, semantic labels are assigned to strokes of sketches of all the categories. The semantic labels are obtained according to the principle, "If the increase or decrease of a stroke (or multiple consecutive strokes) will change one sketch's category, or a stroke (or multiple consecutive strokes) constitutes some class-specific component, the stroke (or strokes) is viewed

²https://concepts.app/en/

as a semantic component." It means that different categories may share semantic components. We numbered the semantic components of all categories uniformly and obtained 139 semantic labels, as displayed in Fig. 2. Our work aims to recognize the categories and semantic components simultaneously. This is different from the grouping or segmentation tasks [17, 42].

3.3 Data Statistics

Total 209K samples of the 374 categories were drawn. The publicly released version of the SketchIME dataset contains 56K samples, sketched by 8 participants for training and 10 participants for testing. Different participants may draw the same symbol using different stroke orders. Each participant only drew no more than 187 categories, and at most 25 samples were drawn for each category. Table 1 illustrates the comparison of our SketchIME dataset and popular uni-modal sketch recognition and segmentation datasets. For different specific tasks, the dataset is divided into different training and testing sets, and the numbers of sketches in the training and testing sets are illustrated in Table 2. Our SketchIME dataset is constructed and annotated from scratch, and contains the largest number of classes for recognition (i.e., 374 categories) and segmentation (i.e., 374 categories and 139 semantic labels).

4 METHODS

The black-box properties of deep learning restrict its applications in professional fields. The interpretability of neural networks can make them reliable. In this study, we propose a simultaneous recognition and segmentation network, whose multilevel supervision between recognition and segmentation based on prior knowledge makes the inference more interpretable. In addition, the use of fewshot domain adaptation improves the network's adaptability to new users' sketch styles, and the use of few-shot class-incremental learning enhance its extendibility to new task-specific classes. These features enhance the network's practicality.

4.1 Recognition and Segmentation

Our study suggests that, CNNs are effective for sketch-level recognition on images, and GNNs are better for stroke-level segmentation on SVG format. CNNs can learn sketch-level features effectively, while GNNs can take full use of stroke-level and point-level relationships between stroke points for segmentation. Furthermore, the prior knowledge about semantic components which should be contained in each sketch category, can also be used. Therefore, a simultaneous Sketch Recognition and Segmentation network (SketchRecSeg) is proposed in this study, whose recognition stream supervises the segmentation stream based on the prior knowledge. This allows the network to not only inference the sketch category, but also explain how the category is predicted based on the semantic component recognition.

The two-stream architecture of our SketchRecSeg is illustrated in Fig.3. The recognition stream uses a CNN network for feature learning and sketches are fed as images to this network. The segmentation stream uses a dynamic graph convolution unit and takes stroke points of SVG-format sketches as input. The features of the CNN network are taken as sketch-level features to augment the node features of the GNN block for segmentation. The recognition



Figure 3: The proposed simultaneous sketch recognition and segmentation architecture. The top half denotes the twostream architecture for simultaneous recognition and segmentation. The bottom half denotes the domain adaptation architecture using conditional domain adversarial. RClassifier and SClassifier denotes classifiers of categories and semantic components. The SPooling, CFA and RSM denotes the Stroke-level Pooling, the CNN Feature Augmentation, and the Recognition Supervision Module, respectively. KLD denotes the Kullback-Leibler divergence loss. © denotes feature concatenation operation, and \odot denotes Hadamard product. f^s and f^t denote the domain-specific feature representation, g^s and g^t denote the classifier predictions, D denotes the domain discriminator, and \otimes denotes multilinear map operation.

stream provides multilevel supervision (i.e., feature-level supervision using CFA, prediction-level supervision using RSM, and losslevel supervision using KLD) on segmentation. Besides, a domain adaptation mechanism [19] is equipped to support SketchRecSeg's adaptability to new users.

Feature Extraction. We use ResNet18 [11] to learn sketch-level features, and use the dynamic graph convolution unit [42] to learn point-level features. ResNet18 takes three-channel images transformed from the original SVG sketches as input, and outputs 128-dimensional sketch-level features by stacking an extra linear layer. The dynamic graph convolution units take a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as input. \mathcal{V} represents the *N*-point set $\mathcal{P} = \{p_i = (p_i^x, p_i^y)\}_{i=1,2,...,N}$, where p_i^x and p_i^y are the 2D absolute coordinates of point *i* in strokes of a given sketch. \mathcal{E} represents the edges that connect adjacent points in each single stroke. The graph convolution operation in [37] is used, and \mathcal{E} is updated layer-by-layer using the Dilated k-NN [16]. The dynamic graph convolution unit contains four convolution blocks, each block outputs a 32-dimensional feature vector for each node, and a 128-dimensional feature vectors.

