Vectorization and Rasterization: Self-Supervised Learning for Sketch and Handwriting

Ayan Kumar Bhunia[1] Pinaki Nath Chowdhury[1,3] Yongxin Yang[1,3] Timothy M. Hospedales[1,2]
Tao Xiang[1,3] Yi-Zhe Song[1,3]
[1]SketchX, CVSSP, University of Surrey, United Kingdom [2]University of Edinburgh, United Kingdom.
[3]iFlyTek-Surrey Joint Research Centre on Artificial Intelligence.
{a.bhunia, p.chowdhury, yongxin.yang, t.xiang, y.song}@surrey.ac.uk, t.hospedales@ed.ac.uk

Abstract

Self-supervised learning has gained prominence due to its efficacy at learning powerful representations from unlabelled data that achieve excellent performance on many challenging downstream tasks. However, supervision-free pre-text tasks are challenging to design and usually modality specific. Although there is a rich literature of self-supervised methods for either spatial (such as images) or temporal data (sound or text) modalities, a common pre-text task that benefits both modalities is largely missing. In this paper, we are interested in defining a self-supervised pre-text task for sketches and handwriting data. This data is uniquely characterised by its existence in dual modalities of rasterized images and vector coordinate sequences. We address and exploit this dual representation by proposing two novel cross-modal translation pre-text tasks for self-supervised feature learning: Vectorization and Rasterization. Vectorization learns to map image space to vector coordinates and rasterization maps vector coordinates to image space. We show that our learned encoder modules benefit both raster-based and vector-based downstream approaches to analysing hand-drawn data. Empirical evidence shows that our novel pre-text tasks surpass existing single and multi-modal self-supervision methods.

1. Introduction

Deep learning architectures [65, 24] have become the de-facto choice for most computer vision applications. However, their success heavily depends on access to large scale labelled datasets [51] that are both costly and time-consuming to collect. In order to alleviate the data annotation bottleneck, many unsupervised methods [43, 31, 15, 10, 23] propose to pre-train a good feature representation from large scale unlabelled data. A common approach is to define a *pre-text task* whose labels can be obtained free-of-cost, e.g. colorization [66], jigsaw solving [43], image rotation prediction [18], etc. The motivation is that a network trained to solve such a pre-text task should encode high-level semantic understanding of the data that can be used to solve other downstream tasks like classification, retrieval, etc. Apart from traditional object classification, detection or semantic segmentation, self-supervision has been extended to sub-domains like human pose-estimation [32], co-part segmentation [27], and depth estimation [19].

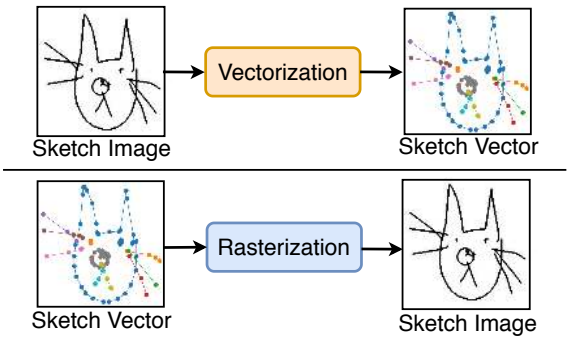

Figure 1. Schematic of our proposed self-supervised method for sketches. Vectorization drives representation learning for sketch images; rasterization is the pre-text task for sketch vectors.

In this paper, we propose a self-supervised method for a class of visual data that is distinctively different than photos: sketches [65, 55] and handwriting [50] images. Although sketch and handwriting have been studied as two separate topics by different communities, there exists an underlying similarity in how they are captured and represented. More specifically, they are both recorded as the user's pen tip follows a trajectory on the canvas, and rendered as sparse black and white lines in image space. Both are abstract, in the sense that the same object or grapheme can be drawn in many possible ways [57, 22]; while sketch in particular poses the challenge of variable levels of detail [52] depicted. Both sketch [62] and handwriting [20] can be represented in rasterized pixel space, or as a temporal point sequence [22]. While each modality has its own benefits, we propose a novel self-supervised task that takes advantage of this dual image/vector space representation. In particular,

we use cross modal translation between image and vector space as a self-supervised task to improve downstream performance using either representation (Figure 1).

Most existing self-supervised methods are defined for single data modalities. Existing methods for images [11, 23, 18] or videos [34] are designed for pixel perfect renderings of scenes or objects, and as such are not suited for sparse black and white handwritten images. For example colorization [66] and super-resolution [35] pre-texts, and augmentation strategies such as color distortion, brightness, and hue adjustment used by state of the art contrastive methods [23, 11, 21] – are not directly applicable to line drawings. For vector sequences, self-supervised methods typically addressed at speech such as Contrastive Predictive Coding (CPC) [25] could be used off-the-shelf but do not explicitly handle the stroke-by-stroke nature of handwriting. Conversely, BERT-like pre-training strategies have had some success with vector-modality sketches [37] but cannot be applied to image-modality sketches. In contrast, our framework can be used to learn a powerful representation for both image and vector domain sketch analysis tasks. Although a multi-view extension of contrastive learning for self-supervision [59] has been attempted, we show empirically that our cross-view rasterization/ vectorization synthesis approach provides a superior self-supervision strategy.

In summary, we design a novel self-supervised framework that exploits the dual raster/vector sequence nature of sketches and handwritten data through cross-modal translation (Figure 1). Our cross-modal framework is simple and easy to implement from off-the-shelf components. Nevertheless it learns powerful representations for both raster and vector represented downstream sketch and handwriting analysis tasks. Empirically, our framework surpasses state of the art self-supervised methods and approaches and sometimes surpasses the fully supervised alternatives.

2. Related Works

Sketch Representation Learning: Learning good sketch representations benefits a variety of sketch-specific problems like classification [64], retrieval [62], scene understanding [38], sketch based image retrieval [7, 45, 46, 14, 16, 4, 53, 52], generative sketch modelling [57, 22, 6] etc. While photos are pixel perfect depictions represented by 2D spatial matrices, sketches can be described either as 2D static pixel level *rasterized images* or vector sketches with an ordered sequence of *point coordinates*. Typically, sketch images are processed by convolutional neural networks [7, 14], whereas vector sketches needs Recurrent Neural Networks (RNNs) or Transformers [62, 54, 37] for sequential modelling. There exists no consensus on which sketch modality (image or vector) is better than the other, as each has its own merits based on the application scenario. While rasterized sketch images are usually claimed to be better for driving fine-grained retrieval [64, 58], they fail to model the varying level of abstraction [57] in the sketch generative process. Conversely, vector sketches are more effective to simulate the human sketching style [22] for generation, however, it fails [7] for fine-grained instance level image retrieval. From a computational cost perspective, coordinate based models provides faster cost-effective real-time performance [63] for sketch-based human-computer interaction, compared to using rasterized sketch images that impose a costly rendering step and transfer cost of a large pixel array. Attempts have been directed towards combining representations from both sketch images and vector sketches for improved performance in sketch hashing and category level sketch-based image retrieval [62].

