Learning Deep Sketch Abstraction

Anonymous CVPR submission

Paper ID 1960

Abstract

Human free-hand sketches have been studied in various contexts including sketch recognition, synthesis and fine-grained sketch-based image retrieval (FG-SBIR). A fundamental challenge for sketch analysis is to deal with drastically different human drawing styles, particularly in terms of abstraction level. In this work, we propose the first stroke-level sketch abstraction model based on the insight of sketch abstraction as a process of trading off between the recognizability of a sketch and the number of strokes used to draw it. Concretely, we train a model for abstract sketch generation through reinforcement learning of a stroke removal policy that learns to predict which strokes can be safely removed without affecting recognizability. We show that our abstraction model can be used for various sketch analysis tasks including: (1) modeling stroke saliency and understanding the decision of sketch recognition models, (2) synthesizing sketches of variable abstraction for a given category, or reference object instance in a photo, and (3) training a FG-SBIR model with photos only, bypassing the expensive photo-sketch pair collection step.

1. Introduction

Sketching is an intuitive process which has been used throughout human history as a communication tool. Due to the recent proliferation of touch-screen devices, sketch is becoming more pervasive: sketches can now be drawn at any time and anywhere on a smartphone using one's finger. Consequently sketch analysis has attracted increasing attention from the research community. Various sketch related problems have been studied, including sketch recognition [9, 47, 46], sketch based image retrieval [11, 18, 45, 40], forensic sketch analysis [26, 33] and sketch synthesis [36, 15, 29].

These studies use free-hand sketches drawn by amateurs based on either a category name, mental recollection, or a reference photo of an object instance. A fundamental challenge in analyzing free-hand sketches is that sketches drawn by different people for the same object category/instance often differ significantly, especially in their levels of abstraction. Fig. 1 shows some examples of both category-level (drawn with only a category name) and instance-level (drawn with a reference photo) sketches. Clearly the large variation in abstraction levels is a challenge for either recognizing the sketch or matching it with a photo. Variation in sketch abstraction level is expected: humans sketch to provide an abstract depiction of an object, and how abstract a sketch is depends both on the task and the individual user's overall and instantaneous preference.

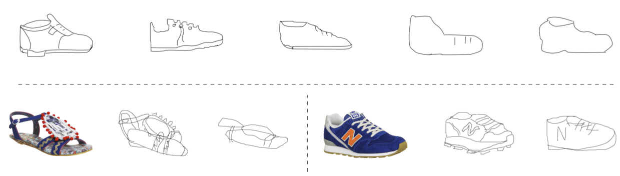


Figure 1: Sketch analysis is difficult because humans draw sketches at very different abstraction levels. Top: different shoe sketches drawn by different people given only the category name. Bottom: sketches are now drawn by different people with a reference photo.

We present the first model of deep sketch abstraction. Our approach to model abstraction is based on the insight that abstraction is a process of tradeoff between recognizability and brevity/compactness (number of strokes). It is thus intuitive that abstraction should vary with task (e.g., sketching for instance- rather than category-level tasks permits less abstraction as the recognition task is more fine-grained), and that abstraction varies between people as their subjective perception (what seems to be recognizable), as might their relative preference for brevity vs identifiability. Based on the same insight, we develop a computational model that learns to abstract concrete input sketches and estimate stroke saliency by finding the most compact subset of input strokes for which the sketch is still recognizable. We consider this similar to the human sketching process: before drawing an object a human has a more detailed mental model of the object, and they work out which details can be safely removed in conveying a compact yet recognizable sketch depiction of the imagined object.

Specifically, we develop a recurrent neural network (RNN) based abstraction model, which learns to measure the importance of each segment and make a decision on whether to skip or keep it. The impact of any given part removal on recognizability is interdependent with which other parts are kept/removed. We model this dependency as a sequential decision making process. Our RNN uses bi-directional gated recurrent units (B-GRU) along with a moving window MLP to capture and extract the contextual information of each sketch-part at each time step. Such a model cannot be learned with conventional supervised learning. We propose a framework for training a sketch abstraction model with reinforcement learning (RL) using a novel reward scheme that uses the classification rank of the sketch at each time step to make rewards more informative.

Using our abstraction model, we can address a number of problems: (1) **Modeling sketch stroke saliency:** We can estimate stroke saliency as a byproduct of learning to produce brief recognizable sketches. (2) **Category-level sketch synthesis with controllable abstraction:** Given an existing category-level sketch synthesizer, our model can be used to control the level of abstraction in the synthesized sketches. (3) **Instance-level photo-to-sketch synthesis:** We propose a new approach to photo →sketch synthesis motivated by human sketching rather than image translation [36, 20]. Given a photo, we extract an edge-map and treat it as a sketch at the most concrete level. Our sketch abstraction model is then applied to abstract the edge-map into a free-hand style sketch. (4) **FG-SBIR without photo-sketch pairs:** The photo-to-sketch synthesis model above is used to synthesize photo-freehand sketch pairs using photo input only. This allows us to train an instance-level fine-grained SBIR (FG-SBIR) model without manual data annotation, and moreover it generates data at diverse abstraction levels so the SBIR model is robust to variable abstraction at runtime.

Our contributions are as follows: (1) For the first time, the problem of stroke-level sketch abstraction is studied. (2) We propose a reinforcement learning framework with novel reward for training a sketch abstraction model (3) Both category- and instance-level sketch synthesis can be performed with controllable abstraction level. We demonstrate that the proposed photo-to-sketch approach is superior than the state-of-the-art alternatives. (4) FG-SBIR can now be tackled without the need to collect photo-sketch pairs. Our experiments on two benchmark datasets show that the resulting FG-SBIR model is quite competitive, thus providing the potential to scale FG-SBIR to an arbitrary number of object categories as long as sufficient photos can be collected.