Stroke-level Pooling (SPooling). Each stroke contains several points, and all stroke points constitute the graph nodes. Although the dynamic graph convolution units have evolved the node features based on the updated neighborhood, using the node features alone still cannot achieve a satisfactory segmentation performance. The information about stroke and its relevant points can be obtained, and a stroke-level pooling strategy can be applied to augment the node features in stroke-level. Given the node features $\mathcal{F}_q = \{f_q^i\}_{i=1,2,...,N}$, SPooling can be implemented as:

$$f_s^j = \max_{i \in \mathcal{V}_j} \text{MLP}(f_g^i), \tag{1}$$

$$f_{sg}^{i} = \operatorname{concat}_{i \in \mathcal{V}_{j}}(f_{s}^{J}, f_{g}^{i}), \qquad (2)$$

where \mathcal{V}_j denotes the node subset belonging to stroke j, f_s^J and f_{sg}^i denote the stroke-level and the augmented features, respectively.

CNN Feature Augmentation (CFA). A simple global pooling on all graph nodes cannot provide effective sketch-level features. In contrast, the features of ResNet18 used for sketch recognition are the ideal representation of the whole sketch. Therefore, given the CNN feature f_c , the node features can be augmented further as:

$$f_{csa}^{i} = \operatorname{concat}_{i \in \mathcal{V}}(f_{c}, f_{sa}^{i}).$$
(3)

Recognition Supervision Module (RSM). This RSM module aims to use the sketch recognition probability to supervise the segmentation task. If a sketch is recognized belonging to some specific category, the semantic components making up the sketch category should also be predicted with high probability for segmentation task. Given the prior knowledge about semantic components, which sketch category contains, a translation matrix $C_{r2s} \in \mathbb{R}^{C_R \times C_S}$ is constructed where $c_{ij} = \gamma_r$, when the *i*-th sketch category contains the *j*-th semantic component and otherwise $c_{ij} = 1 - \gamma_r$. C_R and C_S are the counts of sketch categories and semantic components. Given the sketch prediction probability $P_r \in \mathbb{R}^{1 \times C_R}$ and the indices Idx_P of top-*k* of P_r , a probability vector that enhances the correct prediction of semantic components and suppresses the wrong prediction can be calculated as:

$$\Gamma_{sc}^{r} = MaxPooling(index_select(C_{r2s}, Idx_{P})).$$
(4)

Then, given the output $P_s \in \mathbb{R}^{N \times C_s}$ of the semantic component classifier in segmentation stream, the final semantic component prediction for point *i* can be calculated as:

$$P_s^i = P_s^i \circ \Gamma_{sc}^r, \tag{5}$$

where \circ denotes the Hadamard product. Specifically, Eq. (5) is only applied when the confidence level of the category recognition exceeds a threshold.

Kullback-Leibler Divergence (KLD) Loss. The total loss of SketchRecSeg can be represented as:

$$L = L_s + \lambda_1 L_r + \lambda_2 L_{kl},\tag{6}$$

where L_s is the point-level segmentation loss, L_r is the recognition loss, λ_1 is the coefficient to balance recognition and segmentation, and λ_2 is an indicator of whether L_{kl} is used or not. L_s and L_r use cross-entropy loss. L_{kl} is the KL-divergence loss that represents the recognition's supervision on segmentation, and can be denoted as:

$$L_{kl} = \mathrm{KL}[P_{sc}^r||P_{sc}^s],\tag{7}$$

where P_{sc}^r and P_{sc}^s denote the probability distributions of semantic components derived from the recognition and segmentation results, respectively. P_{sc}^r can be calculated as:

$$P_{sc}^r = P_r * \mathbf{C}_{r2s}.$$
 (8)

The stroke-level semantic component prediction can be obtained by pooling on the point-level prediction results, and it also represents how much each semantic component should exist in this sketch. Therefore, P_{sc}^s can be calculated as:

$$P_{sc}^{s} = \max_{j=1,...,J} \left(\frac{1}{|\mathcal{V}_{j}|} \sum_{i \in \mathcal{V}_{j}} P_{s}^{i} \right), \tag{9}$$

where J is the stroke count of one sketch.