Image-Net pretrained weights are widely used to initialise standard convolutional networks for sketch images, with the first self-supervised alternatives specifically designed for raster sketch images being proposed recently [46]. Vector sketches relying on RNN or Transformer do not have the ImageNet initialization option. Thus, Lin *et al*. [37] employ BERT-like self-supervised learning on vector sketches. Nevertheless, these existing self-supervised are proposed for specific modalities (raster image vs vector sketches) and do not generalize to each other. We therefore propose a unified pipeline that leverages this *dual representation of sketches* to learn powerful features for encoding sketches represented in both vector and raster views.

Handwriting Recognition: Similar to sketches, handwriting recognition has also been heavily explored involving both image space ('offline' recognition) [50, 5, 8, 39, 61] and vector space ('online' recognition) [26, 20]. Unlike sketch, there is a consensus in the handwriting community [9, 30] that vector representation of handwriting provides better recognition accuracy over offline images. Connectionist temporal Classification (CTC) criterion by Graves *et al*. [20] made end-to-end sequence discriminative learning possible. Following this seminal work [20], earlier handcrafted feature extraction methods [3, 26, 1] in both the modalities have now been replaced by data driven feature learning [5, 50, 9] as in many computer vision domains. Nevertheless, data scarcity still remains a bottleneck for both offline and online handwriting recognition, despite advances such as modelling handwriting style variation via adversarial feature deformation module [5] or learning an optimal augmentation strategy [39] using reinforcement learning. We demonstrate the first use of self-supervision to improve both offline and online handwriting recognition.

Self-supervised Learning: Self-supervised learning is now a large field, too big to review in detail here, with recent surveys [29] providing a broader overview. As a brief review: Generative models such as VAEs [31] learn representations by modelling the distribution of the data. Contrastive learning [21, 11, 23] aims to learn discriminative

features by minimising the distance between different augmented views of the same image while maximizing it for views from different images. Clustering based approaches [10] first cluster the data based on the features extracted from a network, followed by re-training the same network using the cluster-index as pseudo-labels for classification.

Different pre-text tasks have been explored for self-supervised feature learning in imagery, e.g., image colorization [66], super-resolution [35], solving jigsaw puzzles [43, 46], in-painting [49], relative patch location prediction [15], frame order recognition [42], etc. Compared to these approaches, our work is more similar to the few approaches addressing multi-modal data. For instance, pre-text tasks like visual-audio correspondence [2, 34], or RGB-flow-depth correspondence [59] within vision. However, these approaches use contrastive losses, which raise a host of complex design issues in batch size, batch sampling strategies, and positive/negative balancing [11, 48, 23, 25, 59] that are necessary to tune, in order to obtain good performance. Furthermore they tend to be extremely expensive to scale due to the ultimately quadratic cost of comparing sample *pairs* [11, 48, 59, 36]. In contrast, our simple cross-modal synthesis avoids all of these design issues and compute costs, while achieving state of the art performance in both vector and raster view downstream performance.

3. Methodology

Overview: Our objective is to design a self-supervised learning method that can be applied over both rasterized image and vector representation of any hand drawn data (e.g., sketch or handwriting); and furthermore it should exploit this complementary information for self-supervised learning. Towards this objective, we pose the feature learning task as a cross-modal translation between image and vector space using state-of-the-art encoder-decoder architectures. In other words, the training objective is to learn a latent space for the source modality from which the corresponding sample in the target modality is predictable. Once the cross-modal translation model is trained, we can remove the decoder and use the encoder as a feature extractor for source modality data. For instance, learning to translate from image space to vector space, we obtain an encoder that can embed raster encoding of hand drawn images into a meaningful latent representation, and vice-versa.

Touch-screen devices and stylus-pens give us easy access to hand drawn data represented in both modalities simultaneously. Our training dataset consists of N samples $\{I_i, V_i\}_{i=1}^{N}$, where $I \in \mathcal{I}$ and $V \in \mathcal{V}$ are rasterized image and vector representation respectively. In particular, I is a spatially extended image of size $\mathbb{R}^{H\times W\times 3}$, and V is a sequence of pen states $(v_1, v_2, \cdots, v_T)$, where T is the length of the sequence. In order to learn feature representation on image space, we learn a *vectorization* operation $\mathcal{I} \mapsto \mathcal{V}$. Conversely, a *rasterization* operation $\mathcal{V} \mapsto \mathcal{I}$ is trained to provide a vector space representation. It is important to note that we do not use any category-label for sketch data or character/word annotation for handwritten data in our feature learning process. Thus, it can be trained in a class agnostic manner without any manual labels, satisfying the criteria of self-supervised learning.

3.1. Model Architecture

For cross-modal translation, encoder $E(\cdot)$ embeds the data from source modality into a latent representation, and decoder $D(\cdot)$ reconstructs the target modality given the latent vector. $E(\cdot)$ and $D(\cdot)$ will be designed differently for the different source and target modalities. While rasterized image space is represented by a three-channel RGB image $\mathbb{R}^{H\times W\times 3}$, we use five-element vector $v_t = (x_t, y_t, q_t^1, q_t^2, q_t^3)$ to represent pen states in stroke-level modelling. In particular, (x_t, y_t) is absolute coordinate value in a normalised $H \times W$ canvas, while the last three elements represent binary one-hot vector [22] of three pen-state situations: pen touching the paper, pen being lifted and end of drawing. Thus, the size of vector representation is $V \in \mathbb{R}^{T\times 5}$, where T is the sequence length.

Vectorization: For translating an image to its sequential point coordinate equivalent, image encoder $E_I(\cdot)$ can be any state-of-the-art convolutional neural network [33] such as ResNet. To predict the sequential point coordinates, decoder $D_V(\cdot)$ could be any sequential network, e.g. RNN. In particular, given an image I, let the extracted convolutional feature map be $F = E_I(I) \in \mathbb{R}^{h\times w\times d}$, where h, w and d signify height, width and number of channels respectively. Applying global max pooling (GAP) to F and flattening, we obtain a vector l_I of size $\mathbb{R}^d$, which will be used as the representation for input image I once the encoder-decoder model is trained. Next, a linear-embedding layer is used to obtain the initial hidden state of the decoder RNN as follows: $h_0 = W_h l_I + b_h$. The hidden state h_t of decoder RNN is updated as follows: $h_t = RNN(h_{t-1}; [l_I, P_{t-1}])$, where P_{t-1} is the last predicted point and $[\cdot]$ stands for a concatenation operation. Thereafter, a fully-connected layer is used to predict five-element vector at each time step as: $P_t = W_y h_t + b_y$, where $P_t = (x_t, y_t, q_t^1, q_t^2, q_t^3)$ is of size $\mathbb{R}^{2+3}$, whose first two logits represent absolute coordinate (x, y) and the latter three for pen's state position (q^1, q^2, q^3). We use simple mean-square error and categorical cross-entropy losses to train the absolute coordinate and pen state prediction (softmax normalised) respectively. Thus, $(\hat{x}_t, \hat{y}_t, \hat{q}_t^1, \hat{q}_t^2, \hat{q}_t^3)$ being the ground-truth coordinate at t-th step, the training loss is:

$$\begin{aligned} L_{I\to V} =& \frac{1}{T}\sum_{t=1}^{T} \|\hat{x}_t - x_t\|_2 + \|\hat{y}_t - y_t\|_2 \\ & -\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{3} \hat{q_t^i} \log\Big(\frac{\exp(q_t^i)}{\sum_{j=1}^{3}\exp(q_t^j)}\Big) \end{aligned} \tag{1}$$

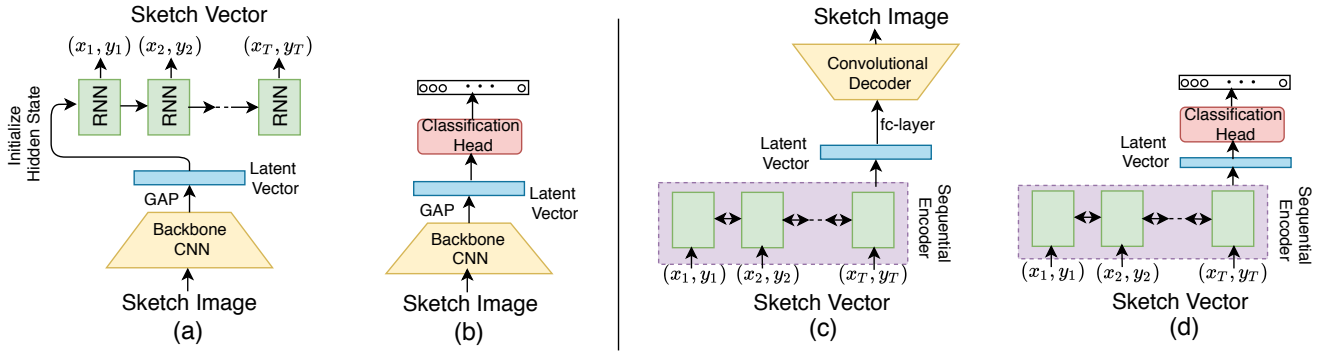

Figure 2. Illustration of the architecture used for our self-supervised task for sketches and handwritten data (a,c), and how it can subsequently be adopted for downstream tasks (b,d). Vectorization involves translating sketch image to sketch vector (a), and the convolutional encoder used in the vectorization process acts as a feature extractor over sketch images for downstream tasks (b). On the other side, rasterization converts sketch vector to sketch image (c), and provides an encoding for vector-based recognition tasks downstream (d).

Rasterization: To translate a sequence of point coordinates V to its equivalent image representation I, any sequential network such as RNN [12, 20] or Transformer [37], could be used as the encoder $E_V(\cdot)$, and we experiment with both. For RNN-like architectures [22], we feed the five elements vector v_t at every time step, and take the hidden state of final time step as the encoded latent representation. For Transformer like encoders, we take input via a trainable linear layer to convert each five element vector to the Transformer's model dimension. Additionally, we prepend a learnable embedding to the input sequence, similar to BERT's *class token* [13], whose state at the output acts as the encoded latent representation. Finally, the encoder latent representation $l_V \in \mathbb{R}^d$ is fed via a fully-connected layer to a standard convolutional decoder $D_I(\cdot)$. $D_I(\cdot)$ consists of series of fractionally-strided convolutional layers [28] to up-sample the spatial size to $H \times W$ at the output. Given the vector V and raster I data pairs, we use mean-square error as the training objective:

$$L_{V \to I} = -\mathbb{E}_{(I,V)\sim(\mathcal{I},\mathcal{V})} \left\| I - D_I(E_V(V)) \right\|_2 \quad (2)$$

We remark that due to the well known regression to mean problem [41], the generated images are indeed blurry. Adding an adversarial loss [28] does not give any improvement in our representation learning task, and sometimes leads to worse results due to mode collapse issue in adversarial learning. However, synthesising realistic images is not our goal in this work. Rather, it is a pretext task for learning latent representations for vector sequence inputs.

3.2. Application of Learned Representation

We apply our self-supervised learning method on both sketch [22, 17] and handwriting data [40], as both can be represented in image and vector space.

Sketch Analysis: We use sketch-recognition [65] and sketch-retrieval [62, 37] as downstream tasks to evaluate the quality of learned latent representation obtained by our self-supervised pre-training. For both classification and retrieval, we evaluate performance with both sketch image and sketch vector representations using vectorization and rasterization as pre-training task respectively. For classification, we simply apply a fully-connected layer with softmax on the extracted latent representation from pre-trained encoder. For retrieval, we could use the latent representation itself. However we find it helpful to project the latent feature through a fully connected layer into 256 dimensional embedding space, and optimise the model through triplet loss [14, 7]. Along with triplet loss, that minimises intra-class distance while maximising inter-class distance, we also apply a classification loss through a linear layer [12] to further aid the retrieval learning framework.

Handwriting Recognition: As handwriting recognition [5] is a (character) sequence task oriented at decoding a whole word from an image, we use a slightly modified vectorization encoder $E_I(\cdot)$ for offline/image recognition compared to the sketch tasks. Following [56], the (word) image encoder extends a conventional ResNet architecture with a 2-layer BLSTM image feature encoder before producing a final state that provides the latent vector for the input image. This feature is then fed to a sequential decoder to 'sketch' the word during cross-modal self-supervised pre-training, and to a recognition model to recognise the word in the downstream task. In contrast, the rasterization encoder $E_V(\cdot)$ is defined similarly as for the sketch tasks. For the downstream task, after encoding either vector and raster inputs, we follow [44] in using an attentional decoder [56, 67] to recognize the word by predicting the characters sequentially. This decoder module consists of a BLSTM layer followed by a GRU layer that predicts the characters.

4. Experiments

Datasets: For sketches, we use the standard QuickDraw [22] and TU-Berlin [17] datasets for evaluation as they contain both raster and vector image representations. QuickDraw contains 50 million sketches from 345 classes. We use the split from [22] where each class has 70K training samples, 2.5K validation, and 2.5K test samples. Mean-

while, TU-Berlin comprises of 250 object categories with 80 sketches in each category. We apply the Ramer-Douglas-Peucker (RDP) algorithm to simplify the sketches. For handwriting we use IAM offline and online datasets [40]: The offline set contains 115,320 word images, while the online set contains point coordinate representation of $13,049$ lines of handwriting. We pre-process line-level online data to segment it into $70,648$ valid words, and use synthetic rasterization to create training data for vector and raster views.