2. Related Work

Sketch recognition Early work on sketch recognition focused on CAD or artistic drawings [21, 31, 41]. Inspired by the release of the first large-scale free-hand sketch dataset [9], subsequent work studied free-hand sketch recognition [9, 37, 28] using various hand-crafted features together with classifiers such as SVM. Yu *et al*. [47] proposed the first deep convolutional neural network (CNN) designed for sketch recognition which outperformed previous hand-crafted features by a large margin. In this work we do not directly address sketch recognition. Instead we exploit a sketch recognizer to quantify sketch recognizability and generate recognizability-based rewards to train our abstraction model using RL. In particular, we move away from the conventional CNN modeling of sketches [47, 46] where sketches are essentially treated the same as static photos, and employ a RNN-based classifier that fully encodes stroke-level ordering information.

Category-level sketch synthesis Recently there has been a surge of interest in deep image synthesis [13, 39, 25, 34]. Following this trend the first free-hand sketch synthesis model was proposed in [15], which exploits a sequence-to-sequence Variational Autoencoder (VAE). In this model the encoder is a bi-directional RNN that inputs a sketch and outputs a latent vector, and the decoder is an autoregressive RNN that samples output sketches conditioned on a latent vector. They combine RNN with Mixture Density Networks (MDN) [14] in order to generate continuous data points in a sequential way. In this paper, we use the unconditional synthesizer in [15] in conjunction with our proposed abstraction model to synthesize sketches of controllable abstraction level.

Instance-level sketch synthesis A sketch can also be synthesized with a reference photo, giving rise to the instance-level sketch synthesis problem. This is an instance of the well studied cross-domain image synthesis problem. Existing approaches typically adopt a cross-domain deep encoder-decoder model. Cross-domain image synthesis approaches fall into two broad categories depending on whether the input and output images have pixel-level correspondence/alignment. The first category includes models for super-resolution [27], restoration and inpainting [32], which assume pixel-to-pixel alignment. The second category relaxes this assumption and includes models for style transfer (e.g., photo to painting) [22] and cross-domain image-conditioned image generation [44]. Photo-to-sketch is extremely challenging due to the large domain gap and the fact that the sketch domain is generated by humans with variable drawing styles. As a result, only sketch-to-photo synthesis has been studied so far [36, 20, 29]. In this work, we study photo-to-sketch synthesis with the novel approach of treating sketch generation as a photo-to-sketch abstraction process. We show that our method generates more visually appealing sketches than the existing deep cross-domain image translation based approaches such as [36].

Sketch based image retrieval Early effort focused on the category-level SBIR problem [10, 11, 17, 5, 6, 42, 19, 30, 18] whereby a sketch and a photo are considered to be a match as long as they belong to the same category. In contrast, in instance-level fine-grained SBIR (FG-SBIR), they are a match only if they depict the same object instance. FG-SBIR has more practical use, e.g., with FG-SBIR one could use sketch to search to buy a particular shoe s/he just saw on the street [45]. It has thus received increasing attention recently. State-of-the-art FG-SBIR models [45, 35] adopt a multi-branch CNN to learn a joint embedding where photo and sketch domains can be compared. They face two major problems: collecting sufficient matching photo-sketch pairs is tedious and expensive, which severely limits their scalability. In addition, the large variation in abstraction level exhibited in sketches for the same photo (see Fig. 1) also makes the cross-domain matching difficult. In this work, both problems are addressed using the proposed sketch abstraction and photo-to-sketch synthesis models.

Visual abstraction The only work on sketch abstraction is that of [4] where a data-driven approach is used to study style and abstraction in human face sketches. An edge-map is computed and edges are then replaced by similar strokes from a collection of artist sketches. In contrast, we take a model-based approach and model sketch abstraction from a very different perspective: abstraction is modeled as the process of trading off between compactness and recognizability by progressively removing the least important parts. Beyond sketch analysis, visual abstraction has been studied in the photo domain including salient region detection [7], feature enhancement [23], and low resolution image generation [12]. None of these approaches can be applied to our sketch abstraction problem.

3. Methodology

3.1. Sketch abstraction

3.1.1 Sketch representation

Sketches are represented in a vectorized format. Strokes are encoded as a sequence of coordinates, consisting of 3 elements $(\Delta x, \Delta y, p)$, as in [14] for representing human handwriting. We define data-segment as one coordinate and stroke-segment as a group of five consecutive coordinates. Each stroke thus comprises a variable number of stroke-segments.

3.1.2 Problem formulation

We formulate the sketch abstraction process as the sequence of decisions made by an abstraction agent which observes stroke-segments in sequence and decides which to keep or remove. The sequence of strokes may come from a model [15] when *generating* abstract sketches, or a buffer when *simplifying* an existing human sketch or edge-map. The agent is trained with reinforcement learning, and learns to estimate the saliency of each stroke in order to achieve its goal of compactly encoding a recognizable sketch.

The RL framework is described by a Markov Decision Process (MDP), which is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$. Here: $\mathcal{S}$ is the set of all possible states, which are observed by the agent in the form of data-segments representing the sketch and the index pointing at the current stroke-segment being processed. $\mathcal{A} = \{0, 1\}$ is the set of binary action space representing skipping (0) or keeping (1) the current stroke-segment. $\mathcal{T}(s_{t+1}|s_t, a_t)$ is the transition probability density from current state $s_t \in \mathcal{S}$ to next state $s_{t+1} \in \mathcal{S}$ when the agent takes an action $a_t \in \mathcal{A}$. It updates the index and the abstracted sketch so far. $\mathcal{R}(s_t, a_t, s_{t+1})$ is the function describing the reward in transitioning from s_t to s_{t+1} with action a_t. At each time step t, the agent's decision procedure is characterized by a stochastic policy $\pi_\theta = \pi(a_t|s_t, \theta)$ parametrized by θ, which represents the conditional probability of taking action a_t in state s_t.

