LiveSketch: Query Perturbations for Guided Sketch-based Visual Search

John Collomosse[1 2], Tu Bui[1], and Hailin Jin[2]

[1]Centre for Vision Speech and Signal Processing, University of Surrey
[2]Creative Intelligence Lab, Adobe Research

Abstract

LiveSketch is a novel algorithm for searching large image collections using hand-sketched queries. LiveSketch tackles the inherent ambiguity of sketch search by creating visual suggestions that augment the query as it is drawn, making query specification an iterative rather than one-shot process that helps disambiguate users' search intent. Our technical contributions are: a triplet convnet architecture that incorporates an RNN based variational autoencoder to search for images using vector (stroke-based) queries; real-time clustering to identify likely search intents (and so, targets within the search embedding); and the use of backpropagation from those targets to perturb the input stroke sequence, so suggesting alterations to the query in order to guide the search. We show improvements in accuracy and time-to-task over contemporary baselines using a 67M image corpus.

1. Introduction

Determining user intent from a visual search query remains an open challenge, particularly in sketch based image retrieval (SBIR) over millions of images where a sketched shape can yield plausible yet unexpected matches. For example, a user's sketch of a dog might return a map of the United States that ostensibly resembles the shape (*structure*) drawn, but is not relevant. Free-hand sketches are often incomplete and ambiguous descriptions of desired image content [8]. This limits the ability of sketch to communicate search intent, particularly over large image datasets.

This paper proposes LiveSketch; a novel interactive SBIR technique in which users iterate to refine their sketched query, selecting and integrating sketch embellishments suggested by the system in order to *disambiguate search intent* and so improve the relevance of results (Fig. 1). A core novelty of our approach lies within the method by which visual suggestions are generated, exploiting the reversibility of deep neural networks (DNNs) that are commonly used to encode image features to create the search index in visual search systems [26, 13, 7, 6]. By identifying clusters of likely target intents for the user's search, we reverse the DNN encoder to explain how such clusters could be gen-

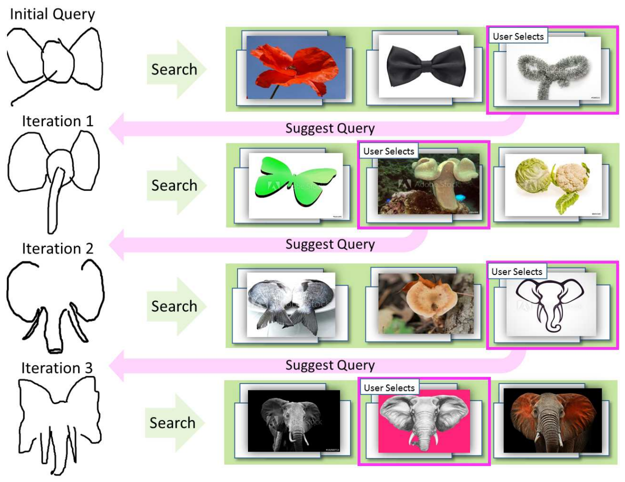


Figure 1. LiveSketch helps disambiguate SBIR for large datasets, where a shape sketched from scratch (top left) can yield results that do not match the users' search intent. LiveSketch iteratively suggests refinements to users' sketched queries to guide the search (Iter.1-3), based on the user indicating relevant clusters of results (right). This interaction disambiguates and quickly guides the search towards results that match users' search intent (subsec. 4.3).

erated by adapting the query. We are inspired by adversarial perturbations (APs); that use backpropagation to generate 'adversarial' image examples [12] that induce object mis-classification [24, 2, 23] to a targeted category. In our context of visual search, we similarly backpropagate to perturb the sketched query from its current state toward one (or more) targets identified in the search embedding by the user. As such, the query becomes a 'living sketch' on the canvas that reacts interactively to intents expressed by the user, forming the basis for subsequent search iterations. The use of a single, live sketch to collaboratively guide the search differs from prior approaches such as ShadowDraw [22] that ghost hundreds of top results on the canvas. We propose three technical contributions:

1) Vector Queries for Sketch based Image Retrieval. We learn a joint search embedding that unifies vector graphic and raster representations of visual structure, encoded by recurrent (RNN) and convnet (CNN) branches of a novel triplet DNN architecture. Uniquely, this embedding enables the retrieval of raster (*e.g.* photo) content using sketched queries encoded as a sequence of strokes. This higher level representation is shown to not only enhance search accuracy (subsec. 4.1) but also enables perturbation of the query to form suggestions, without need for pixel regularization.

2) Guided Discovery of Search Intent. We make use of an auxiliary (semantic) embedding to cluster search results into pools, each representing a candidate search intent. For example, a circle on a stick might return clusters corresponding to balloons, signs, mushrooms. Deriving query suggestions from sketches drawn from these pools guides the user toward relevant content, clarifying intent by supplying contextual information not present in the query.

3) Query Perturbation. We propose an iterative strategy for SBIR query refinement in which the users' query sketch is perturbed to incorporate the appearance of search intent(s) indicated by the user. We cast this as a search for a query perturbation that shifts the encoded query within the search embedding closer toward those selected intent(s), encoding that vector as a loss (in the spirit of APs) that is backpropagated through the DNN to update the sketch.

2. Related Work

Visual search is a long-standing problem within the computer vision and information retrieval communities, where the iterative presentation and refinement of results has been studied extensively as relevance feedback (RF) [29, 21, 20] although only sparsely for SBIR [16]. RF is driven by interactive markup of results at each search iteration. Users tag results as relevant or irrelevant, so tuning internal search parameters to improve results. Our work differs in that we modify the query itself to affect subsequent search iterations; queries may be further augmented by the user at each iteration. Recognizing the ambiguity present in sketched queries we group putative results into semantic clusters and propose edits to the search query for each object class present.

Query expansion (QE) is a automated technique to improve search accuracy from a single, one-off visual query [19, 39, 33, 27] by recursively submitting search results as queries. LiveSketch contrasts with QE as it is an *interactive* system in which query refinements are suggested, and optionally incorporated by the user to help disambiguate search intent; so communicating more information than present in a single, initial sketched query.