Implementation Details: We implemented our framework in PyTorch [47] and conducted experiments on a 11 GB Nvidia RTX 2080-Ti GPU. While a GRU decoder of hidden state size 512 is used in all the vectorization process, we use convolutional decoder from [28] in the rasterization process. Following a recent self-supervised study analysis [33, 21], we use ResNet50 as the CNN encode images, unless otherwise mentioned. For vector sketch recognition, we use a Transformer [60] encoder with 8 layers, hidden state size 768, MLP size 2048, and 12 heads. For offline handwritten images, the encoder architecture is taken from [56] and comprises a ResNet like convolutional architecture followed by a 2 layers BLSTM. For online handwriting, we feed 5-element vectors at every time step of a 4-layers stacked BLSTM [9] with hidden state size 512. We use Adam optimiser with learning 0.0001 and batch size of 64 for all experiments.

Evaluation Metrics: For sketch recognition, Top-1 and Top-5 accuracy is used, and for category level sketch-retrieval, we employ Acc@top1 and mAP@top10 as the evaluation metric. For handwriting recognition, we use Word Recognition Accuracy following [5].

Competitors: We compare with existing self-supervised learning methods that involve pre-text task like context prediction [15], Auto-Encoding [31], jigsaw solving [43], rotation prediction [18]. Clustering based representation learning Deep Cluster [10] is also validated on sketch datasets. Furthermore, we compare with three state-of-the-art contrastive learning based self-supervised learning methods, namely, SimCLR [11], MoCo [23], and BYOL [21]. While these self-supervised learning methods are oriented at RGB photos rather than sketch or handwritten data, we also compare with Sketch-Bert [37] as the only work employing self-supervised learning on vector sketches. We note that self-supervised methods designed for image data can not be used off-the-shelf for vector sketches. The only exception is Contrastive Predictive Coding (CPC) [44] which has been used to handle both images and sequential data (e.g. speech signals). Finally, we compare with a state of the art multi-modal self-supervised method Contrastive Multi-view Coding (CMC) [59], which performs contrastive learning of (mis)matching instances across modalities. Here, we use the same encoder for raster sketch and vector sketch like ours for a fair comparison.

4.1. Results on Sketch Representation Learning

Sketch Recognition: Following the traditional protocol of evaluation for self-supervised learning [11, 21], we first evaluate our representations by training a linear classifier on the top of frozen representation. We report the recognition accuracy in Table(left) 1. On QuickDraw, Top-1 accuracy of 71.9% and 67.2% is obtained for sketch images and sketch vectors respectively, approaching the supervised counterparts of 76.1% and 73.5%. For TU-Berlin accuracies of 70.6% and 55.6% also approach the supervised figures 78.6% and 62.9%. The gap with supervised method is larger for TU-Berlin dataset because of having less data compared to QuickDraw dataset. Interestingly, the performance over image level data is comparatively better than using sketch vectors.

We next evaluate the semi-supervised setup, where we fine-tune the whole network using smaller subset of training data, 1% and 10%. Our self-supervised methods helps to learn good initialization such that in this low data regime, ours is significantly better than its supervised counter part as shown in Table 2. Finally, we evaluate the learned features from various depths of our convolutional encoder for sketch raster image classification in Table 3. Overall, our close competitors are contrastive learning based family of self-supervised methods, e.g. SimCLR, BYOL, MoCo. We attribute the superiority of our method over other self-supervised methods, on the sketch dataset, to the task-design that exploits the intrinsic dual representation of sketch data.

Sketch Retrieval: For sketch retrieval, first we use the extracted latent feature from pre-trained self-supervised network for triplet metric learning of an additional linear embedding layer. From the retrieval performance in Table 1 (right) we see a relative performance between the methods that is similar to the previous sketch classification experiments. However, the retrieval performance using the fixed self-supervised latent feature is $9-10\%$ below the supervised version. In the semi-supervised experiment, we fine-tune the complete model including linear layer and the pre-trained feature extractor using 1% and 10% of the training data respectively. Table 2 shows that our self-supervised method has a clear edge over supervised counter part in this low data regime. Qualitative cross-modal generated and retrieved results are shown in Figure 4 & 5, respectively.

4.2. Results on Handwriting Recognition

To the best of our knowledge, there has been no work applying self-supervised learning to handwritten data. We compare our self-supervised method with CPC which can handle sequential data. In Table 4, we use the extracted frozen sequential feature from each encoder to train an attentional decoder based text recognition network. We see that our Sketch2Vec surpasses CPC, but both methods do

Table 1. Linear model evaluation of fixed pre-trained features. ResNet50 for image space and Transformer for vector space inputs.

	Recognition								Retrieval							
	Image Space				Vector Space				Image Space				Vector Space			
	QuickDraw		TU-Berlin		QuickDraw		TU-Berlin		QuickDraw		TU-Berlin		QuickDraw		TU-Berlin	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	A@T1	mAP@t10	A@T1	mAP@t10	A@T1	mAP@T10	A@T1	mAP@T10
Supervised	76.1%	91.3%	78.6%	90.1%	73.5%	90.1%	62.9%	80.7%	62.3%	69.4%	69.1%	74.7%	58.5%	77.1%	50.2%	67.4%
Random	15.5%	26.2%	18.4%	29.3%	12.7%	23.6%	9.6%	19.4%	10.6%	21.3%	13.4%	26.7%	9.8%	21.5%	9.2%	17.6%
Context [15]	44.6%	69.2%	43.3%	67.5%	-	-	-	-	30.7%	34.9%	28.4%	32.7%	-	-	-	-
Auto-Encoder [31]	26.4%	48.1%	22.6%	47.5%	-	-	-	-	16.4%	24.4%	15.3%	20.4%	-	-	-	-
Jigsaw [43]	46.9%	71.5%	45.7%	69.8%	-	-	-	-	31.6%	38.9%	30.6%	35.4%	-	-	-	-
Rotation [18]	53.5%	78.7%	51.2%	77.1%	-	-	-	-	37.5%	45.1%	36.4%	41.8%	-	-	-	-
Deep Cluster [10]	39.4%	62.7%	38.7%	60.2%	-	-	-	-	29.2%	36.8%	27.3%	31.9%	-	-	-	-
MoCo [23]	65.7%	85.1%	64.3%	82.8%	-	-	-	-	42.5%	46.8%	42.5%	46.9%	-	-	-	-
SimCLR [11]	65.5%	85.1%	64.3%	82.9%	-	-	-	-	43.3%	50.7%	41.5%	46.7%	-	-	-	-
BYOL [21]	66.8%	85.8%	65.7%	83.7%	-	-	-	-	45.4%	52.5%	43.8%	49.1%	-	-	-	-
Sketch-BERT [37]	-	-	-	-	65.6%	85.3%	52.9%	78.1%	-	-	-	-	48.9%	68.1%	40.7%	58.8%
CMC [59]	63.6%	83.9%	61.7%	81.3%	61.2%	81.5%	51.4%	77.5%	40.6%	45.8%	38.5%	43.3%	45.2%	66.7%	40.3%	58.2%
CPC [44]	54.3%	79.0%	52.9%	77.9%	59.3%	81.3%	50.5%	76.6%	37.9%	43.1%	36.4%	40.9%	43.1%	63.6%	39.3%	57.9%
Ours-(L)	71.9%	89.7%	70.6%	85.9%	67.2%	86.5%	55.6%	79.4%	52.3%	59.5%	47.7%	59.1%	49.5%	68.9%	42.1%	59.6%