At the first time step t_1, s_1 corresponds to the data-segments of the complete sketch with index pointing at the first stroke-segment. The agent evaluates s_1 and takes an action a_1 according to its policy π_θ, making a decision on whether to keep or skip the first stroke-segment. The transition $\mathcal{T}(s_2|s_1, a_1)$ says: if $a_1 = 0$ (skip), the next state s_2 corresponds to the updated data-segments which do not contain the skipped stroke-segment and with the index pointing to next stroke-segment. If $a_1 = 1$ (keep), the next state s_2 corresponds to the same data-segments as in s_1 but with the index pointing to the next stroke-segment. This goes on until the last stroke-segment is reached.

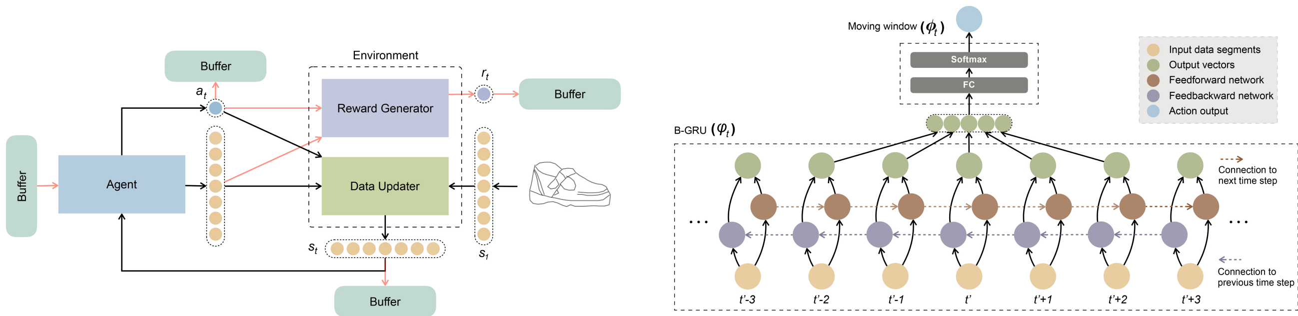
Let $\mathcal{D} = (s_1, a_1, ..., s_M, a_M, s_{M+1})$ be a trajectory of length M, corresponding to the number of stroke-segments in a sketch. Then the goal of RL is to find the optimal policy θ^* that maximizes the expected return (cumulative reward discounted by $\gamma \in [0, 1]$):

$$J(\theta) = \mathbb{E}\left(\sum_{t=1}^{M} \gamma^{t-1} \mathcal{R}(s_t, a_t, s_{t+1}) \,\Big|\, \pi_\theta\right) \quad (1)$$

3.1.3 Model

Our RL-based sketch abstraction model is illustrated in Fig. 2(a). A description of each component follows.

Agent It consists of two modules. In the first *B-GRU module*, data-segments corresponding to state s_t are input sequentially to a recurrent neural network (RNN), i.e., one segment at each time step t' (as shown in Fig. 2(b)). We use bi-directional gated recurrent units [8] (B-GRU) in the RNN to learn and embed past and future information at each



(a) Reinforcement learning framework.

(b) Agent architecture.

Figure 2: Schematic of our sketch abstraction model.

time step t'. This module represents input data in a compact vectorized format φ_t by concatenating the outputs of all time steps. The second *moving window module* consists of a multi-layer perceptron (MLP) with two fully-connected layers. The second layer is softmax activated, and generates probabilities for agent actions ϕ_t. This module slides over the B-GRU module and takes as input those outputs centered at the current stroke-segment under processing, using the index in state s_t. The architecture of our agent is shown in Fig 2(b).

Environment The environment implements state transition and reward generation. The state transition module reads the action a_t and state s_t at each time step t, and transits the environment to state s_{t+1} by updating data-segments and index of the stroke-segment under processing. In case of a skip action, this update consists of eliminating the skipped data-segments, modifying the rest appropriately given the created gap, and moving the index to the next stroke-segment. In case of a keep action, only the index information is updated. The second module is a reward generator which assigns a reward to each state transition. We next describe in detail the proposed reward schemes.

3.1.4 Reward scheme

We want our agent to abstract sketches by dropping the least important stroke-segments while keeping the final remaining sketch recognizable. Therefore our reward is driven by a sketch recognizability signal deduced from the classification result of a multi-class sketch classifier. In accordance with the vectorized sketch format that we use for RL processing, we use a three-layer LSTM [16] classifier trained with cross-entropy loss and Adam optimizer [24]. Using this classifier, we design two types of reward schemes:

Basic reward scheme This reward scheme is designed to encourage high recognition accuracy of the final abstracted sketch while keeping the minimum number of stroke-segments. For a trajectory of length M, the basic reward b_t at each time step t is defined as:

$$R_t = b_t = \begin{cases} +1, & \text{if } t < M \text{ and } a_t = 0 \text{ (skip)} \\ -5, & \text{if } t < M \text{ and } a_t = 1 \text{ (keep)} \\ +100 & \text{if } t = M \text{ and Class}(s_t) \; = \; \text{G} \\ -100 & \text{if } t = M \text{ and Class}(s_t) \; \neq \; \text{G} \end{cases} \tag{2}$$

where G denotes the ground truth class of the sketch, and Class(s_t) denotes the prediction of the sketch classifier on abstracted sketch in s_t. From Eq. 2, it is clear that R_t is defined to encourage compact/abstract sketch generation (positive reward for skip and negative reward for keep action), while forcing the final sketch to be still recognizable (large reward if recognized correctly, large penalty if not).