Domain Adaptation (DA). Users may have their own personal and preferred sketching styles. A trained model may not work well for a new user who has a different sketching style. This is because of the domain shift between the training data (source domain) and the new user's data (target domain). If using few samples of new users to fine-tune the trained model can improve the performance significantly, it can enhance SketchIME's practicality. It is reasonable to assume that few samples (e.g., empirically selected five per category) in the large-scale training data can be used as the source domain data for fine-tuning our model. These samples can be considered as typical examples of each category for user reference. Similar to using common input method editors, when users are using SketchIME, they will select the right symbol from the categories recommended by the recognition results. This process provides some labeled data as the target domain data. Given a small training data and new user's data, few-shot domain adaptation can enhance SketchIME's practicality.

CDAN [19] can be used to enhance the network's adaptability to new users. Given the source classifier G (including the feature extractor and the classifier in each stream of SketchRecSeg) and the domain discriminator D (for each stream), the optimization problem of supervised domain adaptation for the recognition stream (or the segmentation stream) can be denoted as:

$$\min_{G} E(G) - \lambda E(D,G) \text{ and } \min_{D} E(D,G), \tag{10}$$

$$E(G) = \mathbb{E}_{(x_i^s, y_i^s) \sim D_s} L(G(x_i^s), y_i^s)$$
⁽¹¹⁾

$$+\mathbb{E}_{(x_j^t, y_j^t) \sim D_t} L(G(x_j^t), y_j^t),$$

$$E(D,G) = -\mathbb{E}_{x_i^s \sim D_s} \log[D(f_i^s, g_i^s)] -\mathbb{E}_{x_i^t \sim D_t} \log[1 - D(f_j^t, g_j^t)].$$
(12)

where f_i^s and f_j^t are the feature representation of the CNN (or GNN) stream, g_i^s and g_j^t are the classifier predictions, x_i^s and x_j^t are sketch images (or strokes), y_i^s and y_j^t are ground-truth category (or semantic component) labels, and D_s and D_t are the source (few training data) and target (new users' data) domains.

4.2 Extendibility Enhancement

In a professional C4I system, there are more than one thousand types of sketches. Each sketch consists of basic semantic components, and new task-specific sketch categories (even new semantic components) may be derived from combinations of multiple semantic components. It means that, in order to enhance the SketchRecSeg network's forward compatibility to ensure its extendibility, we can pre-assign several virtual prototypes in the embedding space, and treat them as 'virtual classes'. Therefore, the ForwArd Compatible Training (FACT) [47], which assigns virtual prototypes to squeeze the embedding of known classes and reserves for new ones, can be used for class-incremental learning.

Figure 3 shows that we need forward compatible training for recognition and segmentation streams simultaneously. The loss for each stream can be given as $L = L_v + L_f$:

$$L_{v}(\mathbf{x}, y) = l(f_{v}(\mathbf{x}), y) + \gamma l(\operatorname{Mask}(f_{v}(\mathbf{x}), y), \hat{y}),$$
(13)

$$L_f(\mathbf{z}, y) = l(f_v(\mathbf{z}), \hat{y}) + \gamma l(\operatorname{Mask}(f_v(\mathbf{z}), \hat{y}), \hat{\hat{y}}),$$
(14)

$$Mask(f_v(\mathbf{x}), y) = f_v(\mathbf{x}) \otimes (1 - OneHot(y)),$$
(15)

where **x** and *y* are the data and label of known classes, *l* is the cross-entropy loss, γ is the trade-off parameter, **1** is an all-ones vector, and \otimes is Hadamard product. The cosine classifier can be abbreviated as $f_v(\mathbf{x}) = [W, P_v]^{\top} \phi(\mathbf{x})$, where *W* and P_v are the prototypes of known and virtual classes, and $\phi(\cdot)$ is the feature extractor. $P_v = [p_1, \cdots, p_V] \in \mathbb{R}^{d \times V}$ denotes the prototypes of *V* virtual classes. $\hat{y} = \operatorname{argmax}_v p_v^{\top} \phi(\mathbf{x}) + |Y_0|$ is the virtual class with maximum logit, acting as the pseudo label, where $|Y_0|$ is the class count of known classes. **z** is virtual instance obtained by fusing the embeddings of two instances by manifold mixup. $\hat{y} = \operatorname{argmax}_k w_k^{\top} \mathbf{z}$ is the pseudo label among current known classes. The hyper-parameters have same values as in [47]. The forward compatible training is implemented simultaneously for incremental learning of new classes and semantic components.