Table 2. Semi-supervised fine-tuning using 1% and 10% labelled training data on QuickDraw.

	Recognition								Retrieval							
	Image Space				Vector Space				Image Space				Vector Space			
	1% Training		10% Training		1% Training		10% Training		1% Training		10% Training		1% Training		10% Training	
	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	A@T1	mAP@t10	A@T1	mAP@t10	A@T1	mAP@T10	A@T1	mAP@T10
Supervised	25.1%	47.3%	55.4%	79.0%	17.3%	37.5%	43.9%	65.9%	13.4%	34.3%	43.9%	63.7%	9.1%	29.0%	41.0%	60.8%
Context [15]	33.9%	55.8%	56.8%	80.5%	-	-	-	-	24.4%	30.5%	42.6%	48.4%	-	-	-	-
Auto-Encoder [31]	21.5%	40.7%	45.1%	70.6%	-	-	-	-	15.2%	21.4%	32.7%	37.6%	-	-	-	-
Jigsaw [43]	36.5%	57.4%	57.4%	80.3%	-	-	-	-	27.7%	35.2%	44.7%	51.2%	-	-	-	-
Rotation [18]	38.8%	59.1%	59.6%	80.7%	-	-	-	-	28.4%	35.2%	44.7%	51.8%	-	-	-	-
Deep Cluster [10]	32.2%	54.5%	54.7%	79.2%	-	-	-	-	24.4%	31.2%	43.6%	47.7%	-	-	-	-
MoCo [23]	46.0%	70.5%	62.2%	83.7%	-	-	-	-	35.9%	43.1%	52.7%	57.4%	-	-	-	-
SimCLR [11]	46.1%	70.5%	62.1%	83.6%	-	-	-	-	35.1%	42.7%	52.3%	57.4%	-	-	-	-
BYOL [21]	47.3%	72.0%	62.7%	84.1%	-	-	-	-	36.5%	43.0%	52.8%	59.8%	-	-	-	-
Sketch-BERT [37]	-	-	-	-	45.1%	69.8%	62.4%	81.7%	-	-	-	-	36.5%	60.0%	52.9%	72.9%
CMC [59]	44.6%	68.2%	61.7%	82.7%	44.6%	68.4%	61.7%	81.6%	34.7%	41.9%	51.1%	57.4%	35.4%	58.1%	52.6%	72.8%
CPC [44]	40.6%	65.7%	60.7%	81.9%	43.5%	67.7%	61.6%	81.7%	33.4%	40.1%	50.5%	57.7%	34.1%	56.6%	52.3%	72.8%
Ours	51.2%	76.4%	65.6%	85.2%	46.8%	70.9%	63.2%	83.9%	38.6%	45.6%	60.4%	81.4%	37.1%	61.5%	53.2%	74.3%

Table 3. Accuracy on QuickDraw dataset with linear classifier trained on representation from various depth within the network.

Method	Block1	Block2	Block3	Block4	Pre-logits
Supervised	7.0%	14.9%	35.6%	72.5%	76.1%
Jigsaw [43]	4.2%	8.1%	26.8%	39.9%	46.9%
Rotation [18]	5.2%	11.2%	27.5%	45.4%	53.5%
Deep Cluster [10]	4.1%	8.6%	19.6%	33.4%	39.4%
CMC [59]	6.1%	11.9%	29.7%	56.7%	63.6%
MoCo [23]	7.7%	13.5%	31.6%	60.3%	65.7%
SimCLR [11]	6.7%	12.6%	32.0%	59.5%	65.5%
BYOL [21]	9.0%	15.1%	32.2%	61.2%	66.8%
Ours	10.1	15.2%	34.4%	67.5%	71.9%

Table 4. Handwriting recognition using feature extracted from fixed pre-trained encoder.

	Offline		Online	
	Lexicon	No Lexicon	Lexicon	No Lexicon
Supervised [56]	87.1%	81.5%	88.4%	82.8%
Random	10.4%	6.3%	7.4%	4.9%
CPC [44]	72.2%	63.7%	71.5%	62.8%
Ours	75.4%	68.6%	73.1%	66.9%

Table 5. Handwriting recognition under semi-supervised setup.

	Offline		Online	
	1% Training	10% Training	1% Training	10% Training
Supervised [56]	19.7%	40.6%	20.5%	42.4%
CPC [44]	29.1%	55.4%	27.8%	54.2%
Ours	38.5%	59.2%	36.8%	56.7%

not match supervised performance. In a semi-supervised setup (Table 5), we add an attentional decoder and fine-tune the whole pipeline using 1% and 10% training data respectively. In this case, the self-supervised methods achieve a significant margin over the supervised alternative. Furthermore, we observe that initialising the network (both offline and online) with weights pre-trained on our self-supervised setup, followed by training (supervised) the entire pipeline, yields higher results than initialising with random weights, by a margin of 1.4% and 1.2%, under the same experimental setup. This concludes that our smart pre-training strategy is a better option, instead of training handwriting recognition network from scratch.

Table 6. Ablative study (Top-1 accuracy) on architectural design using QuickDraw. (V)ectorization and (R)asterization indicate representation learning on image and vector space, respectively.

Ablation Experiment	Image Space	Vector Space
(a) Absolute coordinate in the decoding (V):	71.9%	–
(b) Offset coordinate in the decoding (V)	69.5%	–
(c) Absolute coordinate in the encoding (R):	–	67.2%
(d) Offset coordinate in the encoding (R)	–	67.1%
(e) LSTM decoder (V) :	70.7%	–
(f) GRU decoder (V) :	71.9%	–
(g) Transformer decoder (V) :	68.6%	–
(h) LSTM encoder (R) :	–	66.7%
(i) GRU encoder (R) :	–	66.1%
(j) Transformer encoder (R) :	–	67.2%
(k) Two-way Translation (V+R) :	70.3%	66.1%
(l) Attentional Decoder (V) :	68.0%	–

4.3. Ablative Study

Data Volume and Layer Dependence: Performance under varying amounts of training data is shown in Figure 3 for both sketch classification and handwriting recognition.