Ranked reward scheme In this scheme we extend the basic reward by proposing a more elaborate reward computation, aiming to learn the underlying saliency of stroke-segments by integrating the classification rank information at each time step t. The total reward is now defined as:

$$R_t = w_b \, b_t + w_r \, r_t \tag{3}$$

$$r_t = \begin{cases} (w_c \, c_t + w_v \, v_t) \, b_t & \text{if } t < M \\ 0 & \text{if } t = M \end{cases} \tag{4}$$

$$c_t = 1 - \left(\frac{\text{K} - C_t}{\text{K}} \right) \tag{5}$$

$$v_t = 1 - \left(\frac{\text{K} - (C_t - C_{t-1})}{2 \cdot \text{K}} \right) \tag{6}$$

where r_t is the ranked reward, w_b and w_r are weights for the basic and ranked reward respectively, C_t is the predicted rank of ground-truth class and K is the number of sketch classes. The *current ranked reward* c_t prefers the ground-truth class to be highly ranked. Thus improving the rank of the ground truth is rewarded even if the classification is not yet correct – a form of reward-shaping [43]. The *varied ranked reward* v_t is given when the ground-truth class rank improves over time steps. w_c and w_v are weights for current ranked reward and varied ranked reward respectively. For example, assuming $w_b = w_r = 0.5$, at time step t, if $a_t = 0$ (skip), then R_t would be 0.5 when $c_t = 0$, $v_t = 0$, and $R_t = 1.0$ when $c_t = 1$, $v_t = 1$; on the other hand if $a_t = 1$ (keep), then R_t would be -2.5 when $c_t = 0$, $v_t = 0$, and $R_t = -5.0$ when $c_t = 1$, $v_t = 1$.

The basic vs ranked reward weights $w_b \in [0, 1]$ and $w_r \in [0, 1]$ ($w_b + w_r = 1$) are computed dynamically as a functions of time step t. At the first time step $t = 1$, w_r is 0; subsequently it increases linearly to the fixed final w_{r_f} value at the last time step $t = M$. Weights w_c and w_v are static with fixed values, such that $w_c + w_v = 1$.

3.1.5 Training procedure

We use a policy gradient method to find the optimal policy θ^* that maximizes the expected return value defined in Eq. 1. Thus the training consists of sampling the stochastic policy and adjusting the parameters θ in the direction of greater expected return via gradient ascent:

$$\theta \longleftarrow \theta - \eta \, \nabla_\theta J(\theta), \tag{7}$$

where η is the learning rate. In order to have a more robust training, we process multiple trajectories accumulating $\langle s_t, a_t, R_t, s_{t+1} \rangle$ in a Buffer B (see Fig. 2(a), and update parameters θ of the agent every N trajectories.

3.1.6 Controlling abstraction level

Our trained agent can be used to perform abstraction in a given sketch by sampling actions $a_t \in \{1, 0\}$ from the agent's output distribution ϕ_t in order to keep or skip stroke-segments. We attempt to control the abstraction level by varying the temperature parameter of the softmax function in the *moving window module* of our agent. However empirically we found out that it does not give the satisfactory result, so instead we introduce a shift δ in the ϕ_t distribution to obtain different variants of ϕ_t, denoted as ϕ_t^*:

$$\phi_t^* = (\phi_t(a_t = 0) + \delta, \; \phi_t(a_t = 1) - \delta) \tag{8}$$

where, $\phi_t(a_t = 0) + \phi_t(a_t = 1) = 1$ and $\delta \in [-1, 1]$. By varying the δ value we can obtain arbitrary level of abstraction in the output sketch by biasing towards skip or keep.

3.2. Sketch stroke saliency

We use the agent trained with the proposed ranked reward and exploit its output distribution ϕ_t to compute a saliency value $\mathbb{S} \in [0, 1]$ for each stroke in a sketch as:

$$\mathbb{S}_l = \frac{\sum_{t=l_{min}}^{l_{max}} \phi_t(a_t = 1)}{l_{max} - l_{min}} \tag{9}$$

where $l \in \{1, 2, \cdots L\}$ is the stroke index, L is the total number of strokes in a sketch, l_{min} is the time step t corresponding to the first stroke-segment in the stroke with index l and l_{max} corresponding to the last one. Thus strokes which the agent learns are important to keep for obtaining high recognition (or ranking) accuracy are more salient.

3.3. Category-level sketch synthesis

Combining our abstraction model with the VAE RNN category-level sketch synthesis model in [15], we obtain a sketch synthesis model with controllable abstraction. Specifically, once the synthesizer is trained to generate sketches for a given category, we use it to generate a sketch of that category. This is then fed to our abstraction model, which can generate different versions of the input sketch at the desired abstraction level as explained in Sec. 3.1.6.