5 EXPERIMENTAL STUDIES

5.1 Datasets

Three datasets are constructed from the collected data, as illustrated in Table 2. *SketchIME-SRS* contains data of 8 participants for training and data of another 10 participants for testing. On the one hand, SketchIME-SRS is used for common evaluations based on the train and test subset. On the other hand, one or two samples from the 10 participants in the test subset can also be used for domain adaptation training when SketchRecSeg is equipped with DA, and another five samples from each category sketched by 10 participants are used for testing.

SketchIME-CIL1 and SketchIME-CIL2 are constructed for the extendibility evaluation. The former is used for the situation where both new sketch categories and new semantic components are expected to emerge in incremental sessions. The latter is used for the situation where all semantic components already exist in the base session and only new sketch categories emerge in incremental sessions. Each sketch category only has five samples in incremental sessions. We cannot design all the incremental sessions with the same count of new sketch categories and new semantic components. Therefore, 16 new sketch categories emerge in each incremental session first, and then the incremental semantic components in this session are extracted from the 16 new sketch categories. This results in different increment counts of semantic components in each incremental session. The details about the sketch categories and semantic components in base and incremental sessions are visualized in the supplementary materials.

Table 2: Three datasets constructed from the collected data for three tasks. The numbers in square brackets denote the sketch category count, the semantic component category count, and the sample count in each subset, respectively. The numbers in the 'Incremental' column denote the total counts of new sketches and semantic components emerging in incremental sessions.

Datasets	Train/Base	Incremental	Test
SketchIME-SRS	[374,139,36693]	-	[374,139,19781]
SketchIME-CIL1	[214,98,20904]	[160,41,800]	[374,139,19781]
SketchIME-CIL2	[214,139,21077]	[160,0,800]	[374,139,18986]

Table 3: Comparison results between the proposed SketchRec-Seg and the state-of-the-art methods of sketch recognition and segmentation on SketchIME-SRS dataset.

Networks	P-Metric	C-Metric	Acc@1
Vision Transformer[6]	-	-	31.66
MultiGraph Transformer [41]	-	-	32.63
ResNet18 [39]	-	-	59.10
SketchSegNet [38]	44.86	25.44	-
MultiGraph Transformer [41]	51.09	27.81	-
SketchGNN [42]	77.96	76.65	-
SketchRecSeg(CFA)	84.18	83.56	76.48
SketchRecSeg(CFA+RSM)	84.96	84.37	76.48
SketchRecSeg(CFA+KLD)	85.45	84.87	77.21
SketchRecSeg(CFA+KLD+RSM)	86.14	85.58	77.21
SketchRecSeg+DA1	94.82	95.01	89.50
SketchRecSeg+DA2	96.16	96.31	92.85
SketchRecSeg+DA5	97.01	97.13	96.21

Besides, the SPG dataset [17] is also used to verify the advantages of the proposed networks. SPG has 25 categories and 800 sketches per category. Similar to SketchGNN [42], only 20 categories are used for evaluation. However, this study not only focuses on grouping strokes of one sketch into semantic parts, but also to recognize the semantic parts without knowing ahead the sketch's category label.

5.2 **Recognition and Segmentation**

Training Details. Image- and SVG-format sketches in SketchIME-SRS are fed to our SketchRecSeg network simultaneously. N = 300 points are sampled from each sketch. The learning rate is initialized to 2×10^{-3} with a batch size of 256. Adam optimizer is used for optimization. Total 100 epochs are implemented. Since the segmentation stream has N = 300 prediction outputs, λ_1 is set to 150 to balance recognition and segmentation losses. λ_2 is set to 1 or 0 depending on whether KL-divergence loss is used or not. The two-streams of SketchRecSeg are simultaneously trained from scratch. Our network is implemented in Pytorch and trained on a single NVIDIA GeForce GTX 3090.

