Sketchformer: Transformer-based Representation for Sketched Structure

Leo Sampaio Ferraz Ribeiro[*1], Tu Bui[*2], John Collomosse[2,3], and Moacir Ponti[1]

[1]ICMC, Universidade de São Paulo – São Carlos/SP, Brazil
leo.sampaio.ferraz.ribeiro@gmail.com, ponti@usp.br
[2]CVSSP, University of Surrey – Guildford, Surrey, UK
{t.bui,j.collomosse}@surrey.ac.uk
[3]Adobe Research, Creative Intelligence Lab – San Jose, CA, USA

Abstract

Sketchformer is a novel transformer-based representation for encoding free-hand sketches input in a vector form, i.e. *as a sequence of strokes. Sketchformer effectively addresses multiple tasks: sketch classification, sketch based image retrieval (SBIR), and the reconstruction and interpolation of sketches. We report several variants exploring continuous and tokenized input representations, and contrast their performance. Our learned embedding, driven by a dictionary learning tokenization scheme, yields state of the art performance in classification and image retrieval tasks, when compared against baseline representations driven by LSTM sequence to sequence architectures: SketchRNN and derivatives. We show that sketch reconstruction and interpolation are improved significantly by the Sketchformer embedding for complex sketches with longer stroke sequences.*

1. Introduction

Sketch representation and interpretation remains an open challenge, particularly for complex and casually constructed drawings. Yet, the ability to classify, search, and manipulate sketched content remains attractive as gesture and touch interfaces reach ubiquity. Advances in recurrent network architectures within language processing have recently inspired sequence modeling approaches to sketch (*e.g.* SketchRNN [1]) that encode sketch as a variable length sequence of strokes, rather than in a rasterized or 'pixel' form. In particular, long-short term memory (LSTM) networks have shown significant promise in learning search embeddings [2, 3] due to their ability to model higher-level structure and temporal order versus convolutional neural networks (CNNs) on rasterized sketches [4, 5, 6, 7]. Yet, the limited temporal extent of LSTM restricts the structural complexity of sketches that may be accommodated in sequence embeddings. In language modeling domain, this shortcoming has been addressed through the emergence of Transformer networks [8, 9, 10] in which slot masking enhances the ability to learn complex structures that are represented by longer sequences.

This paper proposes Sketchformer, the first Transformer based network for learning a deep representation for free-hand sketches. We build on the language modeling Transformer architecture of Vaaswani *et al*. [10] to develop several variants of Sketchformer that process sketch sequences in continuous and tokenized forms. We evaluate the efficacy of each learned sketch embedding for common sketch interpretation tasks. We make three core technical contributions:
1) Sketch Classification. We show that Sketchformer driven by a dictionary learning tokenization scheme outperforms state of the art sequence embeddings for sketched object recognition over QuickDraw! [11]; the largest and most diverse public corpus of sketched objects.
2) Generative Sketch Model. We show that for more complex, detailed sketches comprising lengthy stroke sequences, Sketchformer improves generative modeling of sketch – demonstrated by higher fidelity reconstruction of sketches from the learned embedding. We also show that for sketches of all complexities, interpolation in the Sketchformer embedding is stable, generating more plausible intermediate sketches for both inter- and intra-class blends.
3) Sketch based Image Retrieval (SBIR) We show that Sketchformer can be unified with raster embedding to produce a search embedding for SBIR (after [3] for LSTM) to deliver improved prevision over a large photo corpus (Stock10M).

These enhancements to sketched object understanding, generative modeling and matching demonstrated for a diverse and complex sketch dataset suggest Transformer as a promising direction for stroke sequence modeling.

2. Related Work

Representation learning for sketch has received extensive attention within the domain of visual search. Classical

[*]These authors contributed equally to this work

sketch based image retrieval (SBIR) techniques explored region [12], graph [13], edge-let [14], and sparse gradient features [15, 16] building upon the success of dictionary learning based models (e.g. bag of words) [17, 18, 19]. With the advent of deep learning, convolutional neural networks (CNNs) were rapidly adopted to learn search embedding [20]. Triplet loss models are commonly used for visual search in the photographic domain [21, 22, 23], and have been extended to SBIR. Sangkloy *et al*. [7] used a three-branch CNN with triplet loss to learn a general cross-domain embedding for SBIR. Fine-grained (within-class) SBIR was similarly explored by Yu *et al*. [24]. Qi *et al*. [5] instead use contrastive loss to learn correspondence between sketches and pre-extracted edge maps. Bui *et al*. [25, 4] perform cross-category retrieval using a triplet model and combined their technique with a learned model of visual aesthetics [26] to constrain SBIR using aesthetic cues in [6]. A quadruplet loss was proposed by [27] for fine-grained SBIR. The generalization of sketch embeddings beyond training classes have also been studied [28, 29], and parameterized for zero-shot learning [30]. Such concepts were later applied in sketch-based shape retrieval tasks [31]. Variants of CycleGAN [32] have also shown to be useful as generative models for sketch [33]. Sketch-A-Net was a seminal work for sketch classification that employed a CNN with large convolutional kernels to accommodate the sparsity of stroke pixels [24]. Recognition of partial sketches has also been explored by [34]. Wang *et al*. [35] proposed sketch classification by sampling unordered points of a sketch image to learning a canonical order.