3.4. Photo to sketch synthesis

Based on our abstraction model, we propose a novel photo-to-sketch synthesis model that is completely different from prior cross-domain image synthesis methods [36, 20] based on encoder-decoder training. Our approach consists of the following steps (Fig. 3). (1) Given a photo p, its edge-map e_p is extracted using an existing edge detection method [48]. (2) We do not use a threshold to remove the noisy edges as in [48]. Instead, we keep the noisy edge detector output as it is and use a line tracing algorithm [2] to convert the raster image to a vector format, giving vectorized edge-maps v_p. (3) Since contours in human sketch are much less smooth than those in a photo edge-map, we apply non-linear transformations/distortions to v_p both at the stroke and the whole-sketch (global) level. At global-level, these transformations include rotation, translation, rescaling, and skew both along x-axis and y-axis. At stroke-level they include translation and jittering of stroke curvature. After these distortions, we obtain d_p, which has rougher contours as in a human free-hand sketch (see Fig. 3). (4) The distorted edge-maps are then simplified to obtain s_p to make them more compatible with the type of free-hand sketch data on which our abstraction model is trained. This consists of fixed-length re-sampling of the vectorized representation to reduce the number of data-segments. (5) After all these preprocessing steps, s_p is used as input to our abstraction model to generate abstract sketches corresponding to the input photo p. Before that, the abstraction model is fine-tuned on pre-processed edge-maps s_p.The code for our photo-to-sketch synthesis model together with code for other models introduced in this paper will be made available in the first author's website to ensure reproducibility.

3.5. Fine-grained SBIR

Armed with the proposed sketch abstraction model and the photo-to-sketch synthesis model presented in Sec. 3.4,

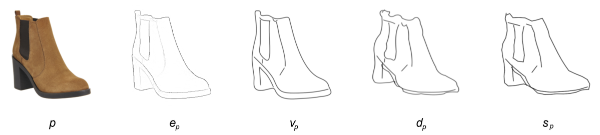


Figure 3: Pre-processing before photo-to-sketch synthesis.

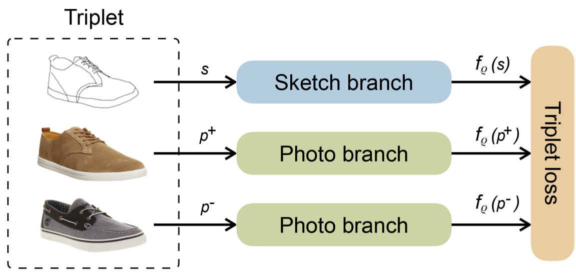


Figure 4: The FG-SBIR model [45].

we can now train a FG-SBIR given photos only.

Given a set of training object photo images, we take each photo p and generate its simplified edge-map s_p. This is then fed into the abstraction model to get three levels of abstraction a_p^1, a_p^2 and a_p^3, by setting δ to -0.1, 0.0 and $+0.1$ respectively (see Eq. 8). This procedure provides three sketches for each simplified edge-map of a training photo, which can be treated as photo-sketch pairs for training a FG-SBIR model. Concretely, we employ the triplet ranking model [45] illustrated in Fig. 4. It is a three-branch Siamese CNN. The input to the model is a triplet including a query sketch s, a positive photo p^+ and negative photo p^-. The network branches aim to learn a joint embedding for comparing photos and sketch such that the distance between s and p^+ is smaller than that between s and p^-. This leads to a triplet ranking loss:

$$\begin{aligned} L_\varrho\left(s, p^+, p^-\right) = \max(0, \Delta + D\left(f_\varrho\left(s\right), f_\varrho\left(p^+\right)\right) \\ -D\left(f_\varrho\left(s\right), f_\varrho\left(p^-\right)\right)) \end{aligned} \tag{10}$$

where ϱ denotes the model parameters, $f_\varrho(\cdot)$ denotes the output of the corresponding network branch, $D(\cdot,\cdot)$ denotes Euclidean distance between two input representations and Δ is the required margin between the positive query and negative query distance. During training we use s_p, a_p^1, a_p^2 and a_p^3 with various distortions (see Sec. 4.4) in turn as the query sketch s. The positive photo p^+ is the photo used to synthesize the sketches, and the negative photo is any other training photo of a different object.

During testing, we have a gallery of test photos which have no overlap with the training photos (containing completely different object instances), and the query sketch now is a real human free-hand sketch. To deal with the variable abstraction in human sketches (see Fig. 1), we also apply our sketch abstraction model to the query test sketch and generate three abstracted sketches as we did in the training stage. The four query sketches are then fed to the trained FG-SBIR model and the final result is obtained by score-level fusion over the four sketches.

4. Experiments

4.1. Sketch abstraction

Datasets We use QuickDraw [15] to train our sketch abstraction model. It is the largest free-hand sketch dataset to date. We select 9 categories (cat, chair, face, fire-truck, mosquito, owl, pig, purse, shoe) with 75000 sketches in each category, using 70000 for training and the rest for testing.

Implementation details Our code is written in Tensorflow [3]. We implement the B-GRU module of the agent using a single layered B-GRU with 128 hidden cells, which is trained with a learning rate η of 0.0001. The RL environment is implemented using standard step and reset functions. In particular, the step function includes the data updater and reward generator module. The sketch classifier used to generate reward is a three-layer LSTM, each layer containing 256 hidden cells. We train the classifier on the 9 categories using cross-entropy loss and Adam optimizer, obtaining an accuracy of 97.00% on the testing set. The parameters of the ranked reward scheme (see Sec. 3.1.4) are set to: $w_{rf} = 0.5$, $w_c = 0.8$ and $w_v = 0.2$.

Baseline We compare our abstraction model with random skipping of stroke-segments from each sketch so that the number of retained data-segments is equal in both models.