Evaluation Metrics. Sketch recognition is a typical classification problem, therefore Top-1 accuracy (Acc@1) is used as the evaluation metric. Similar to [38] and [42], two metrics are selected



Figure 4: The interpretability analysis of the proposed SketchRecSeg. Rec.N and Rec.P denote the situations where the recognition network wrongly and correctly recognize sketches' categories, respectively. "Independent Rec-Seg" denotes the separate training of ResNet18 and SketchGNN networks. "Simultaneous Rec-Seg" denotes the proposed SketchRecSeg network. SketchRecSeg also achieves high segmentation performances when it correctly recognize sketches' categories.

for segmentation evaluation, including (a) Point-based accuracy (**P_Metric**) which evaluates the percentage of the correctly predicted stroke points in all sketches, and (b) Component-based accuracy (**C_Metric**) which evaluates the percentage of the correctly predicted strokes. A stroke is considered to be correctly predicted if more than 75% of its points are correctly predicted.

Benchmark Methods. To the best of our knowledge, the proposed SketchRecSeg is the first network that simultaneously recognizes and segments sketches. Therefore, some popular sketch recognition methods, e.g., CNN- [39] and Transformer-based [41] methods, are used for comparison. RNN-based recognition methods are not used in our experiments due to their sensitiveness to stroke orders. For segmentation, SketchSegNet [38] and SketchGNN [42] are evaluated for comparison. Besides, the multigraph Transformer [41] can easily be modified to support sketch segmentation, therefore it is also evaluated.

Performance Analysis. Table 3 illustrates the evaluation results. *Firstly*, the proposed simultaneous recognition and segmentation network achieves significant performance improvement, compared with the state-of-the-art methods. Vision Transformer [6] splits the image into multiple patches and uses linear layers to extract features from patches before inputting them to Transformer. The linear layers are not effective to learn two-dimensional features from images. Multigraph Transformer's positional embedding uses stroke orders [41], therefore it cannot achieve good performance when samples in the test subset have different stroke orders compared to the ones in the train subset. SketchSegNet [38] uses LSTM layers, and thus does not achieve good segmentation performance.

Secondly, the proposed SketchRecSeg uses ResNet18 and dynamic graph convolutions for feature learning. It achieves significant improvement both for recognition and segmentation tasks, compared with ResNet18 for recognition and SketchGNN for segmentation separately, even when SketchRecSeg only uses featurelevel supervision by CFA. This demonstrates that the two-stream network and the supervision can improve the recognition and segmentation performance simultaneously.

Thirdly, the use of the prior knowledge by the RSM module and KLD loss brings considerable performance improvement on the segmentation stream, without affecting the recognition performance. The computational and memory consumption by RSM and KLD is very little. The multilevel supervision takes full use of the prior knowledge and the advantages of CNN and GNN, and thus results in better performance.

Fourthly, Fig. 4 illustrates our analysis about the mutual effects of recognition and segmentation. The left two groups show the segmentation performance of SketchGNN on the samples for which the recognition network (i.e., ResNet18) makes the wrong (i.e., shown in the Rec.N bars under the scenario of independent Rec-Seg) and the right (i.e., shown in the Rec.P bars under the scenario of independent Rec-Seg) predictions. Note that ResNet18 and SketchGNN are trained independently. The right two groups give the performance of our simultaneous recognition and segmentation network. It can be seen that our network has very high segmentation accuracy when it correctly recognizes the sketch categories. While, it performs poorly for segmentation task when it cannot recognize the sketch correctly. This implies that it can recognize the sketch correctly because it can correctly segment the sketch's semantic components, and vice versa. This makes the network reliable due to its interpretability. In conclusion, the proposed SketchRecSeg achieves better and balanced recognition and segmentation performance simultaneously. Simultaneous recognition and segmentation shows its superiority compared to independent implementations.

Lastly, the bottom three rows of Table 3 show that supervised CDAN domain adaptation training using only one, two, five samples (corresponding to the cases of SketchRecSeg+DA1/DA2/DA5) per category of new users can significantly improve the performance to an application-level. It is reasonable to use few samples of the base training data for domain adaptation, since they can be viewed as the typical examples of each category for user reference. It is also reasonable to use few labeled samples of new users, since the labels can be obtained online when users select the right symbols from the recommended ones while using SketchIME. This provides us a few-shot domain adaptation method for personal sketching style learning, and ensures the adaptability of SketchIME. This also provides us a novel personal sketch style learning strategy to transfer the offline trained model to each new user' data for high-precision sketch recognition and segmentation.

Figure 5 shows some segmentation results on our SketchIME-SRS dataset. It visually shows the composition of a sketch's semantic component. It also demonstrates the advantages of the proposed SketchRecSeg.