Deep learning, specifically CNNs (convnets), have been rapidly adopted for SBIR and more broadly for visual search outperforming classical dictionary learning based models (e.g. bag of words) [30, 3]. Wang et al [34] were arguably the first to explore CNNs for sketched 3D model retrieval via a contrastive loss network mapping sketches to rendered 2D views. Qi *et al.* [25] similarly learned correspondence between sketches and edge maps. Fine-grained SBIR was explored by Yu *et al.* [38] and Sangkloy *et al.* [28] who used a three-branch CNN with triplet loss for learning the cross-domain embedding. Triplet loss models have been used more broadly for visual search e.g. using photographic queries [35, 26, 13]. Bui *et al.* [4, 6] perform cross-category retrieval using a triplet model and currently lead the Flickr15k [15] benchmark for SBIR. Their system was combined with a learned model of visual aesthetics [36] to constrain SBIR using stylistic cues in [7]. All of these prior techniques learn a deep encoder function that maps an image into a point in a metric search embedding where the distance between an image pair correlates to its similarity. Such embeddings can be binarized (*e.g.* via PQ [18]) for scalable search. In prior work search embeddings were learned using rasterized sketches i.e. images, rather than vector representations of sketched strokes. In our approach we adopt the a vector representation for sketches, building upon the SketchRNN variational auto-encoder of Eck *et al.* previously applied to blend [14] and match [37] sketches with sketches. Here we adapt SketchRNN in a more general form for both our interactive search of photographs, and for generating search suggestions, training with the Quickdraw50M dataset [1].

Our work is aligned to ShadowDraw [22] in which ghosts (edge-maps derived from top search results) are averaged and overlaid onto the sketch canvas (similarly, [40] for photo search). However our system differs both in intent and in method. ShadowDraw is intended to teach unskilled users to sketch rather than as a search system in its own right [22]. The technical method also differs - our system uses deep neural networks (DNNs) both for search and for query guidance, and hallucinates a single manipulable sketch rather than a non-edittable cloud of averaged suggestions. This declutters presentation of the suggestions and does not constrain suggestions to the space of existing images in the dataset. Our method produces query suggestions by identifying destination points within the search embedding and employing backpropagation through the deep network (with network weights fixed) in order to update the input query so that it maps to those destination points. The manipulation of input imagery with the goal of affecting change in the output embedding is common in the context of adversarial perturbations (APs) where image pixels are altered to change the classification (softmax) output of a CNN [24, 2]. We are inspired by FGSM [12] which directly backpropagates from classification loss to input pixels in order to induce noise that, whilst near-imperceptible, causes mis-classification with high confidence. Although we also backpropagate, our goal differs in that we aim for observable changes to the query that guide the user in refining their input. Our reimagining of APs for interactive query refinement in visual search is unique.

3. Methodology

LiveSketch accepts a query sketch Q in vector graphics form (as a variable length sequence of strokes), and searches a large ($\sim 10^8$) dataset of raster images $I = \{I_1, ..., I_N\}$. Our two-stream network architecture (Fig. 2) unifies both vector and raster modalities via a common search embedding ($\mathcal{S}$). Sketch and image content are encoded via RNN and CNN branches respectively, unified via 4 fully connected (fc) layers; final layer activations yield $\mathcal{S} \in \Re^{256}$. The end-to-end RNN and CNN paths through the network describe

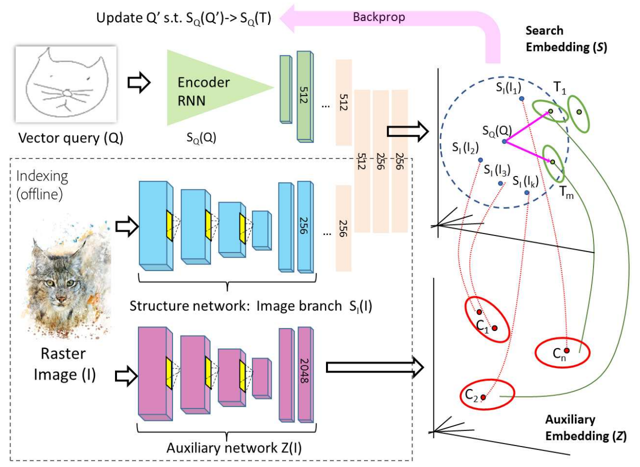


Figure 2. Overview of the proposed SBIR framework. A query sketch (Q, vector graphics form) and images (I, raster form) are encoded into the search embedding $\mathcal{S}$ via RNN and CNN branches, unified via four inner product layers. Images are encoded via $S_I(.)$; the image branch of [6]. Query sketches are encoded via $S_Q(.)$; the encoder stage of Fig. 3. An auxiliary semantic embedding $\mathcal{Z}$ clusters results to help the user pick search target(s) T in the search embedding. In the spirit of adversarial perturbation, the strokes Q are adjusted to minimize $||S_Q(Q) - T_i||_2$ and so evolve the sketch toward the selected target(s).

the pair of encoding functions $S_Q(Q)$ and $S_I(I_i)$ for encoding the visual structure of sketches, and of images, respectively; the process for learning these functions is described in subsec. 3.1. Once learned, the image dataset is indexed (offline) by feeding forward all $I_i \in I$ through $S_I(.)$. At query time, results for a given Q are obtained by ranking on $||S_Q(Q) - S_I(I_i)||_2$ where $||.||_2$ is the L_2 norm.

Fig. 2 provides an overview of our interactive search. Given an initial query Q, images embedded in $\mathcal{S}$ proximate to $S_Q(Q)$ are returned. Whilst these images share the visual structure of Q, the inherent ambiguity of sketch typically results in semantically diverse content, only a subset of which is relevant to the user's search intent. We therefore invite the user to disambiguate their sketch intent via interaction. Search results are clustered within an 'auxiliary' semantic embedding $\mathcal{Z}$. The user assigns relevance weights to a few ($m = 3$) dominant clusters. For each cluster $\{C_1, ..., C_m\}$ in $\mathcal{Z}$, a search target $\{T_1, ..., T_m\}$ is identified in $\mathcal{S}$ (process described in subsec. 3.3). The targets receiving high weighting from the user represent visual structures that we will evolve the existing query sketch Q toward, in order to form a query suggestion (Q') to guide the next search iteration.