All the above works operate over rasterized sketches *e.g*. converting the captured vector representation of sketch (as a sequence of strokes) to pixel form, discarding temporal order of strokes, and requiring the network to recover higher level spatial structure. Recent SBIR work has begun to directly input a vector (stroke sequence) representations for sketches [36], notably SketchRNN; an LSTM based sequence to sequence (seq2seq) variational auto-proposed by Eck *et al*. [1], trained on the largest public sketch corpus 'QuickDraw!' [11]. SketchRNN embedding was incorporated in a triplet network by Xu *et al*. [2] to search for sketches using sketches. A variation using cascaded attention networks was proposed by [37] to improve vector sketch classification over Sketch-A-Net. Later, LiveSketch [3] extended SketchRNN to a triplet network to perform SBIR over tens of millions of images, harnessing the sketch embedding to suggest query improvements and guide the user via relevance feedback. The limited temporal scope of LSTM based seq2seq models can prevent such representations modeling long, complex sketches, a problem mitigated by our Transformer based model which builds upon the success shown by such architectures for language modeling [10, 9, 8]. Transformers encode long term temporal dependencies by modeling direct connections between data units. The temporal range of such dependencies was increased via the Transformer-XL [9] and BERT [8], which recently set new state-of-the-art performance on sentence classification and sentence-pair regression tasks using a cross-encoder. Recent work explores transformer beyond sequence modeling to 2D images [38]. Our work is first to apply these insights to the problem of sketch modeling, incorporating the Transformer architecture of Vaswani *et al*. [10] to deliver a multi-purpose embedding that exceeds the state of the art for several common sketch representation tasks.

3. Sketch Representation

We propose Sketchformer; a multi-purpose sketch representation from stroke sequence input. In this section we discuss the pre-processing steps, the adaptions made to the core architecture proposed by Vaswani *et al*. [10] and the three application tasks.

3.1. Pre-processing and Tokenization

Following Eck *et al*. [1] we simplify all sketches using the RDP algorithm [39] and normalize stroke length. Sketches for all our experiments are drawn from QuickDraw50M [11] (see Sec. 4; for dataset partitions).

1) Continuous. Quickdraw50M sketches are released in the 'stroke-3' format where each point $(\delta x, \delta y, p)$ stores its relative position to the previous point together with its binary pen state. To also include the 'end of sketch' state, the stroke-5 format is often employed: $(\delta x, \delta y, p_1, p_2, p_3)$, where the the pen states p_1 - *draw*, p_2 - *lift* and p_3 - *end* are mutually exclusive [1]. Our experiments with continuous sketch modeling use the 'stroke-5' format.

2) Dictionary learning. We build a dictionary of K code words ($K = 1000$) to model the relative pen motion *i.e*. $(\delta x, \delta y)$. We randomly sample 100k sketched pen movements in the training set for clustering via K-means. We allocate 20% of sketch points for sampling inter-stroke transition, *i.e*. relative transition when the pen is lifted, to balance with the more common within-stroke transitions. Each transition point $(\delta x, \delta y)$ is then assigned to the nearest code word, resulting in a sequence of discrete tokens. We also include 4 special tokens; a Start of Sketch (SOS) token at the beginning of every sketch, an End of Sketch (EOS) token at the end, a Stroke End Point (SEP) token to be inserted between strokes (indicate pen lifting) and a padding (PAD) token to pad the sketch to a fixed length.

3) Spatial Grid. The sketch canvas is first quantized into $n \times n$ ($n = 100$) square cells, each cell is represented by a token in our dictionary. Given the absolute (x, y) sketch points, we determine which cell contains this point and assign the cell's token to the point. The same four special tokens above are used to complete the sketch sequence.

Fig. 2 visualizes sketch reconstruction under the tokenized methods to explore sensitivity to the quantization parameters. Compared with the stroke-5 format (continuous) the tokenization methods are more compact. Dictionary learned tokenization (Tok-Dict) can have a small dictionary size and is invariant to translation since it is derived from stroke-3. On the other hand quantization error could accumulate over longer sketches if dictionary size is too low,

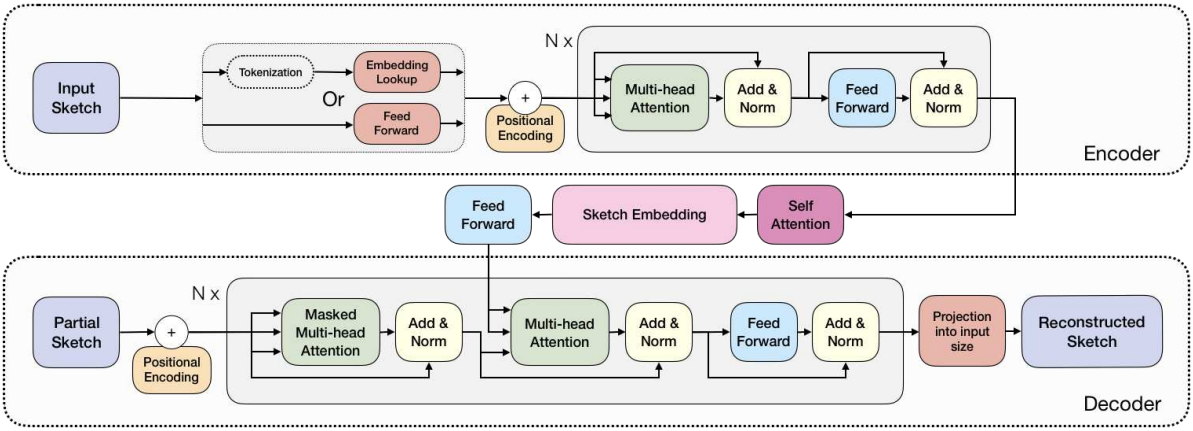

Figure 1. Schematic of the Transformer architecture used for Sketchformer, which utilizes the original architecture of Vaswani *et al*. [10] but modifies it with an alternate mechanism for formulating the bottleneck (sketch embedding) layer using a self-attention block, as well as configuration changes e.g. MHA head count (see Sec. 3.2).

shifting the position of strokes closer to the sequence's end. The spatial grid based tokenization method (Tok-Grid), on the other hand, does not accumulate error but is sensitive to translation and yields a larger vocabulary (n^2).