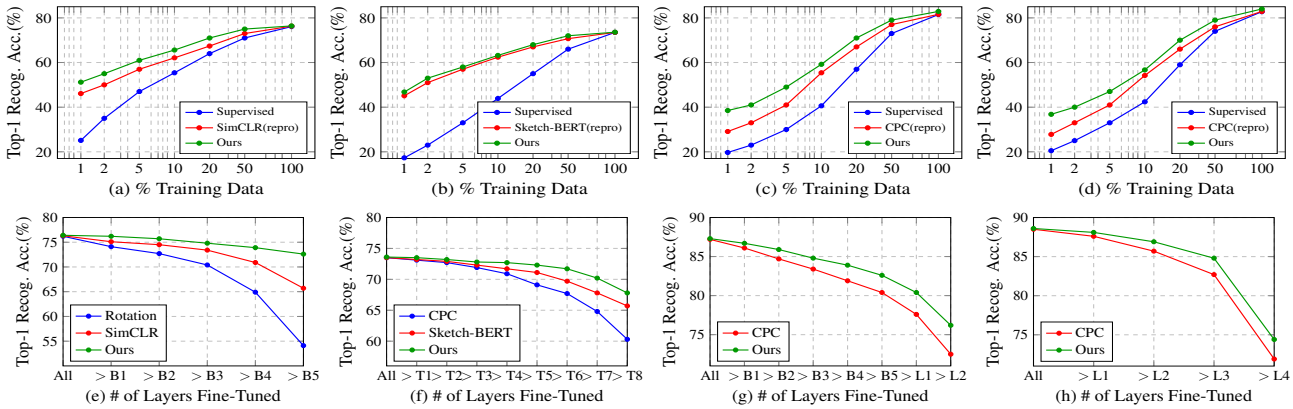

Figure 3. Performance at varying training data size for (a) sketch image classification (b) sketch vector classification on Quick-Draw, and (c) offline handwritten image recognition (d) online handwriting recognition, respectively. In the same order, comparative performance is shown through fine-tuning different number of layers: (e) sketch image encoder uses ResNet-50 having 5-convolutional blocks. (f) 8-layers stacked Transformer is used for sketch vector encoder. (g) ResNet-like convolutional encoder (having 5 blocks) followed by 2 layers BLSTM employs offline word image encoder (h) 4 layers stacked BLSTM is used for encoder online word images.'$>$ X' represents all layers above X are fine-tuned.

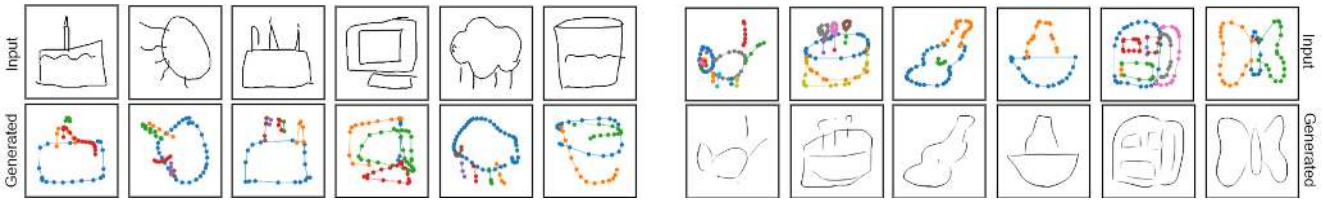

Figure 4. Qualitative results showing generated cross-modal translation. (a) Vectorization: raster sketch image to vector sketch, (b) Rasterization: vector sketch to raster sketch image.

We also evaluate performance as a function of number of trained/frozen layers during fine-tuning. We can see that Sketch2Vec performs favorably to state of the art alternatives SimCLR, SketchBERT, and CPC – especially in the low data, or few tuneable layers regimes.

Architectural Insights: We perform a thorough ablative study to provide insights on our architecture design choices using the sketch recognition task in Table 6. (i) In the sketch image to sketch vector translation process, using *absolute coordinate* is found to give better representative feature for sketch images over using offset coordinate [22] values. However, for representation learning over vector sketches, we did not notice any significant difference in performance, provided the absolute coordinates are normalised. (ii) We found absolute coordinate with regression loss gives better performance than using offset with log-likelihood loss as used in [57]. (iii) We use deterministic cross-modal encoder-decoder architecture since VAE-based design [57] reduces the performance. (iv) We also compare with different sequential decoders in the vectorization process, e.g. LSTM, GRU, and Transformer. Empirically, GRU is found to work better than others. (v) For sketch classification on vector space, we also compare with LSTM, GRU, and Transformer encoder architecture respectively, with Transformer giving optimum results. (vi) Another intuitive alternative could be to use two-way cross-modal translation using additional source-to-source and target-to-target decoder, however, we experience performance drop. (vii) We also add an attentional block for sequential decoding in vectorization process that leads to a drop in performance by 3.9%. We conjecture that adding attention gives a shortcut connection to the convolutional feature map, and the sequential task becomes comparatively easy, which is why the self-supervised pre-training fails to learn global semantic representation for classification.

Cross-category Generalisation: One major objective of unsupervised representation learning is to learn feature representation that can generalise to other categories as well. Thus, we split 345 QuickDraw classes into two random disjoint set [14] of 265 and 80 for self-supervised training and evaluation, respectively. Model trained using our self-supervised task, is further evaluated on unseen classes (Table 7) using a linear classifier on extracted frozen feature. We obtain a top-1 accuracy of 65.1% and 58.4% on sketch images and vectors, respectively, compared to 71.9% and 67.2% while using all classes in the self-supervised pre-training. Under same setting, SimCLR is limited to 53.6% for sketch image classification. This confirms a significant extent of generalizable feature learning through our self-supervised task over sketch data.

Cross-dataset Generalisation: We further use model trained on QuickDraw dataset to extract feature over TU-

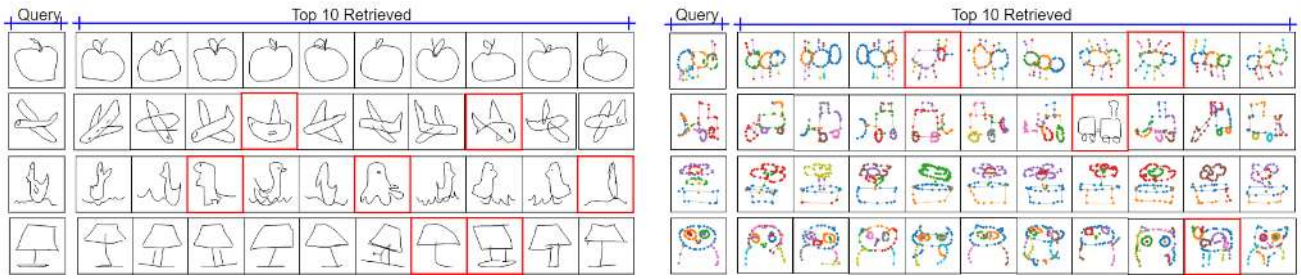

Figure 5. Qualitative retrieved results on (a) raster sketch images (via vectorization task) (b) vector sketches (via rasterization task) using pre-trained latent feature. Red denotes false positive cases.