Our query is represented in vector graphics form to enable suggestions to be generated via direct modification of the stroke sequence encoded by Q, avoiding the need for complex pixel-domain regularization. LiveSketch updates $Q \mapsto Q'$ such that $S_Q(Q')$ is closer to targets $\{T_1, ..., T_m\}$ than $S_Q(Q)$. Treating the weighted distances between those targets and $S_Q(Q')$ as a loss, we fix $S_Q(.)$ and propagate gradients back via the RNN branch to perturb the input sequence of strokes (subsec. 3.4) and so suggest the modified sketch query. Q' may be further augmented by the user, and submitted for a further iteration of search.

$\mathcal{R} \in \Re^{256}$	Raster embedding [6]*
$\mathcal{V} \in \Re^{512}$	Vector graphics embedding*
$\mathcal{S} \in \Re^{256}$	Joint search embedding (structure)
$\mathcal{Z} \in \Re^{2048}$	Auxiliary embedding (semantic)

Table 1. Summary of the feature embeddings used in LiveSketch; * indicates intermediate embeddings not used in the search index.

3.1. Cross-modal Search Embedding ($\mathcal{S}$)

We wish to learn a cross-modal search embedding in which a sketched query expressed as a variable length sequence of strokes, and an image indexed by the system (*e.g.* a photograph) containing similar visual structure, map to similar points within that embedding. We learn this representation using a triplet network (Fig. 4) comprising an RNN anchor (a) and siamese (*i.e.* identical, shared weights) positive and negative CNN branches (p/n). The RNN and CNN branches encode vector and raster content to intermediate embeddings $\mathcal{V}$ and $\mathcal{R}$ respectively; we describe how these are learned in subsecs 3.1.1-3.1.2. The branches are unified by 4 fully-connected (fc) layers, with weight-sharing across all but the first layer to yield the common search embedding $\mathcal{S}$. Thus the fc layers encode two functions mapping $\mathcal{V} \mapsto \mathcal{S}$ and $\mathcal{R} \mapsto \mathcal{S}$ respectively; we write these $F_V(.)$ and $F_R(.)$ and in subsec. 3.1.3 describe incorporation of these into the pair of end-to-end encoding functions $S_Q(.)$ and $S_I(.)$ for our network (Fig. 2).

3.1.1 Sketch Variational Autoencoder

The RNN branch is a forward-backward LSTM encoder comprising the front half of a variational autoencoder (v.a.e.) for sketch encoding-decoding, adapted from the SketchRNN network of Eck *et al.* [14]. In the SketchRNN v.a.e., a deterministic latent representation (z_0) is learned, alongside parameters of multi-variate Gaussian from which a non-deterministic (n.d.) representation (batch_z) is sampled to drive the decoder and reconstruct the sketch through recurrence conditioned on batch_z. The representation is learned via a combination of reconstruction loss (and a regularization 'KL loss' [17] over the multi-variate parameters), but as proposed [14] can represent only up to a few object classes making it unsuitable for web-scale SBIR.

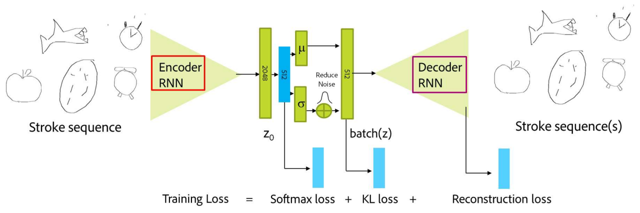


Figure 3. Modified SketchRNN [14] (changes, blue) used to encode/decode stroke sequences via addition of 512-D latent representation and classification loss. Integrates with Fig. 2 (anchor).

We adapt SketchRNN as follows (Fig. 3). We retrain from scratch using 3.5M sketches from Quickdraw50M (QD-3.5M; see Sec. 4) adding a low-dimensional (512-D) bottleneck after z_0 from which batch_z is sampled. We add softmax classification loss to that bottleneck, weighted equally with the original reconstruction and KL loss terms. During training we reduce below 10^{-2} the covariance of the n.d. variate. A query sketch (Q) is coded as a sequence of 3-tuples $Q = [q_1, q_2, ..., q_n]$ where $q_i = (\delta x, \delta y, l)$ representing relative pen movements in $x, y \in \Re^2$ and whether the pen is lifted $l = [0, 1]$; an abbreviated form of the 5-tuple coding in [14]. The intermediate embedding available at the bottleneck ($\mathcal{V} \in \Re^{512}$) is capable of reconstructing sketches across diverse object classes (c.f. Sec.4.2). The encoder forms the anchor of the proposed triplet network (Fig. 4); we denote the encoding and decoding functions as $V_E(Q) \mapsto \mathcal{V}$ and $V_D(\mathcal{V}) \mapsto Q$.

3.1.2 Raster Structure Encoder

To encode raster content, we adopt the architecture of Bui *et al*. [6] for the CNN branch. Their work employs a triplet network with GoogLeNet Inception backbone [32] that unifies sketches (in raster form) and images within a joint search embedding. One important property is the partial sharing of weights between the sketch CNN branch (anchor) and the siamese image CNN (+/-) branches of their triplet network. Once trained, these branches yield two functions: $R_S(.)$ and $R_I(.)$, that map sketched and image content to a joint search embedding. Full details on the multi-stage training of this model are available in [6]; we use their pre-trained model in our work and incorporate their joint embedding as the intermediate embedding $\mathcal{R} \in \Re^{256}$ in our work. Specifically, $R_S(.)$ is used to train our model (Fig. 4, p/n).

3.1.3 Training the Joint Search Embedding

The end-to-end triplet network (Fig. 4) is trained using sketches only; 3.5M sketches (10K $\times$345 object classes) sampled from the public Quickdraw50M dataset [1] (simplified via RDP [9], as in [14]) and rasterized by rendering pen movements as anti-aliased lines of width 1 pixel on a 256×256px canvas. The stroke sequence and the rendering of that sequence are fed through the anchor (a) and positive (p) branches, and a randomly selected rasterized sketch of differing object class to negative branch (n). Weights are optimized via ADAM using triplet loss computed over activations available from the final shared fc layer (the search embedding $\mathcal{S}$):

$$\begin{aligned} L_{\mathrm{train}(a,p,n)} \quad = \quad & [m + ||S_Q(a) - F_R(R_S(p))||\,|_2^2 - \\ & ||S_Q(a) - F_R(R_S(n))||\,|_2^2]_+ \end{aligned} \qquad (1)$$

where $m = 0.2$ is a margin promoting convergence, and $[x]_+$ indicates the non-negative part of x. Training yields weights for the fully connected (fc) layers – recall these are partially shared across the vector (a) and raster (p/n) branches, yielding $F_V(.)$ and $F_R(.)$. The end-to-end functions mapping a