3.2. Transformer Architecture for Sketch

Sketchformer uses the Transformer network of Vaswani *et al*. [10]. We add stages (*e.g*. self-attention and modified bottleneck) and adapt parameters in their design to learn a multi-purpose representation for stroke sequences, rather than language. A transformer network consists of an encoder and decoder blocks, each comprising several layers of multihead attention followed by a feed forward network. Fig. 1 illustrates the architecture with dotted lines indicating re-use of architecture stages from [10]. In Fig. 4 we show how our learned embedding is used across multiple applications. Compared to [10] we use 4 MHA blocks versus 6 and a feed-forward dimension of 512 instead of 2048. Unlike traditional sequence modeling methods (RNN/LSTM) which learns the temporal order of current time steps from previous steps (or future steps in bidirectional encoding), the attention mechanism in transformers allows the network to decide which time steps to focus on to improve the task at hand. Each multihead attention (MHA) layer is formulated as such:

$$SHA(k,q,v) = softmax(\alpha q k^T)v \quad (1)$$

$$MHA(k,q,v) = [SHA_0(kW_0^k, qW_0^q, vW_0^v), ... \quad (2)$$

$$SHA_m(kW_m^k, qW_m^q, vW_m^v)]W^0 \quad (3)$$

where k, q and v are respective *Key*, *Query* and Value inputs to the single head attention (SHA) module. This module computes the similarity between pairs of *Query* and *Key* features, normalizes those scores and finally uses them as a projection matrix for the *Value* features. The multihead attention (MHA) module concatenates the output of multiple single heads and projects the result to a lower dimension. α is a scaling constant and $W_{(.)}^{(.)}$ are learnable weight matrices.

The MHA output is fed to a positional feed forward network (FFN), which consists of two fully connected layers with ReLU activation. The MHA-FFN ($F(.)$) blocks are the basis of the encoder side of our network ($\mathcal{E}(.)$):

$$FFN(x) = max(0, xW_1^f + b_1^f)W_2^f + b_2^f \quad (4)$$

$$F(x) = \overline{FFN}(\overline{MHA}(x,x,x)) \quad (5)$$

$$\mathcal{E}(x) = F_N(..(F_1(x))) \quad (6)$$

where $\overline{X}$ indicates layer normalization over X and N is number of the MHA-FFN units $F(.)$.

The decoder takes as inputs the encoder output and target sequence in an auto-regressive fashion. In our case we are learning an transformer autoencoder so the target sequence is also the input sequence shifted forward by 1:

$$G(h,x) = \overline{FFN}(\overline{MHA}_2(h,h,\overline{MHA}_1(x,x,x))) \quad (7)$$

$$\mathcal{D}(h,x*) = G_N(h, G_{N-1}(h, ...G_1(h,x*))) \quad (8)$$

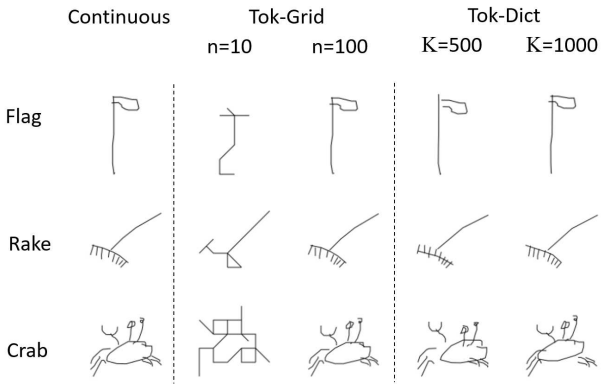

Figure 2. Visualizing the impact of quantization on the reconstruction of short, median and long sequence length sketches. Grid sizes of $n = [10, 100]$ (Tok-Grid) and dictionary sizes of $K = [500, 1000]$ (Tok-Dict). Sketches are generated from the tokenized sketch representations, independent of transformer.

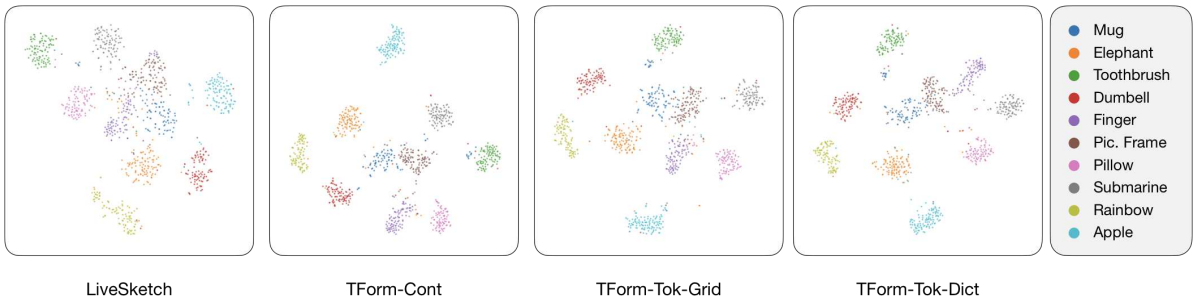

Figure 3. t-SNE visualization of the learned embedding space from Sketchformer's three variants, compared to LiveSketch (left) and computed over the QD-862k test set; 10 categories and 1000 samples were randomly selected.

where h is the encoder output, $x*$ is the shifted auto-regressive version of input sequence x.

The conventional transformer is designed for language translation and thus does not provide a feature embedding as required in Sketchformer (output of $\mathcal{E}$ is also a sequence of vectors of the same length as x). To learn a compact representation for sketch we propose to apply self-attention on the encoder output, inspired by [40]:

$$s = softmax(tanh(hK^T + b)v) \tag{9}$$

$$z = \sum_i s_i h_i \tag{10}$$

which is similar to SHA however the Key matrix K, Value vector v and bias b are now trainable parameters. This self-attention layer learns a weight vector s describing the importance of each time step in sequence h, which is then accumulated to derive the compact embedding z. On the decoder side, z is passed through a FFN to resume the original shape of h. These are the key novel modifications to the original Transformer architecture of Vaswani *et al*. (beyond above-mentioned parameter changes).

We also had to change how masking worked on the decoder. The Transformer uses a padding mask to stop attention blocks from giving weight to out-of-sequence points. Since we want a meaningful embedding for reconstruction and interpolation, we removed this mask from the decoder, forcing our transformer to learn reconstruction without previously knowing the sequence length and using only the embedding representation.