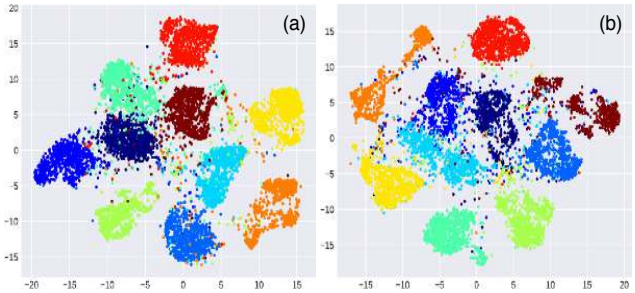

Figure 6. T-SNE Plots on features extracted by our self-supervised method (a) vectorization (sketch images) (b) rasterization (sketch vectors) for 10 QuickDraw classes.

Table 7. Cross-category recognition accuracy on QuickDraw.

	Image Space		Vector Space	
	Top-1	Top-5	Top-1	Top-5
MoCo [23]	53.4%	77.6%	–	–
SimCLR [11]	53.6%	77.6%	–	–
CPC [44]	46.8%	71.3%	48.1%	73.3%
Ours	65.1%	85.6%	58.4%	81.2%

Berlin dataset, followed by linear evaluation. Compared within dataset training accuracy of 70.6% (55.6%), we obtain a cross-dataset accuracy (Table 8) of 58.9% (36.9%) on TU-Berlin sketch-images (sketch-vectors) without much significant drop in accuracy, thus signifying the potential of our self-supervised method for sketch data.

Table 8. Cross-dataset (QuickDraw $\mapsto$ Tu-Berlin) recognition accuracy: Model pre-trained on QuickDraw is used to extract fixed latent feature on TU-Berlin, followed by linear model evaluation.

	Image Space		Vector Space	
	Top-1	Top-5	Top-1	Top-5
MoCo [23]	47.5%	62.1%	–	–
SimCLR [11]	47.2%	62.0%	–	–
CPC [44]	41.4%	60.8%	27.7%	50.9%
Ours	58.9%	80.5%	36.9%	61.7%

Cross-Task Generalisation: Both sketch and handwriting are hand-drawn data having similarity in terms of how they are recorded, and represented in image and vector space. Thus, we explore whether a model trained on handwritten data using self-supervised task can generalise over sketches, and vice versa. The pooling stride is adjusted so that, using sketch convolutional encoder, we can get sequential feature, and handwriting convolutional encoder can give feature vector representation on sketch images using global pooling. Following this protocol, we obtain (Table 9) a reasonable cross-task top-1 accuracy of 37.6% and 33.7% on sketch images and vectors on QuickDraw. Conversely, we get no-lexicon WRA of 28.4% and 26.3% on handwritten offline word images and online word vectors, respectively.

Table 9. Cross-task (Sketch $\leftrightarrow$ Handwriting) generalisation results on extracted fixed latent feature . Lexicon: (L), No-Lexicon: (NL)

	Sketch (QuickDraw)				Handwriting (IAM)			
	Image		Vector		Image		Vector	
	Top-1	Top-5	Top-1	Top-5	L	NL	L	NL
Random	14.6%	25.7%	11.8%	22.9%	9.8%	6.1%	7.1%	4.5%
CPC [44]	19.7%	37.8%	17.6%	36.9%	19.5%	12.5%	15.7%	9.7%
Ours	37.6%	58.4%	33.7%	55.8%	33.8%	28.4%	31.6%	26.3%

Further Analysis: (i) We have also compared with other backbone CNN network, e.g. AlexNet, where we obtain Top-1 accuracy of 63.3% compared to 71.9% on using ResNet50. This confirms suitability of our design across different backbone architecture. (ii) In our implementation, we perform only basic horizontal flipping and random cropping for augmentation. We also experiment with multiple augmentation strategies mentioned in [11], but notice no significant changes. (iii) Furthermore, following the recent works [62, 12] that jointly exploits raster sketch-image and temporal sketch-vector for sketch representation, we simply concatenate extracted latent feature of sketch-image and sketch-vector, and evaluate through linear classifier. This joint feature improves the top-1 accuracy to 72.8% compared to 71.9% which uses raster image only.

5. Conclusion

We have introduced a self-supervision method based on cross-modal rasterization/vectorization that is effective in representation learning for sketch and handwritten data. Uniquely our setup provides powerful representations for both vector and raster format inputs downstream. Results on sketch recognition, sketch retrieval, and handwriting recognition show that our pre-trained representation approaches the performance of supervised deep learning in the full data regime, and surpasses it in the low data regime. Thus Sketch2Vec provides a powerful tool to scale and accelerate deep-learning-based freehand writing analysis going forward.

References

[1] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Word spotting and recognition with embedded attributes. *IEEE T-PAMI*, 2014. 2

[2] Relja Arandjelovic and Andrew Zisserman. Look, listen and learn. In *ICCV*, 2017. 3

[3] Yoshua Bengio, Yann LeCun, Craig Nohl, and Chris Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 1995. 2

[4] Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Aneeshan Sain, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. More photos are all you need: Semi-supervised learning for fine-grained sketch based image retrieval. In *CVPR*, 2021. 2

[5] Ayan Kumar Bhunia, Abhirup Das, Ankan Kumar Bhunia, Perla Sai Raj Kishore, and Partha Pratim Roy. Handwriting recognition in low-resource scripts using adversarial learning. In *CVPR*, 2019. 2, 4, 5

[6] Ayan Kumar Bhunia, Ayan Das, Umar Riaz Muhammad, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, Yulia Gryaditskaya, and Yi-Zhe Song. Pixelor: A competitive sketching ai agent. so you think you can beat me? In *Siggraph Asia*, 2020. 2

[7] Ayan Kumar Bhunia, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Sketch less for more: On-the-fly fine-grained sketch based image retrieval. In *CVPR*, 2020. 2, 4

[8] Théodore Bluche. Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. In *NeurIPS*, 2016. 2

[9] Victor Carbune, Pedro Gonnet, Thomas Deselaers, Henry A Rowley, Alexander Daryin, Marcos Calvo, Li-Lun Wang, Daniel Keysers, Sandro Feuz, and Philippe Gervais. Fast multi-language lstm-based online handwriting recognition. *IJDAR*, 2020. 2, 5

