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

Ayan Kumar Bhunia\textsuperscript{1} Pinaki Nath Chowdhury\textsuperscript{1,3} Yongxin Yang\textsuperscript{1,3} Timothy M. Hospedales\textsuperscript{1,2} Tao Xiang\textsuperscript{1,3} Yi-Zhe Song\textsuperscript{1,3}

\textsuperscript{1}SketchX, CVSSP, University of Surrey, United Kingdom \textsuperscript{2}University of Edinburgh, United Kingdom. \textsuperscript{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


doi: 10.1007/s11263-021-10061-1

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 \cite{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 \cite{51} that are both costly and time-consuming to collect. In order to alleviate the data annotation bottleneck, many unsupervised methods \cite{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 \cite{66}, jigsaw solving \cite{43}, image rotation prediction \cite{18}, etc. The motivation is that a net-work 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 \cite{32}, co-part segmentation \cite{27}, and depth estimation \cite{19}.

In this paper, we propose a self-supervised method for a class of visual data that is distinctly different than photos: sketches \cite{65, 55} and handwriting \cite{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 \cite{57, 22}; while sketch in particular poses the challenge of variable levels of detail \cite{52} depicted. Both sketch \cite{62} and handwriting \cite{20} can be represented in rasterized pixel space, or as a temporal point sequence \cite{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. Connectorist 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 [43, 46], in-painting [49], relative patch location prediction orization [66], super-resolution [35], solving jigsaw puzzles addressing multi-modal data. For instance, pre-text tasks, our work is more similar to the few approaches

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 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.

3.1. 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 modeling. 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 [32] of three pen-state situations: pen touching the paper, pen being lifted and end of drawing. Thus, the size of vector representation is $V = \mathbb{R}^{T \times 5}$, 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 $l_f$ of size $R^d$, 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: $h_0 = W_y l_f + b_y$. The hidden state $h_t$ of decoder RNN is updated as follows: $h_t = RNN(h_{t−1}; [l_t, P_{t−1}])$, 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:

$$L_{I\rightarrow V} = \frac{1}{T} \sum_{t=1}^{T} \left( \| \hat{x}_t - x_t \|_2 + \| \hat{y}_t - y_t \|_2 \right)$$

$$- \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{3} \hat{q}_t^i \log \left( \frac{\exp(q_t^i)}{\sum_{j=1}^{3} \exp(q_t^j)} \right)$$
Latent Initialize self-supervised pre-training. For both classification and retrieval, the quality of learned latent representations obtained by our sketch-retrieval [62, 37] as downstream tasks to evaluate mean-square error as the training objective:

\[ L_{V \rightarrow I} = -E_{I,V \sim (I,V)} \| I - D_{I}(E_{V}(V)) \|_2 \]  

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 tasks 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.
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 multimodal 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 counterpart 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
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.

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.
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 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.

<table>
<thead>
<tr>
<th>Image Space</th>
<th>Vector Space</th>
<th>Top-1</th>
<th>Top-5</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo [23]</td>
<td>–</td>
<td>53.4%</td>
<td>77.6%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SimCLR [11]</td>
<td>–</td>
<td>53.6%</td>
<td>77.6%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CPC [44]</td>
<td>–</td>
<td>46.8%</td>
<td>71.3%</td>
<td>48.1%</td>
<td>73.3%</td>
</tr>
<tr>
<td>Ours</td>
<td>–</td>
<td>65.1%</td>
<td>85.6%</td>
<td>58.4%</td>
<td>81.2%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Image Space</th>
<th>Vector Space</th>
<th>Top-1</th>
<th>Top-5</th>
<th>Top-1</th>
<th>Top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>MoCo [23]</td>
<td>–</td>
<td>47.5%</td>
<td>62.1%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>SimCLR [11]</td>
<td>–</td>
<td>47.2%</td>
<td>62.0%</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CPC [44]</td>
<td>–</td>
<td>41.4%</td>
<td>60.8%</td>
<td>27.7%</td>
<td>50.9%</td>
</tr>
<tr>
<td>Ours</td>
<td>–</td>
<td>58.9%</td>
<td>80.5%</td>
<td>36.9%</td>
<td>61.7%</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Sketch (QuickDraw)</th>
<th>Handwriting (IAM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Space</td>
<td>Vector Space</td>
</tr>
<tr>
<td>Top-1</td>
<td>Top-5</td>
</tr>
<tr>
<td>MoCo [23]</td>
<td>–</td>
</tr>
<tr>
<td>SimCLR [11]</td>
<td>–</td>
</tr>
<tr>
<td>CPC [44]</td>
<td>–</td>
</tr>
<tr>
<td>Ours</td>
<td>–</td>
</tr>
</tbody>
</table>

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.


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


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

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

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

Kaiyue Pang, Yongxian 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


Aneeshan Sain, Ayan Kumar Bhunia, Yongxian Yang, Tao Xiang, and Yi-Zhe Song. Cross-modal hierarchical modelling for fine-grained sketch based image retrieval. In BMVC, 2020. 1, 2

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


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

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

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

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


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


Qian Yu, Yongxian 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

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

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