3.2.1 Training Losses

We employ two losses in training Sketchformer. A classification (softmax) loss is connected to the sketch embedding z to preserve semantic information while a reconstruction loss ensures the decoder can reconstruct the input sequence from its embedding. If the input sequence is continuous (*i.e.* stroke-5) the reconstruction loss consists of a L^2 loss term modeling relative transitions $(\delta x, \delta y)$ and a 3-way classification term modeling the pen states. Otherwise the reconstruction loss uses softmax to regularize a dictionary of sketch tokens as per a language model. We found these losses simple yet effective in learning a robust sketch embedding. Fig. 3 visualizes the learned embedding for each of the three pre-processing variants, alongside that of a state of the art sketch encoding model using stroke sequences [3].

3.3. Cross-modal Search Embedding

To use our learned embedding for SBIR, we follow the joint embedding approach first presented in [3] and train an auxiliary network that unifies the vector (sketch) and raster (image corpus) representations into a common subspace.

This auxiliary network is composed of four fully connected layers (see Fig. 4) with ReLU activations. These are trained within a triplet framework and have input from three pre-trained branches: an anchor branch that models vector representations (our Sketchformer), plus positive and negative branches extracting representations from raster space.

The first two fully connected layers are domain-specific and we call each set $F_V(.), F_R(.)$, referring to vector-specific and raster-specific. The final two layers are shared between domains; we refer to this set as $F_S(.)$. Thus the end-to-end mapping from vector sketch and raster sketch/image to the joint embedding is:

$$u_v = F_S(F_V(\mathcal{E}(x_v))) \tag{11}$$

$$u_r = F_S(F_R(\mathcal{P}(x_r))) \tag{12}$$

where x_v and x_r are the input vector sketches and raster images respectively, and u_v and u_r their corresponding representations in the common embedding. $\mathcal{E}(.)$ is the network that models vector representations and $\mathcal{P}(.)$ is the one for raster images. In the original LiveSketch [3], $\mathcal{E}(.)$ is a SketchRNN [1]-based model, while we employ our multi-task Sketchformer encoder instead. For $\mathcal{P}(.)$ we use the same off-the-shelf GoogLeNet-based network, pre-trained on a joint embedding task (from [28]).

The training is performed using triplet loss regularized with the help of a classification task. Training requires an aligned sketch and image dataset *i.e.* a sketch set and image set that share the same category list. This is not the case for Quickdraw, which is a sketch-only dataset without a corresponding image set. Again following [3], we use the raster sketch as a medium to bridge vector sketch with raster image. The off-the-shelf $\mathcal{P}(.)$ (from [28]) was trained

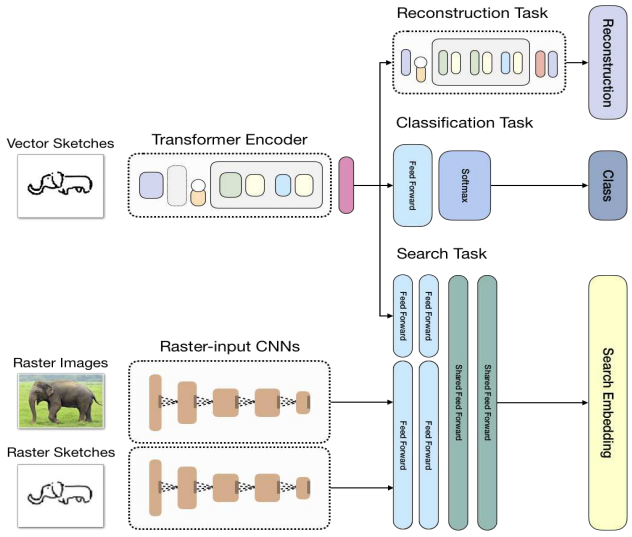

Figure 4. Schematic showing how the learned sketch embedding is leveraged for sketch synthesis (reconstruction/interpolation), classification and cross-modal retrieval experiments (see encoder/embedding inset, refer to Fig. 1 for detail). Classification appends fully-connected (fc) and softmax layers to the embedding space. Retrieval tasks require unification with a raster (CNN) embedding for images [28] via several fc layers trained via triplet loss.

to produce a joint embedding model unifying raster sketch and raster image; This allowed the authors train the F_R, F_V and F_S sets using vector and raster versions of sketch only. By following the same procedure, we eliminate the need of having an aligned image set for Quickdraw as our network never sees an image feature during training.

The training is implemented in two phases. At phase one, the anchor and positive samples are vector and raster forms of random sketches in the same category while raster input of the negative branch is sampled from a different category. At phase two, we sample hard negatives from the same category with the anchor vector sketch and choose the raster form of the exact instance of the anchor sketch for the positive branch. The triplet loss maintains a margin between the anchor-positive and anchor-negative distances:

$$\mathcal{L}_T(x, x_+, x_-) = max(0, |F_S(F_V(\mathcal{E}(x))) - F_S(F_R(\mathcal{P}(x_+)))| - ||F_S(F_V(\mathcal{E}(x))) - F_S(F_R(\mathcal{P}(x_-)))|| + m) \quad (13)$$

and margin $m = 0.2$ in phase one, $m = 0.05$ in phase two.

4. Experiments and Discussion

We evaluate the performance of the proposed transformer embeddings for three common tasks; sketch classification, sketch reconstruction and interpolation, and sketch based image retrieval (SBIR). We compare against two baseline sketch embeddings for encoding stroke sequences; SketchRNN [1] (also used for search in [2]) and LiveSketch [3]. We evaluate using sketches from QuickDraw50M [11], and a large corpus of photos (Stock10M).