Results In this experiment, we take the human free-hand sketches in the test set of the 9 selected QuickDraw categories and generate three versions of the original sketches with different abstraction levels. These are obtained by setting the model parameter δ to -0.1, 0.0 and $+0.1$ respectively (Eq. 8). Some qualitative results are shown in Fig. 5. It can be seen that the abstracted sketches preserve the most distinctive parts of the sketches. For quantitative evaluation, we feed the three levels of abstracted sketches to the sketch classifier trained using the original sketches in the training set and obtain the recognition accuracy. The results in Table 1 show that the original sketches in the test set has 64.79 data segments on average. This is reduced to 51.31, 43.33, and 39.48 using our model with different values of δ. Even at the abstraction level 3 when around 40% of the original data segments have been removed, the remaining sketches can still be recognized at a high accuracy of 70.40%. In contrast, when similar amount of data segments are randomly removed (Baseline), the accuracy is 6.20% lower at 64.20%. This shows that the model has learned which segments can be removed with least impact on recognizability. Table 1 also compares the proposed ranked reward scheme (Eq. 4)

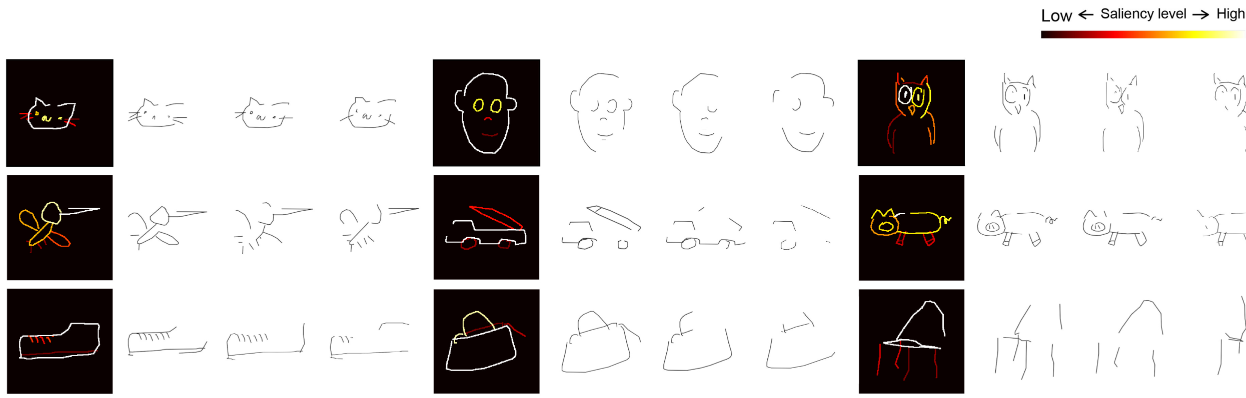


Figure 5: Examples of sketch abstraction and stroke saliency. For each object, the input human sketch annotated with stroke saliency (color coded) computed by model is shown with black background. Three corresponding sketches of different abstraction level (level 1 to 3, left to right) obtained with our model are shown with white background. Best viewed in color.

		#DataSegments	Accuracy
Full Sketch		64.79	97.00%
1st Level Abstraction ($\delta = -0.1$)	Baseline	51.00	85.00%
	Basic Reward	51.12	87.60%
	Ranked Reward	51.31	**88.20%**
2nd Level Abstraction ($\delta = 0.0$)	Baseline	43.00	74.60%
	Basic Reward	43.09	78.80%
	Ranked Reward	43.33	**80.80%**
3rd Level Abstraction ($\delta = +0.1$)	Baseline	39.00	64.20%
	Basic Reward	39.37	68.00%
	Ranked Reward	39.48	**70.40%**

Table 1: Recognizability of abstracted human sketches.

		#DataSegments	Accuracy
Full Sketch		69.61	99.6%
1st Level Abstraction ($\delta = -0.1$)	Baseline	50.00	89.96%
	Basic Reward	50.43	92.60%
	Ranked Reward	50.08	**94.20%**
2nd Level Abstraction ($\delta = 0.0$)	Baseline	44.00	80.20%
	Basic Reward	44.13	88.40%
	Ranked Reward	44.32	**90.80%**
3rd Level Abstraction ($\delta = +0.1$)	Baseline	37.00	69.20%
	Basic Reward	37.15	73.20%
	Ranked Reward	37.56	**79.40%**

Table 2: Recognizability of category-level synthesized sketches.

with the Basic Reward (Eq. 2). It is evident that the ranked reward scheme is more effective.

Measuring sketch stroke saliency Using Eq. 9, we can compute a saliency value $\mathbb{S}$ for each stroke in a sketch, indicating how it contributes towards the overall recognizability of the sketch. Some example stroke saliency maps obtained on the test set are shown in Fig. 5. We observe that high saliency strokes correspond to the more distinctive visual characteristics of the object category. For instance, for shoe, the overall contour is more salient than the shoe-laces because many shoes in the dataset do not have shoe-laces. Similarly, for face, the outer contour is the most distinctive part, followed by eyes and then nose and mouth – again, different people sketch the nose and mouse very differently; but they are more consistent in drawing the outer contour and eyes. These results also shed some light into how deep sketch recognition models make their decisions, providing an alternative to gradient-based classifier-explanation approaches such as [38].

4.2. Sketch synthesis

We train a sketch synthesis model as in [15] for each of the 9 categories, and combine it with our abstraction model

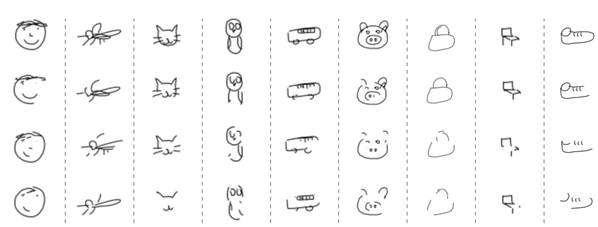


Figure 6: Examples of synthesized sketches at different abstraction levels. Top to bottom: increasing abstraction levels.