5.3 Extendibility Analysis

Experimental Setup. Two experiments are performed on the SketchIME-CIL1 and SketchIME-CIL2 datasets. The former deals with the situation where both new sketch categories and new semantic components emerge in the incremental sessions, while the

MM '23, October 29-November 3, 2023, Ottawa, Canada



Figure 5: Visualization of some segmentation results on the SketchIME-SRS dataset. The numbers under the sketch samples denote the C-Metric values of each sample obtained by each method. It means correct segmentation only if the color of a stroke in each sample in the bottom-four rows matches the color of the same stroke in the top row.

Table 4: Comparison with state-of-the-art methods on SPG.

Networks	P-Metric	C-Metric	Acc@1
Vision Transformer[6]	-	-	76.21
BiGRU[4]	-	-	79.10
ResNet18 [39]	-	-	80.66
MultiGraph Transformer [41]	-	-	91.05
SketchSegNet [38]	56.22	45.46	-
SketchGNN [42]	91.26	87.86	-
SketchRecSeg(CFA)	93.18	90.86	97.37
SketchRecSeg(CFA+KLD)	93.83	91.25	97.47
SketchRecSeg(CFA+KLD+RSM)	94.09	91.65	97.47

latter deals with the situation where only new sketch categories emerge in the incremental sessions. The forward compatible training [47] of the recognition (i.e., ResNet18) and segmentation (i.e., SketchGNN) networks is performed separately as the baseline, and the results are illustrated in Fig. 6 as "Recognition Only" and "Segmentation Only". We also applied the forward compatible training [47] to support our two-stream network to incrementally learn new sketch categories and new semantic components simultaneously, and the results on recognition and segmentation are illustrated in Fig. 6 as "SketchRecSeg-Rec" and "SketchRecSeg-Seg" respectively. Anonymous et al.



Figure 6: Experimental results of class-incremental learning. The proposed SketchRecSeg using forward compatible training has a good extendibility to new task-specific classes.

Performance Analysis. It can be seen from Fig.6 that: (a) When dealing with the class-increment on both the sketch categories and semantic components, the joint optimization on the embedding spaces of categories and components can improve both recognition and segmentation performance (see Fig. 6(a)), and the recognition's supervision on segmentation significantly improves the segmentation performance, (b) When only dealing with the classincrement on the sketch categories, the joint optimization on the embedding spaces of categories and components can force the segmentation stream to learn how the existing semantic components constitute new categories and benefit from the learning process, (c) Fig. 6 reports slight performance reduction for both recognition and segmentation in incremental sessions. This means that applying the forward compatible training strategy of FACT [47] on our SketchRecSeg network can effectively enhance its extendibility to new task-specific sketches.

In conclusion, given the analysis in Section 5.2 and 5.3, conclusions can be obtained: (a) the two-stream architecture and the supervision between two streams can benefit recognition and segmentation simultaneously. (b) Applying the training strategies of few-shot domain adaptation and class-incremental learning can enhance the network's adaptability to new users and extendibility to new task-specific sketches, and enhance its practicality.

5.4 Comparison on Other Dataset

Table 4 illustrates the comparison results of our methods and stateof-the-art methods on the SPG dataset. SketchRecSeg outperforms both the recognition networks and the segmentation networks. Only subtle performance difference exists when SketchRecSeg uses a different supervision. This is because SketchRecSeg has already achieved excellent performance with CFA only.

6 DISCUSSION AND CONCLUSION

The traditional "Search and Select" input styles may be time consuming when frequent inputs are needed, especially when a large number of operational symbols are contained in hierarchical toolbars. Sketch input method editor (SketchIME) is a professional and promising system which can provide user-friendly and efficient input experience. Beyond the base recognition and segmentation functions, the adaptability and extendibility are also the key features to ensure SketchIME can be used by different users for various tasks. This study constructed datasets from scratch, proposed the simultaneous recognition and segmentation network, and explored the few-shot domain adaptation and class-incremental learning mechanism to improve the network's adaptability to new users and extendibility to new task-specific sketches. This study verifies the feasibility to use few samples to adapt to new users and tasks, and this paper provides systematic data and methods for SketchIME.