QuickDraw50M [11] comprises over 50M sketches of 345 object categories, crowd-sourced within a gamified context that encouraged casual sketches drawn at speed. Sketches are often messy and complex in their structure, consistent with tasks such as SBIR. Quickdraw50M captures sketches as stroke sequences, in contrast to earlier raster-based and less category-diverse datasets such as TU-Berlin/Sketchy. We sample 2.5M sketches randomly with even class distribution from the public Quickdraw50M training partition to create training set (QD-2.5M) and use the public test partition of QuickDraw50M (QD-862k) comprising 2.5k $\times 345 = 862k$ sketches to evaluate our trained models. For SBIR and interpolation experiments we sort QD-862k by sequence length, and sample three datasets (QD345-S, QD345-M, QD345-L) at centiles 10, 50 and 90 respectively to create a set of short, medium and long stroke sequences. Each of these three datasets samples one sketch per class at random from the centile yielding three evaluation sets of 345 sketches. We sampled an additional query set QD345-Q for use in sketch search experiments, using the same 345 sketches as LiveSketch [3]. The median stroke lengths of QD345-S, QD345-M, QD345-L are 30, 47 and 75 strokes respectively (after simplification via RDP [39]).

Stock67M is a diverse, unannotated corpus of photos used in prior SBIR work [3] to evaluate large-scale SBIR retrieval performance. We sample 10M of these images at random for our search corpus (Stock10M).

4.1. Evaluating Sketch Classification

We evaluate the class discrimination of the proposed sketch embedding via attaching dense and softmax layers to the transformer encoder stage, and training a 345-way classifier on QD2.5M. Table 1 reports the classification performance over QD-862k for each of the three proposed transformer embeddings, alongside two LSTM baselines – the SketchRNN [1] and LiveSketch [3] variational autoencoder networks. Whilst all transformers outperform the baseline, the tokenized variant of the transformer based on dictionary learning (TForm-Tok-Dict) yields highest accuracy. We explore this further by shuffling the order of the sketch strokes retraining the transformer models from scratch. We were

Method		mAP%
Baseline	LiveSketch [3]	72.93
	SketchRNN [1]	67.69
Shuffled	TForm-Cont	76.95
	TForm-Tok-Grid	76.22
	TForm-Tok-Dict	76.66
Proposed	TForm-Cont	77.68
	TForm-Tok-Grid	77.36
	TForm-Tok-Dict	**78.34**

Table 1. Sketch classification results over QuickDraw! [11] for three variants of the proposed transformer embedding, contrasting each to models learned from randomly permuted stroke order. Comparing to two recent LSTM based approaches for sketch sequence encoding [3, 1].

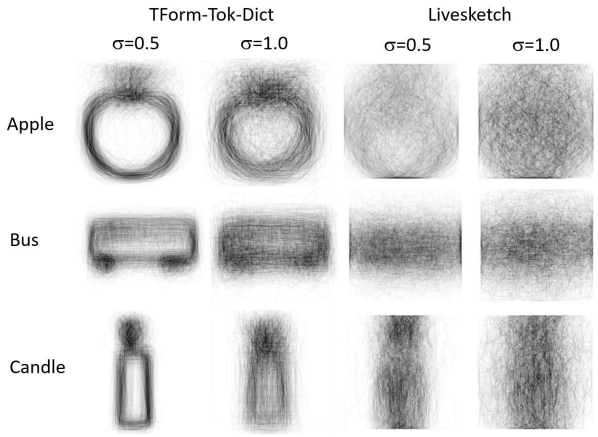

Figure 5. Visualization of sketches reconstructed from mean embedding for 3 object categories. We add Gaussian noise with standard deviation $\sigma = 0.5$ and $\sigma = 1.0$ to the mean embedding of three example categories on the Quickdraw test set. The reconstructed sketches of Tform-Tok-Dict retain salient features even with high noise perturbation.

Method		Short	Mid	Long
Baseline	LiveSketch [3]	**62.1**	**59.2**	27.8
	SketchRNN [1]	4.05	3.70	1.38
Proposed	TForm-Cont	0.00	0.00	0.00
	TForm-Tok-Grid	6.75	6.10	5.56
	TForm-Tok-Dict	24.3	28.4	**51.4**
	Uncertain	2.70	2.47	13.9

Table 2. User study quantifying accuracy of sketch reconstruction. Preference is expressed by 5 independent workers, and results with $> 50\%$ agreement are included. Experiment repeated for short, medium and longer stroke sequences. For longer sketches, the proposed transformer method TForm-Tok-Dict is preferred.

surprised to see comparable performance, perhaps due to the transformer's larger information scope rather than solely temporal information.

4.2. Reconstruction and Interpolation

We explore the generative power of the proposed embedding by measuring the degree of fidelity with which: 1) encoded sketches can be reconstructed via the decoder to resemble the input; 2) a pair of sketches may be interpolated within, and synthesized from, the embedding. The experiments are repeated for short (QD345-S), medium (QD345-M) and long (QD345-L) sketch complexities. We assess the fidelity of sketch reconstruction and the visual plausibility of interpolations via Amazon Mechanical Turk (MTurk). MTurk workers are presented with a set of reconstructions or interpolations and asked to make a 6-way preference choice; 5 methods and a 'cannot determine' option. Each task is presented to five unique workers, and we only include results for which there is $> 50\%$ (*i.e.* > 2 worker) consensus on the choice.

Reconstruction results are shown in Table 2 and favor the LiveSketch [3] embedding for short or medium length strokes, with the proposed tokenized transformer (TForm-Tok-Dict) producing better results for more complex sketches aided by the improved representational power of transformer for longer stroke sequences. Fig 6 provides representative visual examples for each sketch complexity.

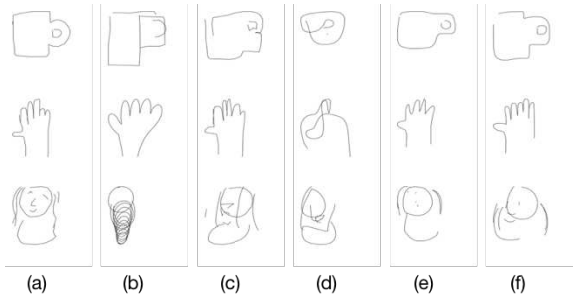

Figure 6. Representative sketch reconstructions from each of the five embeddings evaluated in Table 2. (a) Original, (b) SketchRNN, (c) LiveSketch, (d) TForm-Cont, (e) TForm-Tok-Grid and (f) TForm-Tok-Dict. The last row represents a hard-to-reconstruct sketch.