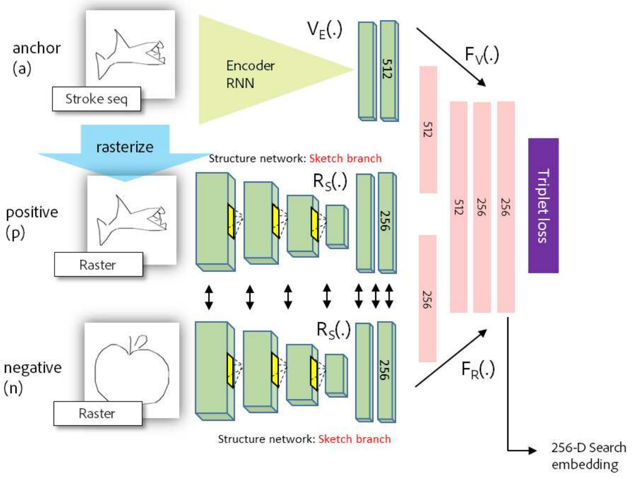


Figure 4. Training the LiveSketch network; an encoder that maps raster and vector content to a common search embedding. The search embedding is trained using raster and vector (stroke sequence) content sampled from QD-3.5M. During training, the CNN branches (p/n) are $R_S(.)$ *i.e.* the sketch branch of [6]. However branch $R_I(.)$ is used at inference time (Fig. 2).

sketched query Q to our common search embedding $\mathcal{S}$ is:

$$S_Q(Q) = F_V(V_E(Q)). \qquad (2)$$

Fig. 5a shows the resulting embedding; raster and vector content representing similar visual structures mix within $\mathcal{S}$ but distinct visual structures form discriminative clusters.

3.2. Search Implementation

Once trained, $S_Q(.)$ forms the RNN path within our search framework (Fig. 2, green) for encoding a vector sketch query Q. The CNN path $S_I(.)$ (Fig. 2, blue) used to index images for search, adopts the *image* branch of [6] (subsec. 3.1.2):

$$S_I(I) = F_R(R_I(I)). \qquad (3)$$

Note substitution of the sketch branch $R_S(.)$ that was used during training (eq.1) for the image branch $R_I(.)$. Both functions map to the same intermediate embedding $\mathcal{R}$, however we index images rather than sketches for SBIR.

3.3. Disambiguating Search Intent

Given a search query Q, a k-NN lookup within $\mathcal{S}$ is performed to identify a set of results $J = [I_1, ..., I_k]$ where $J \subseteq I$ minimising $||S_Q(Q) - S_I(I_i)||_2$; in practice, $||.||_2$ is approximated via product quantization (PQ) [18] for scalability and up to $k = 500$ results are returned. The results are clustered into candidate search intents, and presented to the user for feedback. Clustering is performed within an auxiliary embedding ($\mathcal{Z}$) available from final layer activations of a ResNet50/ImageNet pre-trained CNN. We write this function $Z(I_i)$, pre-computed $\forall I_i \in I$ during indexing.

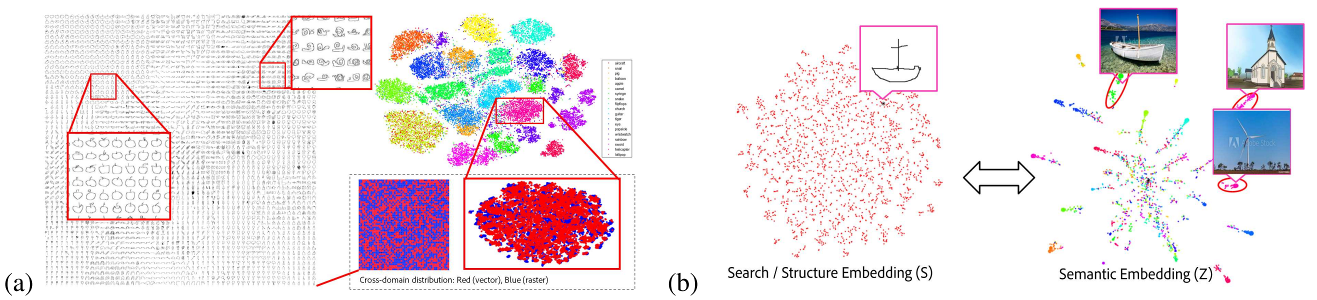


Figure 5. (a) Visualizing the search embedding ($\mathcal{S}$) (for 20/345 random classes sampled from QuickDraw50M); sketches in vector (red) and raster (blue) modalities have been encoded via $S_Q(.)$ and $S_I(.)$ respectively. The learned representation is discriminative on visual structure but invariant to modality. (b) A k-NN search ($L^2, k = 500$) yield search results in $\mathcal{S}$ local to encoded sketch query $S_Q(Q)$; results share similar structure but span diverse semantics *e.g.* a box with a cross atop returns boats, churches, windmills. Results are clustered in auxiliary (semantic) embedding $\mathcal{Z}$ and presented to user for ranking.

3.3.1 Clustering

Images local to $S_Q(Q)$ within the search embedding $\mathcal{S}$ may be semantically diverse; a single visual structure *e.g.* a cross atop a box, may return churches, boats, windmills, etc. However these results will form distinct clusters within $\mathcal{Z}$ (Fig. 5b). We apply affinity propagation [11] to identify the dominant $m = 3$ clusters $C = [c_1, ..., c_m]$ in $\mathcal{Z}$. The algorithm constructs an affinity graph for all image pairs in $(I_a, I_b) \in J \times J$ scoring these:

$$d(I_a, I_b) = ||Z(I_a) - Z(I_b)||_2. \tag{4}$$

Clustering is a greedy process that iteratively constructs C, selecting a best cluster $c_i = I_1, ..., I_k$ from the graph, minimizing $\rho(c_i)$:

$$\rho(c_i) = \sum_{(I_a,I_b)\in c_i \times c_i} d(I_a, I_b) + W(c_i, C). \tag{5}$$

Where $W(C)$ is a penalty term that encourages semantic diversity by discouraging selection of c_i containing images similar to those already in C:

$$W(c_i, C) \propto -log \left(\sum_{(I_a,I_b)\in c_i \times \chi(C)} d(I_a, I_b) \right) \tag{6}$$

where $\chi(c_i)$ represents the set of images already present within clusters in set C.