For an excellent sketch input method editor, online recognition and recommendation are more important. The unused parts of our collected data contains different stroke orders of each sketch. These can be used for online sketch recognition and personal identification based on sketching styles and will be released in our future works. MM '23, October 29-November 3, 2023, Ottawa, Canada

REFERENCES

- Andreas Bulling, Raimund Dachselt, Andrew Duchowski, Robert Jacob, Sophie Stellmach, and Veronica Sundstedt. 2012. Gaze interaction in the post-WIMP world. In CHI. 1221–1224.
- [2] Jungwoo Choi, Heeryon Cho, Jinjoo Song, and Sang Min Yoon. 2019. Sketchhelper: Real-time stroke guidance for freehand sketch retrieval. *IEEE TMM* 21, 8 (2019), 2083–2092.
- [3] Jungwoo Choi, Heeryon Cho, Jinjoo Song, and Sang Min Yoon. 2019. Sketch-Helper: Real-Time Stroke Guidance for Freehand Sketch Retrieval. *IEEE Transactions on Multimedia* 21, 8 (2019), 2083–2092. https://doi.org/10.1109/TMM.2019. 2892301
- [4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. (2014). arXiv:1412.3555 [cs.NE]
- [5] Ayan Das, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. 2021. Cloud2curve: Generation and vectorization of parametric sketches. In *CVPR*. 7088–7097.
- [6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. (2021). arXiv:2010.11929 [cs.CV]
- [7] Mathias Eitz, James Hays, and Marc Alexa. 2012. How do humans sketch objects? ACM TOG 31, 4 (2012), 1–10.
- [8] Danilo Gasques, Janet G Johnson, Tommy Sharkey, and Nadir Weibel. 2019. What you sketch is what you get: Quick and easy augmented reality prototyping with pintar. In CHI. 1–6.
- David Ha and Douglas Eck. 2018. A Neural Representation of Sketch Drawings. In ICLR.
- [10] Jun-Yan He, Xiao Wu, Yu-Gang Jiang, Bo Zhao, and Qiang Peng. 2017. Sketch recognition with deep visual-sequential fusion model. In ACM MM. 448–456.
- [11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In CVPR. 770–778.
- [12] Michael Hersche, Geethan Karunaratne, Giovanni Cherubini, Luca Benini, Abu Sebastian, and Abbas Rahimi. 2022. Constrained Few-shot Class-incremental Learning. In CVPR. 9057–9067.
- [13] Rui Hu and John Collomosse. 2013. A performance evaluation of gradient field hog descriptor for sketch based image retrieval. CVIU 117, 7 (2013), 790–806.
- [14] Forrest Huang, John F Canny, and Jeffrey Nichols. 2019. Swire: Sketch-based user interface retrieval. In CHI. 1–10.
- [15] Qi Jia, Meiyu Yu, Xin Fan, and Haojie Li. 2017. Sequential dual deep learning with shape and texture features for sketch recognition. arXiv preprint arXiv:1708.02716 (2017).
- [16] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. 2019. Deepgcns: Can gcns go as deep as cnns?. In CVPR. 9267–9276.
- [17] Ke Li, Kaiyue Pang, Jifei Song, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Honggang Zhang. 2018. Universal sketch perceptual grouping. In ECCV. 582–597.
- [18] Lei Li, Hongbo Fu, and Chiew-Lan Tai. 2018. Fast sketch segmentation and labeling with deep learning. *IEEE CG&A* 39, 2 (2018), 38–51.
- [19] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2018. Conditional adversarial domain adaptation. In *NeurIPS*. 1647–1657.
- [20] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014).
- [21] Ferran Naya, Manuel Contero, Nuria Aleixos, et al. 2007. ParSketch: a sketchbased interface for a 2D parametric geometry editor. In *ICHCI*. 115–124.
- [22] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. IEEE TKDE 22, 10 (2009), 1345–1359.
- [23] Ameya Prabhu, Vishal Batchu, Sri Aurobindo Munagala, Rohit Gajawada, and Anoop Namboodiri. 2018. Distribution-aware binarization of neural networks for sketch recognition. In WACV. 830–838.
- [24] Alan Preciado-Grijalva and Venkata Santosh Sai Ramireddy Muthireddy. 2021. Evaluation of Deep Neural Network Domain Adaptation Techniques for Image Recognition. arXiv preprint arXiv:2109.13420 (2021).
- [25] Yonggang Qi and Zheng-Hua Tan. 2019. SketchSegNet+: An end-to-end learning of RNN for multi-class sketch semantic segmentation. IEEE Access 7 (2019), 102717-102726.
- [26] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. The sketchy database: learning to retrieve badly drawn bunnies. ACM TOG 35, 4 (2016), 1–12.
- [27] Ravi Kiran Sarvadevabhatla and Jogendra Kundu. 2016. Enabling my robot to play pictionary: Recurrent neural networks for sketch recognition. In ACM MM. 247–251.
- [28] Ravi Kiran Sarvadevabhatla, Sudharshan Suresh, and R. Venkatesh Babu. 2017. Object Category Understanding via Eye Fixations on Freehand Sketches. *IEEE Transactions on Image Processing* 26, 5 (2017), 2508–2518. https://doi.org/10.1109/ TIP.2017.2675539