[10] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 1, 3, 5, 6

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 2, 3, 5, 6, 8

[12] John Collomosse, Tu Bui, and Hailin Jin. Livesketch: Query perturbations for guided sketch-based visual search. In *CVPR*, 2019. 4, 8

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019. 4

[14] Sounak Dey, Pau Riba, Anjan Dutta, Josep Llados, and Yi-Zhe Song. Doodle to search: Practical zero-shot sketch-based image retrieval. In *CVPR*, 2019. 2, 4, 7

[15] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015. 1, 3, 5, 6

[16] Anjan Dutta and Zeynep Akata. Semantically tied paired cycle consistency for zero-shot sketch-based image retrieval. In *CVPR*, 2019. 2

[17] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? *ACM TOG*, 2012. 4

[18] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018. 1, 2, 5, 6

[19] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019. 1

[20] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE-TPAMI*, 2008. 1, 2, 4

[21] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 2, 5, 6

[22] David Ha and Douglas Eck. A neural representation of sketch drawings. *ICLR*, 2017. 1, 2, 3, 4, 7

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 1, 2, 3, 5, 6, 8

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[25] Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. In *ICML*, 2019. 2, 3

[26] Jianying Hu, Michael K Brown, and William Turin. Hmm based online handwriting recognition. *IEEE-TPAMI*, 1996. 2

[27] Wei-Chih Hung, Varun Jampani, Sifei Liu, Pavlo Molchanov, Ming-Hsuan Yang, and Jan Kautz. Scops: Self-supervised co-part segmentation. In *CVPR*, 2019. 1

[28] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 4, 5

[29] L. Jing and Y. Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE TPAMI*, 2020. 2

[30] Daniel Keysers, Thomas Deselaers, Henry A Rowley, Li-Lun Wang, and Victor Carbune. Multi-language online handwriting recognition. *IEEE-TPAMI*, 2016. 2

[31] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 1, 2, 5, 6

[32] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Self-supervised learning of 3d human pose using multi-view geometry. In *CVPR*, 2019. 1

[33] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. Revisiting self-supervised visual representation learning. In *CVPR*, 2019. 3, 5

[34] Bruno Korbar, Du Tran, and Lorenzo Torresani. Cooperative learning of audio and video models from self-supervised synchronization. In *NeurIPS*, 2018. 2, 3

[35] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken,

Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *CVPR*, 2017. 2, 3

[36] Junnan Li, Pan Zhou, Caiming Xiong, Richard Socher, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020. 3

[37] Hangyu Lin, Yanwei Fu, Yu-Gang Jiang, and Xiangyang Xue. Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. In *CVPR*, 2020. 2, 4, 5, 6

[38] Fang Liu, Changqing Zou, Xiaoming Deng, Ran Zuo, Yu-Kun Lai, Cuixia Ma, Yong-Jin Liu, and Hongan Wang. Scenesketcher: Fine-grained image retrieval with scene sketches. In *ECCV*, 2020. 2

[39] Canjie Luo, Yuanzhi Zhu, Lianwen Jin, and Yongpan Wang. Learn to augment: Joint data augmentation and network optimization for text recognition. In *CVPR*, 2020. 2

[40] U-V Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *IJDAR*, 2002. 4, 5

[41] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, 2016. 4

[42] Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *ECCV*, 2016. 3

[43] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016. 1, 3, 5, 6

[44] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 4, 5, 6, 8

[45] Kaiyue Pang, Ke Li, Yongxin Yang, Honggang Zhang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Generalising fine-grained sketch-based image retrieval. In *CVPR*, 2019. 2

[46] Kaiyue Pang, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. Solving mixed-modal jigsaw puzzle for fine-grained sketch-based image retrieval. In *CVPR*, 2020. 2, 3

[47] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Autodiff Workshop*, 2017. 5

[48] Massimiliano Patacchiola and Amos Storkey. Self-supervised relational reasoning for representation learning. In *NeurIPS*, 2020. 3

[49] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 3

[50] Arik Poznanski and Lior Wolf. Cnn-n-gram for handwriting word recognition. In *CVPR*, 2016. 1, 2

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 1

[52] Aneeshan Sain, Ayan Kumar Bhunia, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. Cross-modal hierarchical modelling forfine-grained sketch based image retrieval. In *BMVC*, 2020. 1, 2

[53] Aneeshan Sain, Ayan Kumar Bhunia, Yongxin Yang, Tao Xiang, and Yi-Zhe Song. Stylemeup: Towards style-agnostic sketch-based image retrieval. In *CVPR*, 2021. 2

[54] Leo Sampaio Ferraz Ribeiro, Tu Bui, John Collomosse, and Moacir Ponti. Sketchformer: Transformer-based representation for sketched structure. In *CVPR*, 2020. 2

[55] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: learning to retrieve badly drawn bunnies. *TOG*, 2016. 1

[56] Baoguang Shi, Mingkun Yang, Xinggang Wang, Pengyuan Lyu, Cong Yao, and Xiang Bai. Aster: An attentional scene text recognizer with flexible rectification. *IEEE T-PAMI*, 2018. 4, 5, 6

[57] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Learning to sketch with shortcut cycle consistency. In *CVPR*, 2018. 1, 2, 7

[58] Jifei Song, Qian Yu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Deep spatial-semantic attention for fine-grained sketch-based image retrieval. In *ICCV*, 2017. 2

[59] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019. 2, 3, 5, 6

[60] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 5

[61] Tianwei Wang, Yuanzhi Zhu, Lianwen Jin, Canjie Luo, Xiaoxue Chen, Yaqiang Wu, Qianying Wang, and Mingxiang Cai. Decoupled attention network for text recognition. In *AAAI*, 2020. 2

[62] Peng Xu, Yongye Huang, Tongtong Yuan, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, Zhanyu Ma, and Jun Guo. Sketchmate: Deep hashing for million-scale human sketch retrieval. In *CVPR*, 2018. 1, 2, 4, 8

[63] Peng Xu, Chaitanya K Joshi, and Xavier Bresson. Multi-graph transformer for free-hand sketch recognition. *arXiv preprint arXiv:1912.11258*, 2019. 2

[64] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. Sketch me that shoe. In *CVPR*, 2016. 2

[65] Qian Yu, Yongxin Yang, Feng Liu, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Sketch-a-net: A deep neural network that beats humans. *IJCV*, 2017. 1, 4

[66] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016. 1, 2, 3

[67] Yaping Zhang, Shuai Nie, Wenju Liu, Xing Xu, Dongxiang Zhang, and Heng Tao Shen. Sequence-to-sequence domain adaptation network for robust text image recognition. In *CVPR*, 2019. 4