3.3.2 Identifying Search Targets

For each cluster $i = [1, m]$ we identify a representative image I^*_i closest to the visual structure of the query:

$$I^*_i = \arg\min_{I_j} ||S_Q(Q) - S_I(I_j)||_2; \quad \forall I_j \in C_i. \tag{7}$$

Leveraging the Quickdraw50M dataset (QD-3.5M) of sketches (H), we identify the closest sketch Q^*_i to each representative image:

$$Q^*_i = \min_{q \in H} ||S_Q(q) - S_I(I^*_i)||_2. \tag{8}$$

The set of these sketches $T = \{T_1, ..., T_m\}$, where $T_i = Q^*_i$, represent the set of search targets and the basis for perturbing the user's query (Q) to suggest a new sketch Q' that guides subsequent iterations of search.

3.4. Sketch Perturbations for User Guidance

The search targets T are presented to the user, alongside sliders that enable the relevance of each to be interactively expressed as a set of weights $\Omega = \{\omega_1, ..., \omega_m\}$. We seek a new sketch query Q' that updates the original query Q to resemble the visual structure of those targets, in proportion to these user supplied weights.

For brevity we introduce the following notation. $Q^{V*}_i = V_E(Q^*_i)$ describes each search target within the RNN embedding $\mathcal{V}$. Similarly $Q^{S*}_i = S_Q(Q^*_i)$ describes each target within the search embedding $\mathcal{S}$. We similarly use $Q^V = V_E(Q)$ and $Q^S = S_Q(Q)$ to denote the user's sketch Q in $\mathcal{V}$ and $\mathcal{S}$ respectively. To perturb the sketch we seek Q' (similarly written $Q^{V'}$ and $Q^{S'}$ within those embeddings).

The availability of the v.a.e. decoder (subsec. 3.1.1) enables generation of new sketches (sequences of strokes) conditioned on any point within $\mathcal{V}$. Our approach is to seek $Q^{V'}$ such that $Q' = V_D(Q^{V'})$ may be generated. The task of updating $Q \mapsto Q'$ is therefore one of obtaining $Q^{V'}$ via interpolation between Q^V and targets Q^V_i*, as a function of user supplied weights and targets.

$$Q^{V'} = f(Q^V; \Omega, T). \tag{9}$$

We describe two strategies (instances of f) for computing $Q^{V'}$ from query Q^V (evaluating these in subsec. 4.2).

3.4.1 Linear Interpolation

A naïve solution is to linearly interpolate within the RNN embedding $\mathcal{V}$, *i.e.*:

$$f_{\text{linear}}(Q^V; \Omega, T) = Q^V + \sum_{i=1}^{m} \omega_j (Q^{V*}_i - Q^V) \tag{10}$$

yielding $Q^{V'}$ via eq. 9, from which the sketch suggestion Q' is generated via RNN decoder $Q' = V_D(Q^{V'})$. However,

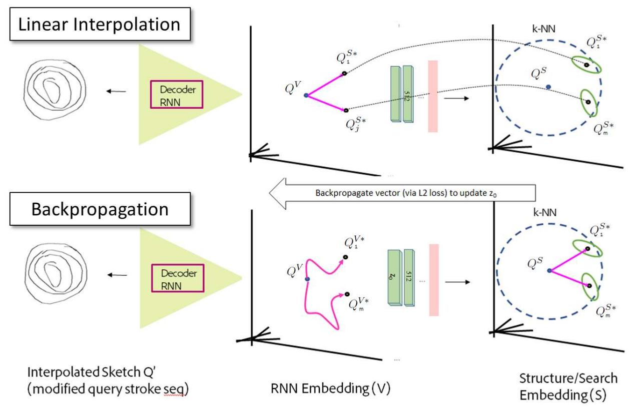


Figure 6. Generating Q': Linear (top) vs. non-linear (bottom) interpolation in RNN space ($\mathcal{V}$); the latter due to backpropagation of loss eq. 12 that updates $Q \mapsto Q'$ s.t. $Q^{S'}$ tends toward the search targets identified local to Q^S by the user. See Fig. 8 for examples.

although Q^S and Q_i^{S*} are local by construction, it is unlikely that Q^V and Q_i^{V*} will be local; nor is the manifold of plausible sketches within $\mathcal{V}$ linear. This can lead to generation of implausible sketches (c.f. Sec. 4.2, Fig. 8).

3.4.2 Back-propagation

We therefore perform a non-linear interpolation in $\mathcal{V}$, minimizing an objective that updates $Q^V \mapsto Q^{V'}$ closer to search target(s) Q_i^{V*} via backprop through the fc layers $F_V(.)$ (eq.2) to reduce the distance between Q^S and Q_i^{S*}.

$$D(Q^{V'}) = \frac{1}{m}\sum_{j=1}^{m} \omega_j ||Q^{S'} - Q_j^{S*}||_2^2. \quad (11)$$

This is analogous to FGSM adversarial perturbation (AP) of images in object recognition [12], where the input the network is modified via backprop to influence its mapping to a classification embedding. In our context, we define a loss based on this distance in $\mathcal{S}$ regularized by the constraint that the original and updated sketch should be nearby in $\mathcal{V}$:

$$L_{\text{AP}}(Q') = D(Q^{V'}) + \alpha||Q^{V'} - Q^V||_2. \quad (12)$$

Weight $\alpha = 0.1$ was empercally set. An optimal $Q^{V'}$ is sought by backpropagation through the fc layer $F_V(.)$:

$$f_{\text{AP}}(Q'; \Omega, T) = \arg\min_{q'} L_{\text{AP}}(V_E(q')). \quad (13)$$

Equipped with sliders to control relevance weights Ω on each of the targets, the user in effect performs a linear interpolation (between $S_Q(Q)$ and T) within $\mathcal{S}$ that causes a non-linear extrapolation from Q^V to output point Q'^V, and ultimately the sketch suggestion via RNN decoder $Q' = F_D(Q'^V)$. Fig. 6 contrasts the linear and non-linear (backprop) approaches; visual examples in Fig. 8.