(Sec. 4.1) to generate abstract versions of the synthesized sketches. Again, we compare our abstraction results with the same random removal baseline. From the quantitative results in Table 2, we can draw the same set of conclusions: the synthesized sketches are highly recognizable even at the most abstract level, and more so than the sketches generated with random segment removal. Fig. 6 shows some examples of synthesized sketches at different abstraction levels.

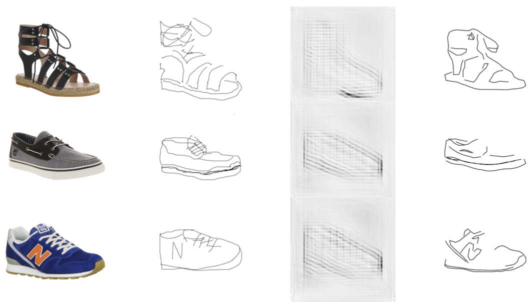


Figure 7: Examples of synthesized sketches using [36] (third col) and ours (fourth) vs human sketch (second).

4.3. Photo to sketch synthesis

Dataset We use the QMUL Shoe-V2 dataset [1]. It is the largest single-category FG-SBIR dataset with 1800 training and 200 testing photo-sketch pairs.

Implementation details As described in Sec. 3.4, we fine-tune our abstraction model, previously trained on the 9 classes of QuickDraw dataset, on the simplified edge-maps s_p of the training photos from Shoe-V2.

Baseline We compare our model with our implementation of the cross-domain deep encoder-decoder based synthesis model in [36]. Note that although it is designed for synthesis across any direction between photo and sketch, only sketch-to-photo synthesis results are shown in [36].

Results We show some examples of the synthesized sketches using our model and [36] in Fig. 7. We observe that our model produces much more visually appealing sketches than the ones obtained using [36], which is very blurry and seems to suffer from mode collapse. This is not surprising: the dramatic domain gaps and the mis-alignment between photo and sketch makes a deep encoder-decoder model such as [36] unsuitable. Furthermore, treating a sketch as a 2D matrix of pixels is also inferior to treating it as a vectorized coordinate list as in our model.

4.4. Fine-grained SBIR

Dataset Apart from Shoe-V2, we also use QMUL Chair-V2, with 200 training and 158 testing photo-sketch pairs.

Implementation details As described in Sec. 4.3, we generate 5 distortion representations d_p^m, $m \in \{1,2,3,4,5\}$, for each input vectorized edge-map v_p. We then use all $a_p^{m,n}$ representations and simplified edge-maps s_p^m to train the state of the art FG-SBIR model [45].

Baseline Apart from comparing with the same model [45] trained with the annotated photo-to-sketch pairs ('Upper Bound'), we compare with two baselines using the same FG-SBIR model but trained with different synthesized sketches. Baseline1 is trained with synthesized sketches using the model in [36]. Baseline2 uses the simplified edge-maps s_p^m directly as replacement for human sketches.

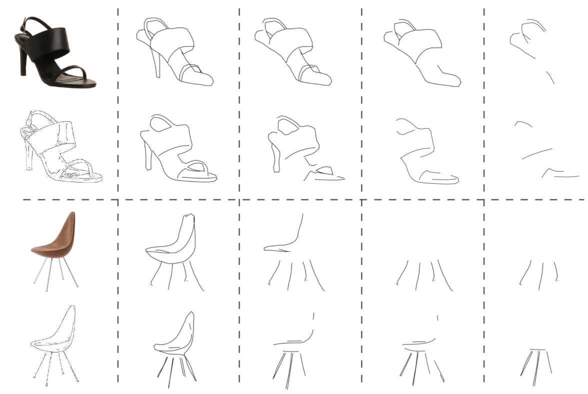


Figure 8: Human and synthesized sketches at different abstraction level used in the FG-SBIR experiments. For each object: First row: photo, sketch and the abstracted sketches. Second row: edge-map and synthesized sketches.

Results Table 3 shows that the model trained with synthesized sketches from our photo-to-sketch synthesizer is quite competitive, e.g., on chair, it is only 7.12% lower on Top 1 accuracy. It decisively beats the model trained with sketches synthesized using [36]. The gap over Baseline2 indicates that the abstraction process indeed makes the generated sketches more like the human sketches. Some qualitative results are shown in Fig. 8. Note the visual similarity between synthesized sketches at different abstraction levels and the corresponding abstracted human sketches. They are clearly more similar at the more abstract levels, explaining why it is important to include sketches at different abstraction levels during both training and testing.

	Shoe-V2		Chair-V2	
Method	Top1	Top10	Top1	Top10
Baseline1 [36]	8.86%	32.28%	31.27%	78.02%
Baseline2	16.67%	50.90%	34.67%	73.99%
Ours	**21.17%**	**55.86%**	**41.80%**	**84.21%**
Upper Bound	34.38%	79.43%	48.92%	90.71%

Table 3: FG-SBIR results. Top 1 and 10 matching accuracy.

5. Conclusion

We have for the first time proposed a stroke-level sketch abstraction model. Given a sketch, our model learns to predict which strokes can be safely removed without affecting overall recognizability. We proposed a reinforcement learning framework with a novel rank-based reward to enforce stroke saliency. We showed the model can be used to address a number of existing sketch analysis tasks. In particular, we demonstrated that a FG-SBIR model can now be trained with photos only. In future work we plan to make this model more practical by extending it to work with edge-maps in the wild.