We explore interpolation in Table 3 blending between pairs of sketches within (intra-) class and between (inter-) class. In all cases we encode sketches separately to the embedding, interpolate via slerp (after [1, 3] in which slerp was shown to offer best performance), and decode the interpolated point to generate the output sketch. Fig. 7 provides visual examples of inter- and intra- class interpolation for each method evaluated. In all cases the proposed tokenized transformer (TForm-Tok-Dict) outperforms other transformer variants and baselines, although the performance separation is narrower for shorter strokes echoing results of the reconstruction experiment. The stability of our representation is further demonstrated via local sampling within the embedding in Fig. 5.

4.3. Cross-modal Matching

We evaluate the performance of Sketchformer for sketch based retrieval of sketches (S-S) and images (S-I).

Sketch2Sketch (S-S) Matching. We quantify the accuracy of retrieving sketches in one modality (raster) given a sketched query in another (vector, *i.e.* stroke sequence) – and vice-versa. This evaluates the performance of Sketchformer in discriminating between sketched visual structures invariant to their input modality. Sketchformer is trained on QD-2.5M and we query the test corpus QD-826k using QD-345Q as the query set. We measure overall mean average precision (mAP) for both coarse grain (*i.e.* class-specific) and fine-grain (*i.e.* instance-level) similarity. As per [3] for the former we consider a retrieved record a match if it matches the sketched object class. For the latter, exactly the same single sketch must match (in its different modality). To run raster variants, a rasterized version of QD-862k (for V-R) and of QD345-Q (for R-V) is produced by rendering strokes to a 256×256 pixel canvas using the CairoSVG Python library. Table 4 show that for both class and instance level retrieval, the R-V con-

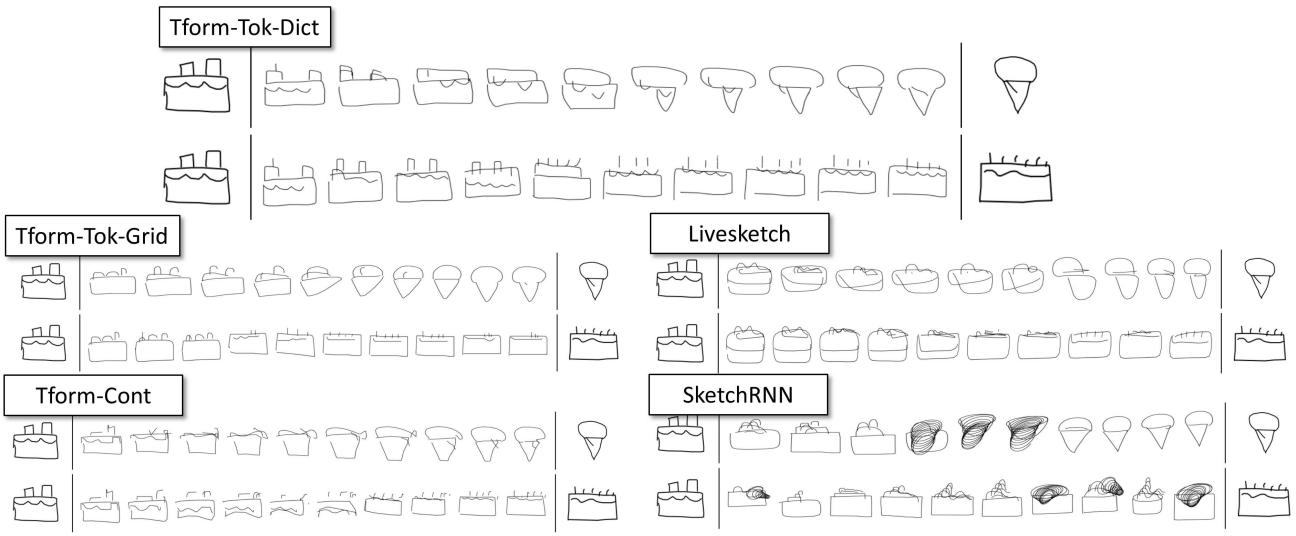

Figure 7. Representative sketch interpolations from each of the five embeddings evaluated in Table 3. For each embedding: (first row) inter-class interpolation from 'birthday cake' to 'ice-cream' and (second row) intra-class interpolation between two 'birthday cakes'.

Figure 8. Representative visual search results over Stock10M indexed by our proposed embedding (TForm-Tok-Dict) for a vector sketch query. The two bottom rows ('animal migration' and 'tree') are failure cases.

figuration outperforms V-R indicating a performance gain due to encoding this large search index using the vector representation. In contrast to other experiments reported, the continuous variant of Sketchformer appears slightly preferred, matching higher for early ranked results for the S-S case – see Fig. 9a for category-level precision-recall curve. Although Transformer outperforms RNN baselines by 1-3% in the V-R case the gain is more limited and indeed the performance over baselines is equivocal in the S-S where the search index is formed of rasterized sketches.

Sketch2Image (S-I) Matching. We evaluate sketch based image retrieval (SBIR) over Stock10M dataset of diverse photos and artworks, as such data is commonly indexed for large-scale SBIR evaluation [6, 3]. We compare against the state of the art SBIR algorithms accepting vector (LiveSketch [3]) and raster (Bui *et al*. [28]) sketched queries. Since no ground-truth annotation is possible for this size of corpus, we crowd-source per-query annotation via Mechanical Turk (MTurk) for the top-k (k=15) results and compute both mAP% and precision@k curve averaged across all QD345-Q query sketches, both with $k = [1-15]$. Table 5 compares performance of our tokenized variants to these baselines, alongside associated Precision@k curves in Fig. 9b. The proposed dictionary learned transformer embedding (TForm-Tok-Dict) delivers the best performance (visual results in Fig. 8).