	Method	Class-level	Instance-level
S-I	LS *(Ours)*	**38.40**	**30.81**
	LS-R	35.26	29.48
	LS-R-I [6]	35.15	27.48
	Sketchy [28]	33.21	27.06
	Bui *et al.* [5]	12.59	8.76
S-S	V-R	34.88	**18.80**
	R-V	29.31	18.29
	V-R-shuffle	**35.94**	15.71
	R-V-shuffle	29.61	18.57

Table 2. Accuracy for sketch based recall of sketches (S-S) and images (S-I); evaluated using class and instance level mAP (%) over 345 vector queries (QD-345). Top: S-I ablations; raster query (LS-R) and raster intermediate embedding (LS-R-I) [6]. Bot. S-S retrieval across query modalities; raster querying vector (R-V) and vector querying raster (V-R); also variants stroke order (-shuffle).

4. Experiments and Discussion

We evaluate the performance of the LiveSketch using the QuickDraw50M dataset [1] and a corpus of 67M stock photo and artwork images (Stock67M) from Adobe Stock[1].

QuickDraw50M is a dataset of 50M hand-drawn sketches crowdsourced via a gamified classification exercise (Quick, Draw!) [1]. Quickdraw50M is well suited to our work due to its class diversity (345 object classes), vector graphics (stroke sequence) format, and the casual/fast, throwaway act of the sketching encouraged in the exercise that reflects typical SBIR user behaviour [8] (vs. smaller, less category-diverse datasets such as TUBerlin/Sketchy that contain higher fidelity sketches drawn with reference to a target photograph [28, 10]). We sample 3.5M sketches randomly with even class distribution from the Quickdraw50M training partition to create training set (**QD-3.5M**; detail in subsec 3.1.3). For sketch retrieval and interpolation experiments we sample 500 sketches per class (173K total) at random from the Quickdraw50M test partition to create an evaluation set **QD-173K**). A query set of sketches (**QD-345**) is sampled from QD-173K, one sketch per object class, to serve as queries for our non-interactive experiments.

Stock67M is a diverse, unannotated corpus of images used to evaluate large-scale SBIR retrieval performance. The dataset was created by scraping harvesting every public, unwatermarked image thumbnails from the Adobe Stock website in late 2016 yielding approximately 67M images at QVGA resolution.

4.1. Evaluating cross-modal search

We evaluate the performance of our cross-modal embedding ($\mathcal{S}$) for sketch based retrieval of sketches and images.
Sketch2Sketch (S-S) Matching. We evaluate the ability of our embedding trained in subsec. 3.1.3 to discriminate between sketched visual structure, invariant to input modality (vector vs. raster). We train our model on QD-3.5M and retrieve sketches from the QD-173K corpus, using QD-345 as

[1]Downloaded from https://stock.adobe.com in late 2016

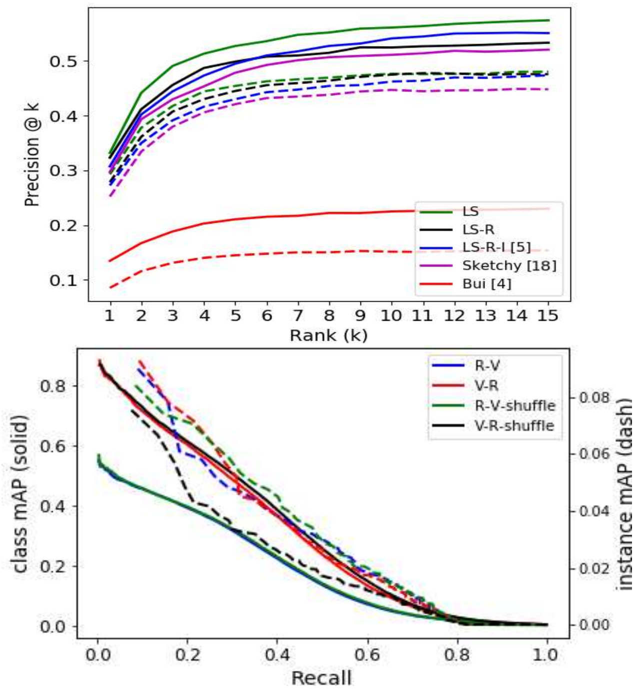


Figure 7. Performance of joint search embedding for retrieval. Top: Sketch2Image (S-I) Precision@k curve for SBIR – see Tbl. 2 for mAP% and key to ablation notation. Bottom: Sketch2Sketch (S-S) cross-modal matching. Class-level (solid) and instance level (dash) mAP-recall curves for Vector-Raster (V-R), Raster-Vector (R-V), and stroke shuffling experiments (-shuffle).

queries. Both vector queries retrieving raster content (V-R) and vice versa (R-V) are explored, using category (class-level) and fine-grain (instance-level) metrics. For the former we consider a retrieved record a match if it matches the sketched object class. For the latter, the exact same sketch must be returned. To run raster variants, a rasterized version of QD-173K is produced by rendering strokes to a 256×256 pixel canvas (see method of subsec. 3.1.3). Sketches from QD-173K are encoded to the search embedding $\mathcal{S}$ from their vector and raster form respectively via functions $S_Q(.)$ and $F_R(R_S(.))$. Fig. 5 visualizes the sketches within the search embedding; similar structures cluster together whilst the vector/raster modalities mix. Tbl. 2 (bot.) and Fig. 7 (bot.) characterize performance; vector queries ($\sim$ 35% mAP) outperform raster ($\sim$ 29% mAP) by a $\sim$ 5% margin. To explore this gain further, we shuffled the ordering of the vector strokes retraining the model from scratch. We were surprised to see comparable performance at class-level, suggesting this gain is due to the spatial continuity inherent to the stroke representation, rather than temporal information. The fractional increase may be due to the shuffling acting as data augmentation enhancing invariance to stroke order. However when at instance level, ordering appears more significant ($\sim$ 3% gain; V-R vs. V-R-shuffle).