References

[1] http://sketchx.eecs.qmul.ac.uk. 8

[2] Imagemagick studio, llc. https://www.imagemagick.org. 5

[3] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org, 2015. 6

[4] I. Berger, A. Shamir, M. Mahler, E. Carter, and J. Hodgins. Style and abstraction in portrait sketching. *TOG*, 2013. 3

[5] Y. Cao, C. Wang, L. Zhang, and L. Zhang. Edgel index for large-scale sketch-based image search. In *CVPR*, 2011. 3

[6] Y. Cao, H. Wang, C. Wang, Z. Li, L. Zhang, and L. Zhang. Mindfinder: interactive sketch-based image search on millions of images. In *ACM*, 2010. 3

[7] M.-M. Cheng, J. Warrell, W.-Y. Lin, S. Zheng, V. Vineet, and N. Crook. Efficient salient region detection with soft image abstraction. In *ICCV*, 2013. 3

[8] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, 2014. 3

[9] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *TOG*, 2012. 1, 2

[10] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. An evaluation of descriptors for large-scale image retrieval from sketched feature lines. *Computers & Graphics*, 2010. 3

[11] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *TVCG*, 2011. 1, 3

[12] T. Gerstner, D. DeCarlo, M. Alexa, A. Finkelstein, Y. Gingold, and A. Nealen. Pixelated image abstraction with integrated user constraints. *Computers & Graphics*, 2013. 3

[13] I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 2

[14] A. Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013. 2, 3

[15] D. Ha and D. Eck. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477*, 2017. 1, 2, 3, 5, 6, 7

[16] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 1997. 4

[17] R. Hu, M. Barnard, and J. Collomosse. Gradient field descriptor for sketch based retrieval and localization. In *ICIP*, 2010. 3

[18] R. Hu and J. Collomosse. A performance evaluation of gradient field hog descriptor for sketch based image retrieval. *CVIU*, 2013. 1, 3

[19] R. Hu, T. Wang, and J. Collomosse. A bag-of-regions approach to sketch-based image retrieval. In *ICIP*, 2011. 3

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 2, 5

[21] M. F. A. Jabal, M. S. M. Rahim, N. Z. S. Othman, and Z. Jupri. A comparative study on extraction and recognition method of cad data from cad drawings. In *ICIME*, 2009. 2

[22] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016. 2

[23] H. Kang, S. Lee, and C. K. Chui. Flow-based image abstraction. *TVCG*, 2009. 3

[24] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, 2014. 4

[25] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling. Improving variational inference with inverse autoregressive flow. *NIPS*, 2016. 2

[26] B. Klare, Z. Li, and A. K. Jain. Matching forensic sketches to mug shot photos. *TPAMI*, 2011. 1

[27] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CVPR*, 2017. 2

[28] Y. Li, T. M. Hospedales, Y.-Z. Song, and S. Gong. Free-hand sketch recognition by multi-kernel feature learning. *CVIU*, 2015. 2

[29] Y. Li, Y.-Z. Song, T. M. Hospedales, and S. Gong. Free-hand sketch synthesis with deformable stroke models. *IJCV*, 2017. 1, 2

[30] Y.-L. Lin, C.-Y. Huang, H.-J. Wang, and W. Hsu. 3d subquery expansion for improving sketch-based multi-view image retrieval. In *ICCV*, 2013. 3

[31] T. Lu, C.-L. Tai, F. Su, and S. Cai. A new recognition model for electronic architectural drawings. *CAD*, 2005. 2

[32] M. Mathieu, C. Couprie, and Y. LeCun. Context encoders: Feature learning by inpainting. In *ICLR*, 2016. 2

[33] S. Ouyang, T. Hospedales, Y.-Z. Song, and X. Li. Cross-modal face matching: beyond viewed sketches. In *ACCV*, 2014. 1

[34] S. Reed, A. v. d. Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, D. Belov, and N. de Freitas. Parallel multiscale autoregressive density estimation. *ICML*, 2017. 2

[35] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: learning to retrieve badly drawn bunnies. *TOG*, 2016. 3

[36] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays. Scribbler: Controlling deep image synthesis with sketch and color. *CVPR*, 2017. 1, 2, 3, 5, 8

[37] R. G. Schneider and T. Tuytelaars. Sketch classification and classification-driven analysis using fisher vectors. *TOG*, 2014. 2

[38] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 7

[39] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther. Ladder variational autoencoders. In *NIPS*, 2016. 2

[40] J. Song, Y. Qian, Y.-Z. Song, T. Xiang, and T. Hospedales. Deep spatial-semantic attention for fine-grained sketch-based image retrieval. In *CVPR*, 2017. 1

[41] P. Sousa and M. J. Fonseca. Geometric matching for clip-art drawing retrieval. *VCIR*, 2009. 2

[42] C. Wang, Z. Li, and L. Zhang. Mindfinder: image search by interactive sketching and tagging. In *WWW*, 2010. 3

[43] E. Wiewiora. *Reward Shaping*, pages 863–865. Springer US, Boston, MA, 2010. 4

[44] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. Kweon. Pixel-level domain transfer. In *ECCV*, 2016. 2

[45] Q. Yu, F. Liu, Y.-Z. SonG, T. Xiang, T. Hospedales, and C. C. Loy. Sketch me that shoe. In *CVPR*, 2016. 1, 3, 6, 8

[46] Q. Yu, Y. Yang, F. Liu, Y.-Z. Song, T. Xiang, and T. M. Hospedales. Sketch-a-net: A deep neural network that beats humans. *IJCV*, 2017. 1, 2

[47] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales. Sketch-a-net that beats humans. *BMVC*, 2015. 1, 2

[48] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 5