Partition	Embedding	Intra-	Inter-
Short (10^{th} cent.)	SketchRNN [1]	0.00	2.06
	LiveSketch [3]	25.8	30.9
	TForm-Cont	14.0	6.18
	TForm-Tok-Grid	19.4	17.5
	TForm-Tok-Dict	**31.2**	**33.0**
	Uncertain	8.60	10.3
Mean (50^{th} cent.)	SketchRNN [1]	0.00	0.00
	LiveSketch [3]	25.2	20.2
	TForm-Cont	15.6	16.0
	TForm-Tok-Grid	15.6	19.1
	TForm-Tok-Dict	**35.8**	**35.1**
	Uncertain	7.36	9.57
Long (90^{th} cent.)	SketchRNN [1]	0.00	0.00
	LiveSketch [3]	25.0	21.1
	TForm-Cont	12.5	8.42
	TForm-Tok-Grid	16.7	10.5
	TForm-Tok-Dict	**40.6**	**50.5**
	Uncertain	5.21	9.47

Table 3. User study quantifying interpolation quality for a pair of sketches of same (intra-) or between (inter-) classes. Preference is expressed by 5 independent workers, and results with $> 50\%$ agreement are included. Experiment repeated for short, medium and longer stroke sequences.

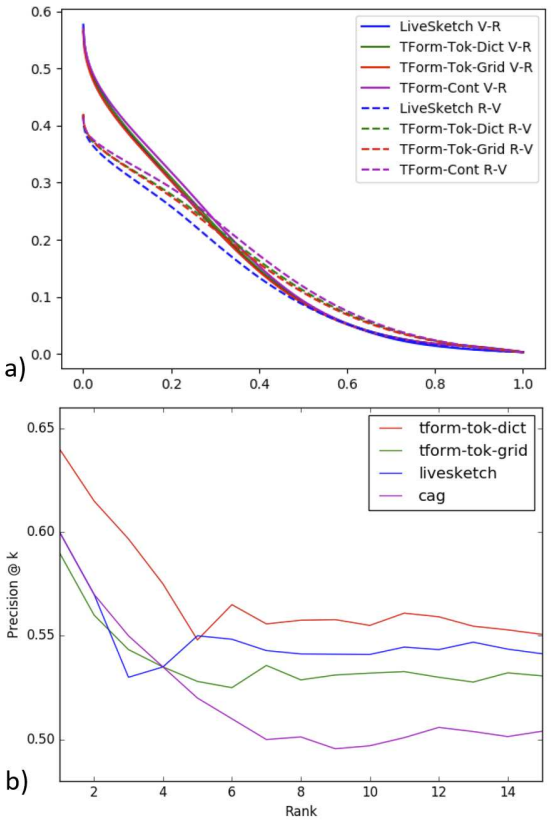

Figure 9. Quantifying search accuracy. a) Sketch2Sketch via precision-recall (P-R) curves for Vector-2-Raster and Raster-2-Vector category-level retrieval. b) Sketch2Image (SBIR) accuracy via precision @ k=[1,15] curve over Stock10M.

Method		Instance	Category
Livesketch [3]	V-R	6.71	20.49
	R-V	7.15	20.93
TForm-Cont	V-R	5.29	22.22
	R-V	6.21	**23.48**
TForm-Tok-Grid	V-R	6.42	21.26
	R-V	**7.38**	22.10
TForm-Tok-Dict	V-R	6.07	21.56
	R-V	7.08	22.51

Table 4. Quantifying the performance of Sketch2Sketch retrieval under two RNN baselines and three proposed variants. We report category- and instance-level retrieval (mAP%).

Method		mAP%
Baseline	LiveSketch [3]	54.80
	CAG [4]	51.97
Proposed	TForm-Tok-Grid	53.75
	TForm-Tok-Dict	**56.96**

Table 5. Quantifying accuracy of Sketchformer for Sketch2Image search (SBIR). Mean average precision (mAP) computed to rank 15 over Stock10M for the QD345-Q query set.

5. Conclusion

We presented Sketchformer: a learned representation for sketches based on the Transformer architecture [10]. Several variants were explored using continuous and tokenized input; a dictionary learning based tokenization scheme delivers classification performance gains of 6% on previous LSTM autoencoder models (SketchRNN and derivatives). We showed interpolation within the embedding yields plausible blending of sketches within and between classes, and that reconstruction (auto-encoding) of sketches is also improved for longer, more complex sketches. Sketchformer was also shown effective as a basis for indexing sketch and image collections for sketch based visual search. Future work could further explore our continuous representation variant, or other variants with more symmetric encoder-decoder structure. We have demonstrated the potential for Transformer networks to learn a multi-purpose representation for sketch, but believe many further applications of Sketchformer exist beyond the three tasks studied here. For example, fusion with additional modalities might enable sketch driven photo generation [41] using complex sketches, or with a language embedding for novel sketch synthesis applications.

Acknowledgments

This work was supported in part by FAPESP (grants 2017/22366-8, 2019/02808-1 and 2018/22482-0), CNPq Fellowship (307973/2017-4), and a charitable donation from Adobe Inc.

References

[1] D. Ha and D. Eck. A neural representation of sketch drawings. In *Proc. ICLR*. IEEE, 2018. 1, 2, 4, 5, 6, 8

[2] P. Xu, Y. Huang, T. Yuan, K. Pang, Y-Z. Song, T. Xiang, and T. Hospedales. Sketchmate: Deep hashing for million-scale human sketch retrieval. In *Proc. CVPR*, 2018. 1, 2, 5

[3] J. Collomosse, T. Bui, and H. Jin. Livesketch: Query perturbations for guided sketch-based visual search. In *Proc. CVPR*, pages 1–9, 2019. 1, 2, 4, 5, 6, 7, 8

[4] T. Bui, L. Ribeiro, M. Ponti, and J. Collomosse. Compact descriptors for sketch-based image retrieval using a triplet loss convolutional neural network. *Computer Vision and Image Understanding (CVIU)*, 2017. 1, 2, 8