Sketch2Image (S-I) Matching. We evaluate the performance of the search embedding for SBIR over Stock67M, using all QD-345 sketch queries (without user interaction in this experiment). Annotation of 67M images for every

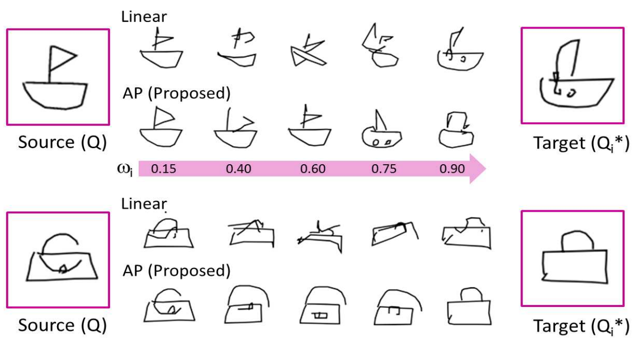
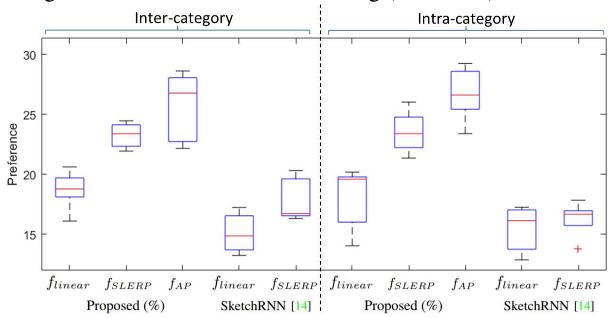


Figure 8. Comparison of sketch interpolation via our backprop (f_{AP}) approach and linear interpolation (f_{linear}) within $\mathcal{V}$, for fine-grain variations of a boat and a bag (c.f. Tbl. 3).



Class	Proposed (%)			SketchRNN [14] (%)	
	f_{linear}	f_{SLERP}	f_{AP}	f_{linear}	f_{SLERP}
Inter-	18.0 ± 2.3	23.5 ± 1.6	26.7 ± 2.0	15.4 ± 1.7	16.3 ± 1.3
Intra-	18.7 ± 1.5	23.0 ± 1.3	25.3 ± 2.1	15.1 ± 1.5	17.9 ± 1.6

Table 3. Perturbation method user study (MTurk) comparing the proposed query perturbation scheme inspired by adversarial perturbations (f_{AP}) vs. linear interpolation variants (f_{linear}, f_{SLERP}).

query is impractical; we instead 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 345 queries for each experiment. The annotation is crowd-source with 5 repetitions. Results are summarized in Tbl. 2 (S-I) and Fig. 7 (top). We perform two ablations to our proposed LiveSketch (LS) system: 1) querying with rasterized versions of the QD-345 queries (-R) using the proposed embedding $\mathcal{S}$; 2) querying with rasterized queries in the intermediate embedding $\mathcal{R}$ (-R-I) which degenerates to [6]; we also baseline against two further recent SBIR techniques: the unshared triplet GoogleNet-V1 architecture proposed by Sangkloy *et al.* [28], and the triplet edgemap approach of Bui *et al.* [5]. We compute class- and instance- level precision for all queries resulting in $345 \times 15 \times 5 =\sim$ 26K MTurk annotations. Our embedding (LS) outperforms all ablations and baselines, with vector query alone contributing significant margin over raster. The addition of fc layers to create cross-modal embedding (-R) slightly improves (importantly, does not degrade) the intermediate raster embedding $\mathcal{R}$ available via [6]. The method significantly outperforms recent triplet SBIR approaches [28, 5]. Note that the S-I and S-S figures are non-comparable; they search different datasets.

Method	Ablations: $seconds(missed)$			Baselines $seconds(missed)$		
	LS (Ours)	LS-NI	LS-NI-R	LS-NI-R-I [6]	Sketchy [28]	Bui *et al.* [5]
Class-level T-T	24.90 (1.33)	38.33 (1.33)	31.74 (0.33)	**19.12 (0.00)**	46.20 (1.00)	40.13 (1.33)
Instance-level T-T	**30.74 (2.00)**	45.43 (1.67)	66.46 (3.67)	95.27 (3.67)	80.28 (2.67)	75.02 (1.33)
Mean Avg. T-T	**27.67 (3.33)**	41.72 (3.00)	45.92 (4.00)	42.69 (3.67)	60.90 (3.67)	54.88 (2.67)

Table 4. Time-to-task user study. Average time to retrieve 20 class- and instance-level search targets (18 participants, 3 per method). Comparing LiveSketch (LS) interactive method with ablations (-NI) non-interactive/one-shot; (-R) raster substitutes vector query; (-I) intermediate structure embedding, and with the three baselines [6, 28, 5]. Times in seconds; parentheses total the averaged missed queries.

4.2. Evaluating Search Suggestions

MTurk was used to evaluate the relative performance of sketch interpolation techniques used to form query suggestions (Q'). We benchmark linear (f_{linear}) and spherical linear (SLERP, [14] f_{SLERP}) interpolation[14] in our RNN embedding $\mathcal{V}$ with the proposed approach f_{AP} inspired by adversarial perturbations, in which the sketch is perturbed via non-linear interpolation in $\mathcal{V}$ due to backpropagation. We also compare to linear and SLERP embedding within the embedding of the original SketchRNN network of Eck *et al*. [14] trained using the same data (QD-3.5M).

MTurkers were presented with a pair of sketches Q and Q' sampled from QD-173K and shown a sequence of 10 sketches produced by each of the 5 interpolation methods. MTurkers were asked to indicate which interpolation looked most natural / human drawn. Each experiment run sampled 300 intra- and 300 inter-category pairs (Q, Q') picked at random from QD-173K. The experiment was repeated 5 times yielding 3k annotations from 25 unique MTurkers.

Tbl. 3 summarizes user study results; an un-paired t-test [31] was run to determine significance (p). Backpropagation (f_{AP}, proposed) outperformed direct linear interpolation (f_{linear}) in V for inter- (18.0% vs. 26.7%, $p < 0.002$) and intra-category (18.7% vs. 25.3, $p < 0.030$) cases (see Fig. 8 for visual examples). Statistically significant results were obtained for f_{SLERP} at $p < 0.03$. In both cases preference was stronger for inter-category interpolation, likely due to non-local nature of (Q, Q') causing linear interpolation to deviate from the manifold of plausible sketches (enforced by f_{AP}). Even linear interpolation in $\mathcal{V}$ enabled more natural interpolations in both inter- and intra-category cases vs. original SketchRNN [14]; but this was significant only for the former.

4.3. Evaluating Iterative Retrieval

We evaluate the efficacy of LiveSketch via a time-to-task experiment in which 18 participants were timed searching for 20 targets using 6 methods. We perform 3 ablations to our method (LS): 1) non-interactive (-NI), users are not offered sketch suggestions; 2) sketches are rasterized (-R) rather than processed as vector queries; 3) as -R but searching within intermediate embedding $\mathcal{R}$ which degenerates to [6] (-R-I). We also baseline against [28, 5].