- [29] Sarah Suleri, Vinoth Pandian Sermuga Pandian, Svetlana Shishkovets, and Matthias Jarke. 2019. Eve: A sketch-based software prototyping workbench.
- In CHI. 1-6.
 [30] Baochen Sun, Jiashi Feng, and Kate Saenko. 2017. Correlation alignment for unsupervised domain adaptation. In Domain Adaptation in Computer Vision Applications, Gabriela Csurka (Ed.). Springer, 153–171.
- [31] Baochen Sun and Kate Saenko. 2016. Deep coral: Correlation alignment for deep domain adaptation. In ECCV. Springer, 443–450.
- [32] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. 2020. Few-shot class-incremental learning. In CVPR. 12183–12192.
- [33] Caglar Tirkaz, Berrin Yanikoglu, and T. Metin Sezgin. 2012. Sketched symbol recognition with auto-completion. *Pattern Recognition* 45, 11 (2012), 3926–3937. https://doi.org/10.1016/j.patcog.2012.04.026
- [34] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. arXiv preprint arXiv:1412.3474 (2014).
- [35] Fang Wang, Le Kang, and Yi Li. 2015. Sketch-based 3d shape retrieval using convolutional neural networks. In CVPR. 1875–1883.
- [36] Fei Wang, Shujin Lin, Hefeng Wu, Hanhui Li, Ruomei Wang, Xiaonan Luo, and Xiangjian He. 2019. Spfusionnet: Sketch segmentation using multi-modal data fusion. In *ICME*. 1654–1659.
- [37] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. ACM TOG 38, 5 (2019), 1–12.
- [38] Xingyuan Wu, Yonggang Qi, Jun Liu, and Jie Yang. 2018. Sketchsegnet: A rnn model for labeling sketch strokes. In MLSP. 1–6.
- [39] Peng Xu, Timothy M Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. 2022. Deep learning for free-hand sketch: A survey. IEEE TPAMI (2022).
- [40] Peng Xu, Yongye Huang, Tongtong Yuan, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, Zhanyu Ma, and Jun Guo. 2018. Sketchmate: Deep hashing for million-scale human sketch retrieval. In CVPR 8090–8098.
- [41] Peng Xu, Chaitanya K Joshi, and Xavier Bresson. 2021. Multigraph transformer for free-hand sketch recognition. *IEEE TNNLS* (2021).
- [42] Lumin Yang, Jiajie Zhuang, Hongbo Fu, Xiangzhi Wei, Kun Zhou, and Youyi Zheng. 2021. Sketchgnn: Semantic sketch segmentation with graph neural networks. ACM TOG 40, 3 (2021), 1–13.
- [43] Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. 2017. Sketch-a-net: A deep neural network that beats humans. *IJCV* 122, 3 (2017), 411–425.
- [44] Qian Yu, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. 2015. Sketch-a-Net that Beats Humans. In BMVC. 1–12.
- [45] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. 2021. Few-shot incremental learning with continually evolved classifiers. In CVPR. 12455–12464.
- [46] An Zhao, Mingyu Ding, Zhiwu Lu, Tao Xiang, Yulei Niu, Jiechao Guan, and Ji-Rong Wen. 2021. Domain-adaptive few-shot learning. In WACV. 1390–1399.
- [47] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, Liang Ma, Shiliang Pu, and De-Chuan Zhan. 2022. Forward compatible few-shot class-incremental learning. In CVPR. 9046–9056.
- [48] Da-Wei Zhou, Han-Jia Ye, Liang Ma, Di Xie, Shiliang Pu, and De-Chuan Zhan. 2022. Few-shot class-incremental learning by sampling multi-phase tasks. *IEEE TPAMI* (2022).
- [49] Kai Zhu, Yang Cao, Wei Zhai, Jie Cheng, and Zheng-Jun Zha. 2021. Self-promoted prototype refinement for few-shot class-incremental learning. In CVPR. 6801– 6810.