[5] Yonggang Qi, Yi-Zhe Song, Honggang Zhang, and Jun Liu. Sketch-based image retrieval via siamese convolutional neural network. In *Proc. ICIP*, pages 2460–2464. IEEE, 2016. 1, 2

[6] J. Collomosse, T. Bui, M. Wilber, C. Fang, and H. Jin. Sketching with style: Visual search with sketches and aesthetic context. In *Proc. ICCV*, 2017. 1, 2, 7

[7] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly drawn bunnies. In *Proc. ACM SIGGRAPH*, 2016. 1, 2

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, v.1*, pages 4171–4186. Association for Computational Linguistics, 2019. 1, 2

[9] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019. 1, 2

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. NeurIPS*. IEEE, 2017. 1, 2, 3, 8

[11] The Quick, Draw! Dataset. https://github.com/googlecreativelab/quickdraw-dataset. Accessed: 2018-10-11. 1, 2, 5

[12] R. Hu, T. Wang, and J. Collomosse. A bag-of-regions approach to sketch based image retrieval. In *Proc. ICIP*, 2011. 2

[13] S. James, R. Hu, T. Wang, and J. Collomosse. Markov random fields for sketch based video retrieval. In *Proc. ICMR*, 2013. 2

[14] Y. Cao, H. Wang, C. Wang, Z. Li, L. Zhang, and L. Zhang. Mindfinder: Interactive sketch based image search on millions of images. In *Proc. ACM MM*, 2010. 2

[15] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? In *Proc. ACM SIGGRAPH*, volume 31, pages 44:1–44:10, 2012. 2

[16] Rui Hu and John Collomosse. A performance evaluation of gradient field HOG descriptor for sketch based image retrieval. *Computer Vision and Image Understanding (CVIU)*, 117(7):790–806, 2013. 2

[17] R. Hu, S. James, T. Wang, and J. Collomosse. Motion-sketch based video retrieval using a trellis levenshtein distance. In *Proc. ICPR*, pages 121–124, 2010. 2

[18] Tu Bui and John Collomosse. Scalable sketch-based image retrieval using color gradient features. In *Proc. ICCV Workshops*, pages 1–8, 2015. 2

[19] Rosália G Schneider and Tinne Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors. *ACM Transactions on Graphics (TOG)*, 33(6):174, 2014. 2

[20] Hua Zhang, Si Liu, Changqing Zhang, Wenqi Ren, Rui Wang, and Xiaochun Cao. Sketchnet: Sketch classification with web images. In *Proc. CVPR*, pages 1105–1113, 2016. 2

[21] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Proc. CVPR*, pages 1386–1393, 2014. 2

[22] Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *Proc. ECCV*, pages 3–20, 2016. 2

[23] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *Proc. ECCV*, pages 241–257, 2016. 2

[24] Qian Yu, Feng Liu, Yi-Zhe Song, Tao Xiang, Timothy M Hospedales, and Chen-Change Loy. Sketch me that shoe. In *Proc. CVPR*, pages 799–807, 2016. 2

[25] T. Bui, L. Ribeiro, M. Ponti, and J. Collomosse. Generalisation and sharing in triplet convnets for sketch based visual search. *CoRR Abs*, arXiv:1611.05301, 2016. 2

[26] M. Wilber, C. Fang, H. Jin, A. Hertzmann, J. Collomosse, and S. Belongie. Bam! the behance artistic media dataset for recognition beyond photography. In *Proc. ICCV*, 2017. 2

[27] O. Seddati, S. Dupont, and S. Mahoudi. Quadruplet networks for sketch-based image retrieval. In *Proc. ICMR*, 2017. 2

[28] T. Bui, L. Ribeiro, M. Ponti, and J. Collomosse. Sketching out the details: Sketch-based image retrieval using convolutional neural networks with multi-stage regression. *Elsevier Computers & Graphics*, 2018. 2, 4, 5, 7

[29] K. Pang, K. Li, Y. Yang, H. Zhang, T. Hospedales, T. Xiang, and Y. Song. Generalising fine-grained sketch-based image retrieval. In *Proc. CVPR*, 2019. 2

[30] S. Dey, P. Riba, A. Dutta, J. Llados, and Y. Song. Doodle to search: Practical zero-shot sketch-based image retrieval. In *Proc. CVPR*, 2019. 2

[31] Yongzhe Xu, Jiangchuan Hu, Kun Zeng, and Yongyi Gong. Sketch-based shape retrieval via multi-view attention and generalized similarity. In *2018 7th International Conference on Digital Home (ICDH)*, pages 311–317. IEEE, 2018. 2

[32] J. Zhu, T. Park, P. Isola, and A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV*, 2017. 2

[33] Y. Song T. Xiang T. Hospedales J. Song, K. Pang. Learning to sketch with shortcut cycle consistency. In *Proc. CVPR*, 2018. 2

[34] Omar Seddati, Stephane Dupont, and Saïd Mahmoudi. Deepsketch 2: Deep convolutional neural networks for partial sketch recognition. In *2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pages 1–6. IEEE, 2016. 2

[35] Xiangxiang Wang, Xuejin Chen, and Zhengjun Zha. Sketchpointnet: A compact network for robust sketch recognition. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 2994–2998. IEEE, 2018. 2

[36] Umar Riaz Muhammad, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M Hospedales. Learning deep sketch abstraction. In *Proc. CVPR*, pages 8014–8023, 2018. 2

[37] Lei Li, Changqing Zou, Youyi Zheng, Qingkun Su, Hongbo Fu, and Chiew-Lan Tai. Sketch-r2cnn: An attentive network for vector sketch recognition. *arXiv preprint arXiv:1811.08170*, 2018. 2

[38] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran. Image transformer. In *Proc. NeurIPS*, 2019. 2

[39] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973. 2, 5

[40] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015. 4

[41] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Gaugan: semantic image synthesis with spatially adaptive normalization. In *ACM SIGGRAPH 2019 Real-Time Live!*, page 2. ACM, 2019. 8