Fig. 1 provides a representative query, suggestions and clustered results sampled from the study. Tbl. 4 summarizes the results, partitioning across class- and instance- level queries (10 each). Class-level (category) queries prompted the user to search for a specific object ('*cruise ship sailing on the ocean*'). Instance-level (fine-grain) queries prompted the user to search for a specific object with specific pose or visual attributes ('*church with three spires*','*side view of a shark swimming left*'). Timing began on the first stroke drawn and ended when the user was satisfied that the target had been found (self-assessed). If a user took longer than three minutes to find a target, then the search time was capped and noted as a miss (bracketed in Tbl. 4).

Significant decrease in time-to-task ($\sim$ 15s) was observed with the interactive method (LS) over non-interactive variants using vector (LS-NI, LS-NI-R, LS-NI-R) although query modality had negligible effect on mean time to task for non-interactive cases. Baselines performed $\sim 10-20$s slower onexplainable via lower retrieval performance in subsec. 4.1. In all cases, fine-grain queries took longer to identify with greater instances of missed searches – however the margin over class-level searches was only $\sim$ 6s vs. interactive. Whilst category level search time was not enhanced by the proposed method, time taken to produce successful fine-grain sketch queries was significantly reduced by up to 15s over non-interactive ablations and by a factor of 3 over baselines. All 6 methods used a PQ [18] index and took $30-40ms$ to run each query over the Stock67M corpus.

5. Conclusion

We presented an novel architecture that, for the first time, enables search of large image collections using sketched queries expressed as a variable length sequence of strokes (*i.e.* in 'vector' form). The vector modality enables a seocnd contribution; live perturbation of the sketched query strokes to guide the user's search. The search is guided towards likely search intents — interactively specified by user weights attributed to clusters of results returned at each search iteration. Perturbations were generated via backpropagation through the feature encoder network, inspired by adversarial perturbations (FGSM [12]) typically used to attack object classification systems, and applied for the first time here in the context of relevance feedback for visual search. We showed that our interactive system significantly reduces search time over a large (67M) image corpus, particularly for instance-level (fine-grain) SBIR, and that our search embedding (unifying vector/RNN and image/CNN modalities) performs competitively against three baselines [6, 28, 5]. Future work could focus on improvement of the RNN embedding which can still produce implausible sketches for very detailed (high stroke count) drawings.

References

[1] The Quick, Draw! Dataset. `https://github.com/googlecreativelab/quickdraw-dataset`. Accessed: 2018-10-11. 2, 4, 6

[2] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok. Synthesizing robust adversarial examples. *CoRR Abs*, arXiv:1707.07397v2, 2017. 1, 2

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

[4] 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

[5] 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. 6, 7, 8

[6] 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. 1, 2, 3, 4, 6, 7, 8

[7] 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

[8] J P Collomosse, G McNeill, and L Watts. Free-hand sketch grouping for video retrieval. In *Proc. ICPR*, 2008. 1, 6

[9] D. H. Douglas and T. 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*, pages 112–122, 1973. 4

[10] Mathias Eitz, James Hays, and Marc Alexa. How do humans sketch objects? In *Proc. ACM SIGGRAPH*, volume 31, pages 44:1–44:10, 2012. 6

[11] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007. 5

[12] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR Abs*, arXiv:1412.6572, 2015. 1, 2, 6, 8

[13] 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. 1, 2

[14] D. Ha and D. Eck. A neural representation of sketch drawings. In *Proc. ICLR*. IEEE, 2018. 2, 3, 4, 8

[15] 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

[16] S. James and J. Collomosse. Interactive video asset retrieval using sketched queries. In *Proc. CVMP*, 2014. 2

[17] N. Jaques, S. Gu, D. Bahdanau, J. Hernandez-Lobato, R. Turner, and D. Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proc. ICML*. IEEE, 2017. 3

[18] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *Proc. CVPR*, 2010. 2, 4, 8

[19] J.Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proc. CVPR*, 2007. 2

[20] A. Kovashka and K. Grauman. Attribute pivots for guiding relevance feedback in image search. In *Proc. ICCV*, 2013. 2

[21] A. Kovashka, D. Pariks, and K. Grauman. Whittlesearch: Image search with relative attribute feedback. In *Proc. CVPR*, 2012. 2

[22] Y. Lee, C. Zitnick, and M. Cohen. Shadowdraw: real-time user guidance for freehand drawing. In *Proc. SIGGRAPH*, 2011. 1, 2

[23] J. Lu, H. Sibai, E. Fabry, and D. Forsyth. No need to worry about adversarial examples in object detection for autonomous vehicles. *CoRR Abs*, arXiv:1707.03501, 2017. 1

[24] S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *Proc. CVPR*, 2017. 1, 2

[25] 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. 2

[26] 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. 1, 2

[27] F. Radenovic, G. Tolias, and O. Chum. Deep shape matching. In *Proc. ECCV*, 2018. 2

[28] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly drawn bunnies. In *Proc. ACM SIGGRAPH*, 2016. 2, 6, 7, 8

[29] L. Setia, J. Ick, H. Burkhardt, and A. I. Features. Svm-based relevance feedback in image retrieval using invariant feature histograms. In *Proc. ACM Multimedia*, 2005. 2

[30] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proc. ICCV*, 2003. 2

[31] W. Student. The probable error of a mean. *Biometrika*, 6:1–25, 1908. 8

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015. 4

[33] G. Tolias and O. Chum. Asymmetric feature maps with application to sketch based retrieval. In *Proc. CVPR*, 2017. 2

[34] Fang Wang, Le Kang, and Yi Li. Sketch-based 3d shape retrieval using convolutional neural networks. In *Proc. CVPR*, pages 1875–1883, 2015. 2

[35] 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

[36] 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

[37] 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. 2

[38] 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

[39] J. Yuan, S. Bhattacharjee, W. Hong, and X. Ruan. Query adaptive instance search using object sketches. In *Proc. ACM Multimedia*, 2016. 2

[40] J-Y. Zhu, Y-J. Lee, and A. Efros. Averageexplorer: Interactive exploration and alignment of visual data collections. In *Proc. SIGGRAPH*, 2014. 2